X Window System Protocol
X Consortium Standard
X Version 11, Release 6.9/7.0

Robert W. Scheifler
X Consortium, Inc.
X Window System is a trademark of The Open Group.


Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE OPEN GROUP BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of the Open Group shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from the Open Group.
Acknowledgments

The primary contributors to the X11 protocol are:

Dave Carver (Digital HPW)
Branko Gerovac (Digital HPW)
Jim Gettys (MIT/Project Athena, Digital)
Phil Karlton (Digital WSL)
Scott McGregor (Digital SSG)
Ram Rao (Digital UEG)
David Rosenthal (Sun)
Dave Winchell (Digital UEG)

The implementors of initial server who provided useful input are:

Susan Angebranndt (Digital)
Raymond Drewry (Digital)
Todd Newman (Digital)

The invited reviewers who provided useful input are:

Andrew Cherenson (Berkeley)
Burns Fisher (Digital)
Dan Garfinkel (HP)
Leo Hourvitz (Next)
Brock Krizan (HP)
David Laidlaw (Stellar)
Dave Mellinger (Interleaf)
Ron Newman (MIT)
John Ousterhout (Berkeley)
Andrew Palay (ITC CMU)
Ralph Swick (MIT)
Craig Taylor (Sun)
Jeffery Vroom (Stellar)

Thanks go to Al Mento of Digital’s UEG Documentation Group for formatting this document.

This document does not attempt to provide the rationale or pragmatics required to fully understand the protocol or to place it in perspective within a complete system.

The protocol contains many management mechanisms that are not intended for normal applications. Not all mechanisms are needed to build a particular user interface. It is important to keep in mind that the protocol is intended to provide mechanism, not policy.

Robert W. Scheifler
X Consortium, Inc.
1. Protocol Formats

Request Format
Every request contains an 8-bit major opcode and a 16-bit length field expressed in units of four bytes. Every request consists of four bytes of a header (containing the major opcode, the length field, and a data byte) followed by zero or more additional bytes of data. The length field defines the total length of the request, including the header. The length field in a request must equal the minimum length required to contain the request. If the specified length is smaller or larger than the required length, an error is generated. Unused bytes in a request are not required to be zero. Major opcodes 128 through 255 are reserved for extensions. Extensions are intended to contain multiple requests, so extension requests typically have an additional minor opcode encoded in the second data byte in the request header. However, the placement and interpretation of this minor opcode and of all other fields in extension requests are not defined by the core protocol. Every request on a given connection is implicitly assigned a sequence number, starting with one, that is used in replies, errors, and events.

Reply Format
Every reply contains a 32-bit length field expressed in units of four bytes. Every reply consists of 32 bytes followed by zero or more additional bytes of data, as specified in the length field. Unused bytes within a reply are not guaranteed to be zero. Every reply also contains the least significant 16 bits of the sequence number of the corresponding request.

Error Format
Error reports are 32 bytes long. Every error includes an 8-bit error code. Error codes 128 through 255 are reserved for extensions. Every error also includes the major and minor opcodes of the failed request and the least significant 16 bits of the sequence number of the request. For the following errors (see section 4), the failing resource ID is also returned: Colormap, Cursor, Drawable, Font, GContext, IDChoice, Pixmap, and Window. For Atom errors, the failing atom is returned. For Value errors, the failing value is returned. Other core errors return no additional data. Unused bytes within an error are not guaranteed to be zero.

Event Format
Events are 32 bytes long. Unused bytes within an event are not guaranteed to be zero. Every event contains an 8-bit type code. The most significant bit in this code is set if the event was generated from a SendEvent request. Event codes 64 through 127 are reserved for extensions, although the core protocol does not define a mechanism for selecting interest in such events. Every core event (with the exception of KeymapNotify) also contains the least significant 16 bits of the sequence number of the last request issued by the client that was (or is currently being) processed by the server.

2. Syntactic Conventions
The rest of this document uses the following syntactic conventions.

- The syntax {...} encloses a set of alternatives.
- The syntax [...] encloses a set of structure components.
- In general, TYPES are in uppercase and AlternativeValues are capitalized.
- Requests in section 9 are described in the following format:
**RequestName**

\[ arg_1 : \text{type}_1 \]
\[ \ldots \]
\[ arg_N : \text{type}_N \]

\[ \rightarrow \]

\[ \text{result}_1 : \text{type}_1 \]
\[ \ldots \]
\[ \text{result}_M : \text{type}_M \]

Errors: kind_1, ..., kind_K

Description.

If no \( \rightarrow \) is present in the description, then the request has no reply (it is asynchronous), although errors may still be reported. If \( \rightarrow \rightarrow \) is used, then one or more replies can be generated for a single request.

- Events in section 11 are described in the following format:

**EventName**

\[ value_1 : \text{type}_1 \]
\[ \ldots \]
\[ value_N : \text{type}_N \]

Description.

### 3. Common Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LISTofFOO</td>
<td>A type name of the form LISTofFOO means a counted list of elements of type FOO. The size of the length field may vary (it is not necessarily the same size as a FOO), and in some cases, it may be implicit. It is fully specified in Appendix B. Except where explicitly noted, zero-length lists are legal.</td>
</tr>
<tr>
<td>BITMASK</td>
<td>The types BITMASK and LISTofVALUE are somewhat special. Various requests contain arguments of the form: value-mask: BITMASK value-list: LISTofVALUE These are used to allow the client to specify a subset of a heterogeneous collection of optional arguments. The value-mask specifies which arguments are to be provided; each such argument is assigned a unique bit position. The representation of the BITMASK will typically contain more bits than there are defined arguments. The unused bits in the value-mask must be zero (or the server generates a Value error). The value-list contains one value for each bit set to 1 in the mask, from least significant to most significant bit in the mask. Each value is represented with four bytes, but the actual value occupies only the least significant bytes as required. The values of the unused bytes do not matter.</td>
</tr>
<tr>
<td>LISTofVALUE</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>OR</td>
<td>A type of the form “T1 or ... or Tn” means the union of the indicated types. A single-element type is given as the element without enclosing braces.</td>
</tr>
<tr>
<td>WINDOW</td>
<td>32-bit value (top three bits guaranteed to be zero)</td>
</tr>
<tr>
<td>PIXMAP</td>
<td>32-bit value (top three bits guaranteed to be zero)</td>
</tr>
<tr>
<td>CURSOR</td>
<td>32-bit value (top three bits guaranteed to be zero)</td>
</tr>
<tr>
<td>FONT</td>
<td>32-bit value (top three bits guaranteed to be zero)</td>
</tr>
<tr>
<td>GCONTEXT</td>
<td>32-bit value (top three bits guaranteed to be zero)</td>
</tr>
<tr>
<td>COLORMAP</td>
<td>32-bit value (top three bits guaranteed to be zero)</td>
</tr>
<tr>
<td>DRAWABLE</td>
<td>WINDOW or PIXMAP</td>
</tr>
<tr>
<td>FONTABLE</td>
<td>FONT or GCONTEXT</td>
</tr>
<tr>
<td>ATOM</td>
<td>32-bit value (top three bits guaranteed to be zero)</td>
</tr>
<tr>
<td>VISUALID</td>
<td>32-bit value (top three bits guaranteed to be zero)</td>
</tr>
<tr>
<td>VALUE</td>
<td>32-bit quantity (used only in LISTofVALUE)</td>
</tr>
<tr>
<td>BYTE</td>
<td>8-bit value</td>
</tr>
<tr>
<td>INT8</td>
<td>8-bit signed integer</td>
</tr>
<tr>
<td>INT16</td>
<td>16-bit signed integer</td>
</tr>
<tr>
<td>INT32</td>
<td>32-bit signed integer</td>
</tr>
<tr>
<td>CARD8</td>
<td>8-bit unsigned integer</td>
</tr>
<tr>
<td>CARD16</td>
<td>16-bit unsigned integer</td>
</tr>
<tr>
<td>CARD32</td>
<td>32-bit unsigned integer</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>CARD32</td>
</tr>
<tr>
<td>BITGRAVITY</td>
<td>{Forget, Static, NorthWest, North, NorthEast, West, Center, East, SouthWest, South, SouthEast}</td>
</tr>
<tr>
<td>WINGRAVITY</td>
<td>{Unmap, Static, NorthWest, North, NorthEast, West, Center, East, SouthWest, South, SouthEast}</td>
</tr>
<tr>
<td>BOOL</td>
<td>{True, False}</td>
</tr>
<tr>
<td>EVENT</td>
<td>{KeyPress, KeyRelease, OwnerGrabButton, ButtonPress, ButtonRelease, EnterWindow, LeaveWindow, PointerMotion, PointerMotionHint, Button1Motion, Button2Motion, Button3Motion, Button4Motion, Button5Motion, ButtonMotion, Exposure, VisibilityChange, StructureNotify, ResizeRedirect, SubstructureNotify, SubstructureRedirect, FocusChange, PropertyChange, ColormapChange, KeymapState}</td>
</tr>
<tr>
<td>POINTEREVENT</td>
<td>{ButtonPress, ButtonRelease, EnterWindow, LeaveWindow, PointerMotion, PointerMotionHint, Button1Motion, Button2Motion, Button3Motion, Button4Motion, Button5Motion, ButtonMotion, KeymapState}</td>
</tr>
<tr>
<td>DEVICEEVENT</td>
<td>{KeyPress, KeyRelease, ButtonPress, ButtonRelease, PointerMotion, Button1Motion, Button2Motion, Button3Motion, Button4Motion, Button5Motion, ButtonMotion}</td>
</tr>
<tr>
<td>KEYSYM</td>
<td>32-bit value (top three bits guaranteed to be zero)</td>
</tr>
<tr>
<td>KEYCODE</td>
<td>CARD8</td>
</tr>
<tr>
<td>BUTTON</td>
<td>CARD8</td>
</tr>
<tr>
<td>KEYMASK</td>
<td>{Shift, Lock, Control, Mod1, Mod2, Mod3, Mod4, Mod5}</td>
</tr>
<tr>
<td>BUTMASK</td>
<td>{Button1, Button2, Button3, Button4, Button5}</td>
</tr>
<tr>
<td>KEYBUTMASK</td>
<td>KEYMASK or BUTMASK</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>STRING8</td>
<td>LISTofCARD8</td>
</tr>
<tr>
<td>STRING16</td>
<td>LISTofCHAR2B</td>
</tr>
<tr>
<td>CHAR2B</td>
<td>[byte1, byte2: CARD8]</td>
</tr>
<tr>
<td>POINT</td>
<td>[x, y: INT16]</td>
</tr>
<tr>
<td>RECTANGLE</td>
<td>[x, y: INT16, width, height: CARD16]</td>
</tr>
<tr>
<td>ARC</td>
<td>[x, y: INT16, width, height: CARD16, angle1, angle2: INT16]</td>
</tr>
<tr>
<td>HOST</td>
<td>[family: {Internet, InternetV6, ServerInterpreted, DECnet, Chaos}] address: LISTofBYTE</td>
</tr>
</tbody>
</table>

The [x,y] coordinates of a RECTANGLE specify the upper-left corner.

The primary interpretation of large characters in a STRING16 is that they are composed of two bytes used to index a two-dimensional matrix, hence, the use of CHAR2B rather than CARD16. This corresponds to the JIS/ISO method of indexing 2-byte characters. It is expected that most large fonts will be defined with 2-byte matrix indexing. For large fonts constructed with linear indexing, a CHAR2B can be interpreted as a 16-bit number by treating byte1 as the most significant byte. This means that clients should always transmit such 16-bit character values most significant byte first, as the server will never byte-swap CHAR2B quantities.

The length, format, and interpretation of a HOST address are specific to the family (see ChangeHosts request).

4. Errors

In general, when a request terminates with an error, the request has no side effects (that is, there is no partial execution). The only requests for which this is not true are ChangeWindowAttributes, ChangeGC, PolyText8, PolyText16, FreeColors, StoreColors, and ChangeKeyboardControl.

The following error codes result from various requests as follows:

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>An attempt is made to grab a key/button combination already grabbed by another client.</td>
</tr>
<tr>
<td></td>
<td>An attempt is made to free a colormap entry not allocated by the client or to free an entry in a colormap that was created with all entries writable.</td>
</tr>
<tr>
<td></td>
<td>An attempt is made to store into a read-only or an unallocated colormap entry.</td>
</tr>
<tr>
<td></td>
<td>An attempt is made to modify the access control list from other than the local host (or otherwise authorized client).</td>
</tr>
<tr>
<td></td>
<td>An attempt is made to select an event type that only one client can select at a time when another client has already selected it.</td>
</tr>
<tr>
<td>Error</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Alloc</td>
<td>The server failed to allocate the requested resource. Note that the explicit listing of Alloc errors in request only covers allocation errors at a very coarse level and is not intended to cover all cases of a server running out of allocation space in the middle of service. The semantics when a server runs out of allocation space are left unspecified, but a server may generate an Alloc error on any request for this reason, and clients should be prepared to receive such errors and handle or discard them.</td>
</tr>
<tr>
<td>Atom</td>
<td>A value for an ATOM argument does not name a defined ATOM.</td>
</tr>
<tr>
<td>Colormap</td>
<td>A value for a COLORMAP argument does not name a defined COLORMAP.</td>
</tr>
<tr>
<td>Cursor</td>
<td>A value for a CURSOR argument does not name a defined CURSOR.</td>
</tr>
<tr>
<td>Drawable</td>
<td>A value for a DRAWABLE argument does not name a defined WINDOW or PIXMAP.</td>
</tr>
<tr>
<td>Font</td>
<td>A value for a FONT argument does not name a defined FONT.</td>
</tr>
<tr>
<td></td>
<td>A value for a FONTABLE argument does not name a defined FONT or a defined GCONTEXT.</td>
</tr>
<tr>
<td>GContext</td>
<td>A value for a GCONTEXT argument does not name a defined GCONTEXT.</td>
</tr>
<tr>
<td>IDChoice</td>
<td>The value chosen for a resource identifier either is not included in the range assigned to the client or is already in use.</td>
</tr>
<tr>
<td>Implementation</td>
<td>The server does not implement some aspect of the request. A server that generates this error for a core request is deficient. As such, this error is not listed for any of the requests, but clients should be prepared to receive such errors and handle or discard them.</td>
</tr>
<tr>
<td>Length</td>
<td>The length of a request is shorter or longer than that required to minimally contain the arguments.</td>
</tr>
<tr>
<td></td>
<td>The length of a request exceeds the maximum length accepted by the server.</td>
</tr>
<tr>
<td>Match</td>
<td>An InputOnly window is used as a DRAWABLE.</td>
</tr>
<tr>
<td></td>
<td>In a graphics request, the GCONTEXT argument does not have the same root and depth as the destination DRAWABLE argument.</td>
</tr>
<tr>
<td></td>
<td>Some argument (or pair of arguments) has the correct type and range, but it fails to match in some other way required by the request.</td>
</tr>
<tr>
<td>Name</td>
<td>A font or color of the specified name does not exist.</td>
</tr>
<tr>
<td>Pixmap</td>
<td>A value for a PIXMAP argument does not name a defined PIXMAP.</td>
</tr>
<tr>
<td>Request</td>
<td>The major or minor opcode does not specify a valid request.</td>
</tr>
<tr>
<td>Error</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Value</td>
<td>Some numeric value falls outside the range of values accepted by the request. Unless a specific range is specified for an argument, the full range defined by the argument’s type is accepted. Any argument defined as a set of alternatives typically can generate this error (due to the encoding).</td>
</tr>
<tr>
<td>Window</td>
<td>A value for a WINDOW argument does not name a defined WINDOW.</td>
</tr>
</tbody>
</table>

Note

The Atom, ColorMap, Cursor, Drawable, Font, GContext, Pixmap, and Window errors are also used when the argument type is extended by union with a set of fixed alternatives, for example, <WINDOW or PointerRoot or None>.

5. Keyboards

A KEYCODE represents a physical (or logical) key. Keycodes lie in the inclusive range [8,255]. A keycode value carries no intrinsic information, although server implementors may attempt to encode geometry information (for example, matrix) to be interpreted in a server-dependent fashion. The mapping between keys and keycodes cannot be changed using the protocol.

A KEYSYM is an encoding of a symbol on the cap of a key. The set of defined KEYSYMs include the character sets Latin-1, Latin-2, Latin-3, Latin-4, Kana, Arabic, Cyrillic, Greek, Tech, Special, Publish, APL, Hebrew, Thai, and Korean as well as a set of symbols common on keyboards (Return, Help, Tab, and so on). KEYSYMs with the most significant bit (of the 29 bits) set are reserved as vendor-specific.

A list of KEYSYMs is associated with each KEYCODE. The list is intended to convey the set of symbols on the corresponding key. If the list (ignoring trailing NoSymbol entries) is a single KEYSYM “K”, then the list is treated as if it were the list “K NoSymbol K NoSymbol”. If the list (ignoring trailing NoSymbol entries) is a pair of KEYSYMs “K1 K2”, then the list is treated as if it were the list “K1 K2 K1 K2”. If the list (ignoring trailing NoSymbol entries) is a triple of KEYSYMs “K1 K2 K3”, then the list is treated as if it were the list “K1 K2 K3 NoSymbol”. When an explicit “void” element is desired in the list, the value VoidSymbol can be used.

The first four elements of the list are split into two groups of KEYSYMs. Group 1 contains the first and second KEYSYMs, Group 2 contains the third and fourth KEYSYMs. Within each group, if the second element of the group is NoSymbol, then the group should be treated as if the second element were the same as the first element, except when the first element is an alphabetic KEYSYM “K” for which both lowercase and uppercase forms are defined. In that case, the group should be treated as if the first element were the lowercase form of “K” and the second element were the uppercase form of “K”.

The standard rules for obtaining a KEYSYM from a KeyPress event make use of only the Group 1 and Group 2 KEYSYMs; no interpretation of other KEYSYMs in the list is defined. The modifier state determines which group to use. Switching between groups is controlled by the KEYSYM named MODE SWITCH, by attaching that KEYSYM to some KEYCODE and attaching that KEYCODE to any one of the modifiers Mod1 through Mod5. This modifier is called the “group modifier”. For any KEYCODE, Group 1 is used when the group modifier is off, and Group 2 is used when the group modifier is on.
The **Lock** modifier is interpreted as CapsLock when the KEYSYM named CAPS LOCK is attached to some KEYCODE and that KEYCODE is attached to the **Lock** modifier. The **Lock** modifier is interpreted as ShiftLock when the KEYSYM named SHIFT LOCK is attached to some KEYCODE and that KEYCODE is attached to the **Lock** modifier. If the **Lock** modifier could be interpreted as both CapsLock and ShiftLock, the CapsLock interpretation is used.

The operation of “keypad” keys is controlled by the KEYSYM named NUM LOCK, by attaching that KEYSYM to some KEYCODE and attaching that KEYCODE to any one of the modifiers **Mod1** through **Mod5**. This modifier is called the “numlock modifier”. The standard KEYSYMs with the prefix KEYPAD in their name are called “keypad” KEYSYMs; these are KEYSYMs with numeric value in the hexadecimal range #xFF80 to #xFFBD inclusive. In addition, vendor-specific KEYSYMs in the hexadecimal range #x11000000 to #x1100FFFF are also keypad KEYSYMs.

Within a group, the choice of KEYSYM is determined by applying the first rule that is satisfied from the following list:

- The numlock modifier is on and the second KEYSYM is a keypad KEYSYM. In this case, if the Shift modifier is on, or if the Lock modifier is on and is interpreted as ShiftLock, then the first KEYSYM is used; otherwise, the second KEYSYM is used.
- The Shift and Lock modifiers are both off. In this case, the first KEYSYM is used.
- The Shift modifier is off, and the Lock modifier is on and is interpreted as CapsLock. In this case, the first KEYSYM is used, but if that KEYSYM is lowercase alphabetic, then the corresponding uppercase KEYSYM is used instead.
- The Shift modifier is on, and the Lock modifier is on and is interpreted as CapsLock. In this case, the second KEYSYM is used, but if that KEYSYM is lowercase alphabetic, then the corresponding uppercase KEYSYM is used instead.
- The Shift modifier is on, or the Lock modifier is on and is interpreted as ShiftLock, or both. In this case, the second KEYSYM is used.

The mapping between KEYCODEs and KEYSYMs is not used directly by the server; it is merely stored for reading and writing by clients.

**6. Pointers**

Buttons are always numbered starting with one.

**7. Predefined Atoms**

Predefined atoms are not strictly necessary and may not be useful in all environments, but they will eliminate many **InternAtom** requests in most applications. Note that they are predefined only in the sense of having numeric values, not in the sense of having required semantics. The core protocol imposes no semantics on these names, but semantics are specified in other X Window System standards, such as the *Inter-Client Communication Conventions Manual* and the *X Logical Font Description Conventions*.

The following names have predefined atom values. Note that uppercase and lowercase matter.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARC</td>
<td>ITALIC_ANGLE</td>
</tr>
<tr>
<td>ATOM</td>
<td>MAX_SPACE</td>
</tr>
<tr>
<td>BITMAP</td>
<td>MIN_SPACE</td>
</tr>
<tr>
<td>CAP_HEIGHT</td>
<td>NORM_SPACE</td>
</tr>
<tr>
<td>CARDINAL</td>
<td>NOTICE</td>
</tr>
<tr>
<td>COLORMAP</td>
<td>PIXMAP</td>
</tr>
<tr>
<td>STRING</td>
<td>SUBSCRIPT_X</td>
</tr>
<tr>
<td>SUBSCRIPT_Y</td>
<td>SUPERSCRIPT_X</td>
</tr>
<tr>
<td>UNDERLINE_POSITION</td>
<td></td>
</tr>
</tbody>
</table>
To avoid conflicts with possible future names for which semantics might be imposed (either at the protocol level or in terms of higher level user interface models), names beginning with an underscore should be used for atoms that are private to a particular vendor or organization. To guarantee no conflicts between vendors and organizations, additional prefixes need to be used. However, the protocol does not define the mechanism for choosing such prefixes. For names private to a single application or end user but stored in globally accessible locations, it is suggested that two leading underscores be used to avoid conflicts with other names.

8. Connection Setup

For remote clients, the X protocol can be built on top of any reliable byte stream.

Connection Initiation

The client must send an initial byte of data to identify the byte order to be employed. The value of the byte must be octal 102 or 154. The value 102 (ASCII uppercase B) means values are transmitted most significant byte first, and value 154 (ASCII lowercase l) means values are transmitted least 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 will be transmitted with this byte order.

Following the byte-order byte, the client sends the following information at connection setup:

- protocol-major-version: CARD16
- protocol-minor-version: CARD16
- authorization-protocol-name: STRING8
- authorization-protocol-data: STRING8

The version numbers indicate what version of the protocol the client expects the server to implement.

The authorization name indicates what authorization (and authentication) protocol the client expects the server to use, and the data is specific to that protocol. Specification of valid authorization mechanisms is not part of the core X protocol. A server that does not implement the protocol the client expects or that only implements the host-based mechanism may simply ignore this information. If both name and data strings are empty, this is to be interpreted as “no explicit authorization.”
Server Response
The client receives the following information at connection setup:

success: \{Failed, Success, Authenticate\}

The client receives the following additional data if the returned success value is Failed, and the connection is not successfully established:

- protocol-major-version: CARD16
- protocol-minor-version: CARD16
- reason: STRING8

The client receives the following additional data if the returned success value is Authenticate, and further authentication negotiation is required:

- reason: STRING8

The contents of the reason string are specific to the authorization protocol in use. The semantics of this authentication negotiation are not constrained, except that the negotiation must eventually terminate with a reply from the server containing a success value of Failed or Success.

The client receives the following additional data if the returned success value is Success, and the connection is successfully established:

- protocol-major-version: CARD16
- protocol-minor-version: CARD16
- vendor: STRING8
- release-number: CARD32
- resource-id-base, resource-id-mask: CARD32
- image-byte-order: \{LSBFirst, MSBFirst\}
- bitmap-scanline-unit: \{8, 16, 32\}
- bitmap-scanline-pad: \{8, 16, 32\}
- bitmap-bit-order: \{LeastSignificant, MostSignificant\}
- pixmap-formats: LISTofFORMAT
- roots: LISTofSCREEN
- motion-buffer-size: CARD32
- maximum-request-length: CARD16
- min-keycode, max-keycode: KEYCODE

where:

- FORMAT: \[depth: CARD8,
- bits-per-pixel: \{1, 4, 8, 16, 24, 32\}
- scanline-pad: \{8, 16, 32\}]
SCREEN:  [root: WINDOW
  width-in-pixels, height-in-pixels: CARD16
  width-in-millimeters, height-in-millimeters: CARD16
  allowed-depths: LISTofDEPTH
  root-depth: CARD8
  root-visual: VISUALID
  default-colormap: COLOORMAP
  white-pixel, black-pixel: CARD32
  min-installed-maps, max-installed-maps: CARD16
  backing-stores: {Never, WhenMapped, Always}
  save-unders: BOOL
  current-input-masks: SETofEVENT]

DEPTH:  [depth: CARD8
  visuals: LISTofVISUALTYPE]

VISUALTYPE:  [visual-id: VISUALID
  class: {StaticGray, StaticColor, TrueColor, GrayScale, PseudoColor, DirectColor}
  red-mask, green-mask, blue-mask: CARD32
  bits-per-rgb-value: CARD8
  colormap-entries: CARD16]

Server Information
The information that is global to the server is:

The protocol version numbers are an escape hatch in case future revisions of the protocol are necessary. In general, the major version would increment for incompatible changes, and the minor version would increment for small upward compatible changes. Barring changes, the major version will be 11, and the minor version will be 0. The protocol version numbers returned indicate the protocol the server actually supports. This might not equal the version sent 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.

The vendor string gives some identification of the owner of the server implementation. The vendor controls the semantics of the release number.

The resource-id-mask contains a single contiguous set of bits (at least 18). The client allocates resource IDs for types WINDOW, PIXMAP, CURSOR, FONT, GCONTEXT, and COLOORMAP by choosing a value with only some subset of these bits set and ORing it with resource-id-base. Only values constructed in this way can be used to name newly created resources over this connection. Resource IDs never have the top three bits set. The client is not restricted to linear or contiguous allocation of resource IDs. Once an ID has been freed, it can be reused. An ID must be unique with respect to the IDs of all other resources, not just other resources of the same type. However, note that the value spaces of 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.

Although the server is in general responsible for byte-swapping data to match the client, images are always transmitted and received in formats (including byte order) specified by the server. The byte order for images is given by image-byte-order and applies to each scanline unit in XY format (bitmap format) and to each pixel value in Z format.
A bitmap is represented in scanline order. Each scanline is padded to a multiple of bits as given by bitmap-scanline-pad. The pad bits are of arbitrary value. The scanline is quantized in multiples of bits as given by bitmap-scanline-unit. The bitmap-scanline-unit is always less than or equal to the bitmap-scanline-pad. Within each unit, the leftmost bit in the bitmap is either the least significant or most significant bit in the unit, as given by bitmap-bit-order. If a pixmap is represented in XY format, each plane is represented as a bitmap, and the planes appear from most significant to least significant in bit order with no padding between planes.

Pixmap-formats contains one entry for each depth value. The entry describes the Z 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 that Z format for that depth. In Z format, the pixels are in scanline order, left to right within a scanline. The number of bits used to hold each pixel is given by bits-per-pixel. Bits-per-pixel may be larger than strictly required by the depth, in which case the least significant bits are used to hold the pixmap data, and the values of the unused high-order bits are undefined. When the bits-per-pixel is 4, the order of nibbles in the byte is the same as the image byte-order. When the bits-per-pixel is 1, the format is identical for bitmap format. Each scanline is padded to a multiple of bits as given by scanline-pad. When bits-per-pixel is 1, this will be identical to bitmap-scanline-pad.

How a pointing device roams the screens is up to the server implementation and is transparent to the protocol. No geometry is defined among screens.

The server may retain the recent history of pointer motion and do so to a finer granularity than is reported by MotionNotify events. The GetMotionEvents request makes such history available. The motion-buffer-size gives the approximate maximum number of elements in the history buffer.

Maximum-request-length specifies the maximum length of a request accepted by the server, in 4-byte units. That is, length is the maximum value that can appear in the length field of a request. Requests larger than this maximum generate a Length error, and the server will read and simply discard the entire request. Maximum-request-length will always be at least 4096 (that is, requests of length up to and including 16384 bytes will be accepted by all servers).

Min-keycode and max-keycode specify the smallest and largest keycode values transmitted by the server. Min-keycode is never less than 8, and max-keycode is never greater than 255. Not all keycodes in this range are required to have corresponding keys.

**Screen Information**

The information that applies per screen is:

The allowed-depths specifies what pixmap and window depths are supported. Pixmaps are supported for each depth listed, and windows of that depth are supported if at least one visual type is listed for the depth. A pixmap depth of one is always supported and listed, but windows of depth one might not be supported. A depth of zero is never listed, but zero-depth InputOnly windows are always supported.

Root-depth and root-visual specify the depth and visual type of the root window. Width-in-pixels and height-in-pixels specify the size of the root window (which cannot be changed). The class of the root window is always InputOutput. Width-in-millimeters and height-in-millimeters can be used to determine the physical size and the aspect ratio.

The default-colormap is the one initially associated with the root window. Clients with minimal color requirements creating windows of the same depth as the root may want to allocate from this map by default.

Black-pixel and white-pixel can be used in implementing a monochrome application. These pixel values are for permanently allocated entries in the default-colormap. The actual RGB values may
be settable on some screens and, in any case, may not actually be black and white. The names are intended to convey the expected relative intensity of the colors.

The border of the root window is initially a pixmap filled with the black-pixel. The initial background of the root window is a pixmap filled with some unspecified two-color pattern using black-pixel and white-pixel.

Min-installed-maps specifies the number of maps that can be guaranteed to be installed simultaneously (with InstallColormap), regardless of the number of entries allocated in each map. Max-installed-maps specifies the maximum number of maps that might possibly be installed simultaneously, depending on their allocations. Multiple static-visual colormaps with identical contents but differing in resource ID should be considered as a single map for the purposes of this number. For the typical case of a single hardware colormap, both values will be 1.

Backing-stores indicates when the server supports backing stores for this screen, although it may be storage limited in the number of windows it can support at once. If save-unders is True, the server can support the save-under mode in CreateWindow and ChangeWindowAttributes, although again it may be storage limited.

The current-input-events is what GetWindowAttributes would return for the all-event-masks for the root window.

Visual Information

The information that applies per visual-type is:

A given visual type might be listed for more than one depth or for more than one screen.

For PseudoColor, a pixel value indexes a colormap to produce independent RGB values; the RGB values can be changed dynamically. GrayScale is treated in the same way as PseudoColor except which primary drives the screen is undefined; thus, the client should always store the same value for red, green, and blue in colormaps. For DirectColor, a pixel value is decomposed into separate RGB subfields, and each subfield separately indexes the colormap for the corresponding value. The RGB values can be changed dynamically. TrueColor is treated in the same way as DirectColor except the colormap has predefined read-only RGB values. These values are server-dependent but provide linear or near-linear increasing ramps in each primary. StaticColor is treated in the same way as PseudoColor except the colormap has predefined read-only RGB values, which are server-dependent. StaticGray is treated in the same way as StaticColor except the red, green, and blue values are equal for any single pixel value, resulting in shades of gray. StaticGray with a two-entry colormap can be thought of as monochrome.

The red-mask, green-mask, and blue-mask are only defined for DirectColor and TrueColor. Each has one contiguous set of bits set to 1 with no intersections. Usually each mask has the same number of bits set to 1.

The bits-per-rgb-value specifies the log base 2 of the number of distinct color intensity values (individually) of red, green, and blue. This number need not bear any relation to the number of colormap entries. Actual RGB values are always passed in the protocol within a 16-bit spectrum, with 0 being minimum intensity and 65535 being the maximum intensity. On hardware that provides a linear zero-based intensity ramp, the following relationship exists:

\[ \text{hw-intensity} = \frac{\text{protocol-intensity}}{(65536 / \text{total-hw-intensities})} \]

Colormap entries are indexed from 0. The colormap-entries defines the number of available colormap entries in a newly created colormap. For DirectColor and TrueColor, this will usually be 2 to the power of the maximum number of bits set to 1 in red-mask, green-mask, and blue-mask.
CreateWindow

$\text{wid, parent: WINDOW}$

$\text{class: [InputOutput, InputOnly, CopyFromParent]}$

$\text{depth: CARD8}$

$\text{visual: VISUALID or CopyFromParent}$

$x, y: INT16$

$\text{width, height, border-width: CARD16}$

$\text{value-mask: BITMASK}$

$\text{value-list: LISTofVALUE}$

Errors: Alloc, Colormap, Cursor, IDChoice, Match, Pixmap, Value, Window

This request creates an unmapped window and assigns the identifier $\text{wid}$ to it.

A class of $\text{CopyFromParent}$ means the class is taken from the parent. A depth of zero for class $\text{InputOutput}$ or $\text{CopyFromParent}$ means the depth is taken from the parent. A visual of $\text{CopyFromParent}$ means the visual type is taken from the parent. For class $\text{InputOutput}$, the visual type and depth must be a combination supported for the screen (or a $\text{Match}$ error results). The depth need not be the same as the parent, but the parent must not be of class $\text{InputOnly}$ (or a $\text{Match}$ error results). For class $\text{InputOnly}$, the depth must be zero (or a $\text{Match}$ error results), and the visual must be one supported for the screen (or a $\text{Match}$ error results). However, the parent can have any depth and class.

The server essentially acts as if $\text{InputOnly}$ windows do not exist for the purposes of graphics requests, exposure processing, and $\text{VisibilityNotify}$ events. An $\text{InputOnly}$ window cannot be used as a drawable (as a source or destination for graphics requests). $\text{InputOnly}$ and $\text{InputOutput}$ windows act identically in other respects—properties, grabs, input control, and so on.

The coordinate system has the X axis horizontal and the Y axis vertical with the origin $[0, 0]$ at the upper-left corner. Coordinates are integral, in terms of pixels, and coincide with pixel centers. Each window and pixmap has its own coordinate system. For a window, the origin is inside the border at the inside, upper-left corner.

The $x$ and $y$ coordinates for the window are relative to the parent’s origin and specify the position of the upper-left outer corner of the window (not the origin). The width and height specify the inside size (not including the border) and must be nonzero (or a $\text{Value}$ error results). The border-width for an $\text{InputOnly}$ window must be zero (or a $\text{Match}$ error results).

The window is placed on top in the stacking order with respect to siblings.

The value-mask and value-list specify attributes of the window that are to be explicitly initialized. The possible values are:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>background-pixmap</td>
<td>PIXMAP or None or ParentRelative</td>
</tr>
<tr>
<td>background-pixel</td>
<td>CARD32</td>
</tr>
<tr>
<td>border-pixmap</td>
<td>PIXMAP or CopyFromParent</td>
</tr>
<tr>
<td>border-pixel</td>
<td>CARD32</td>
</tr>
<tr>
<td>bit-gravity</td>
<td>BITGRAVITY</td>
</tr>
</tbody>
</table>
Attribute | Type
---|---
win-gravity | WINGRAVITY
backing-store | \{NotUseful, WhenMapped, Always\}
backing-planes | CARD32
backing-pixel | CARD32
save-under | BOOL
event-mask | SETofEVENT
do-not-propagate-mask | SETofDEVICEEVENT
override-redirect | BOOL
colormap | COLORMAP or CopyFromParent
cursor | CURSOR or None

The default values when attributes are not explicitly initialized are:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>background-pixmap</td>
<td>None</td>
</tr>
<tr>
<td>border-pixmap</td>
<td>CopyFromParent</td>
</tr>
<tr>
<td>bit-gravity</td>
<td>Forget</td>
</tr>
<tr>
<td>win-gravity</td>
<td>NorthWest</td>
</tr>
<tr>
<td>backing-store</td>
<td>NotUseful</td>
</tr>
<tr>
<td>backing-planes</td>
<td>all ones</td>
</tr>
<tr>
<td>backing-pixel</td>
<td>zero</td>
</tr>
<tr>
<td>save-under</td>
<td>False</td>
</tr>
<tr>
<td>event-mask</td>
<td>{} (empty set)</td>
</tr>
<tr>
<td>do-not-propagate-mask</td>
<td>{} (empty set)</td>
</tr>
<tr>
<td>override-redirect</td>
<td>False</td>
</tr>
<tr>
<td>colormap</td>
<td>CopyFromParent</td>
</tr>
<tr>
<td>cursor</td>
<td>None</td>
</tr>
</tbody>
</table>

Only the following attributes are defined for **InputOnly** windows:
- win-gravity
- event-mask
- do-not-propagate-mask
- override-redirect
- cursor

It is a **Match** error to specify any other attributes for **InputOnly** windows.

If background-pixmap is given, it overrides the default background-pixmap. The background pixmap and the window must have the same root and the same depth (or a **Match** error results). Any size pixmap can be used, although some sizes may be faster than others. If background None is specified, the window has no defined background. If background ParentRelative is specified, the parent’s background is used, but the window must have the same depth as the parent (or a **Match** error results). If the parent has background None, then the window will also have background None. A copy of the parent’s background is not made. The parent’s background is reexamined each time the window background is required. If background-pixel is given, it overrides the default background-pixmap and any background-pixmap given explicitly, and a pixmap
of undefined size filled with background-pixel is used for the background. Range checking is not performed on the background-pixel value; it is simply truncated to the appropriate number of bits. For a ParentRelative background, the background tile origin always aligns with the parent’s background tile origin. Otherwise, the background tile origin is always the window origin.

When no valid contents are available for regions of a window and the regions are either visible or the server is maintaining backing store, the server automatically tiles the regions with the window’s background unless the window has a background of None. If the background is None, the previous screen contents from other windows of the same depth as the window are simply left in place if the contents come from the parent of the window or an inferior of the parent; otherwise, the initial contents of the exposed regions are undefined. Exposure events are then generated for the regions, even if the background is None.

The border tile origin is always the same as the background tile origin. If border-pixmap is given, it overrides the default border-pixmap. The border pixmap and the window must have the same root and the same depth (or a Match error results). Any size pixmap can be used, although some sizes may be faster than others. If CopyFromParent is given, the parent’s border pixmap is copied (subsequent changes to the parent’s border attribute do not affect the child), but the window must have the same depth as the parent (or a Match error results). The pixmap might be copied by sharing the same pixmap object between the child and parent or by making a complete copy of the pixmap contents. If border-pixel is given, it overrides the default border-pixmap and any border-pixmap given explicitly, and a pixmap of undefined size filled with border-pixel is used for the border. Range checking is not performed on the border-pixel value; it is simply truncated to the appropriate number of bits.

Output to a window is always clipped to the inside of the window, so that the border is never affected.

The bit-gravity defines which region of the window should be retained if the window is resized, and win-gravity defines how the window should be repositioned if the parent is resized (see ConfigureWindow request).

A backing-store of WhenMapped advises the server that maintaining contents of obscured regions when the window is mapped would be beneficial. A backing-store of Always advises the server that maintaining contents even when the window is unmapped would be beneficial. In this case, the server may generate an exposure event when the window is created. A value of NotUseful advises the server that maintaining contents is unnecessary, although a server may still choose to maintain contents while the window is mapped. Note that if the server maintains contents, then the server should maintain complete contents not just the region within the parent boundaries, even if the window is larger than its parent. While the server maintains contents, exposure events will not normally be generated, but the server may stop maintaining contents at any time.

If save-under is True, the server is advised that when this window is mapped, saving the contents of windows it obscures would be beneficial.

When the contents of obscured regions of a window are being maintained, regions obscured by noninferior windows are included in the destination (and source, when the window is the source) of graphics requests, but regions obscured by inferior windows are not included.

The backing-planes indicates (with bits set to 1) which bit planes of the window hold dynamic data that must be preserved in backing-stores and during save-unders. The backing-pixel specifies what value to use in planes not covered by backing-planes. The server is free to save only the specified bit planes in the backing-store or save-under and regenerate the remaining planes with the specified pixel value. Any bits beyond the specified depth of the window in these values are simply ignored.
The event-mask defines which events the client is interested in for this window (or for some event
types, inferiors of the window). The do-not-propagate-mask defines which events should not be
propagated to ancestor windows when no client has the event type selected in this window.

The override-redirect specifies whether map and configure requests on this window should over-
ride a SubstructureRedirect on the parent, typically to inform a window manager not to tamper
with the window.

The colormap specifies the colormap that best reflects the true colors of the window. Servers
capable of supporting multiple hardware colormaps may use this information, and window man-
gagers may use it for InstallColormap requests. The colormap must have the same visual type
and root as the window (or a Match error results). If CopyFromParent is specified, the parent’s
colormap is copied (subsequent changes to the parent’s colormap attribute do not affect the child).
However, the window must have the same visual type as the parent (or a Match error results),
and the parent must not have a colormap of None (or a Match error results). For an explanation of
None, see FreeColormap request. The colormap is copied by sharing the colormap object
between the child and the parent, not by making a complete copy of the colormap contents.

If a cursor is specified, it will be used whenever the pointer is in the window. If None is speci-
fied, the parent’s cursor will be used when the pointer is in the window, and any change in the
parent’s cursor will cause an immediate change in the displayed cursor.

This request generates a CreateNotify event.

The background and border pixmaps and the cursor may be freed immediately if no further
explicit references to them are to be made.

Subsequent drawing into the background or border pixmap has an undefined effect on the window
state. The server might or might not make a copy of the pixmap.

---

**ChangeWindowAttributes**

```plaintext
window: WINDOW
value-mask: BITMASK
value-list: LISTofVALUE
```

Errors: Access, Colormap, Cursor, Match, Pixmap, Value, Window

The value-mask and value-list specify which attributes are to be changed. The values and restric-
tions are the same as for CreateWindow.

Setting a new background, whether by background-pixmap or background-pixel, overrides any
previous background. Setting a new border, whether by border-pixel or border-pixmap, overrides
any previous border.

Changing the background does not cause the window contents to be changed. Setting the border
or changing the background such that the border tile origin changes causes the border to be
repainted. Changing the background of a root window to None or ParentRelative restores the
default background pixmap. Changing the border of a root window to CopyFromParent
restores the default border pixmap.

Changing the win-gravity does not affect the current position of the window.

Changing the backing-store of an obscured window to WhenMapped or Always or changing
the backing-planes, backing-pixel, or save-under of a mapped window may have no immediate
effect.
Multiple clients can select input on the same window; their event-masks are disjoint. When an event is generated, it will be reported to all interested clients. However, only one client at a time can select for **SubstructureRedirect**, only one client at a time can select for **ResizeRedirect**, and only one client at a time can select for **ButtonPress**. An attempt to violate these restrictions results in an **Access** error.

There is only one do-not-propagate-mask for a window, not one per client.

Changing the colormap of a window (by defining a new map, not by changing the contents of the existing map) generates a **ColormapNotify** event. Changing the colormap of a visible window might have no immediate effect on the screen (see **InstallColormap** request).

Changing the cursor of a root window to **None** restores the default cursor.

The order in which attributes are verified and altered is server-dependent. If an error is generated, a subset of the attributes may have been altered.

---

**GetWindowAttributes**

```
window: WINDOW

visual: VISUALID
class: { InputOutput, InputOnly }
bit-gravity: BITGRAVITY
win-gravity: WINGRAVITY
backing-store: { NotUseful, WhenMapped, Always }
backing-planes: CARD32
backing-pixel: CARD32
save-under: BOOL
colormap: COLOMAP or None
map-is-installed: BOOL
map-state: { Unmapped, Unviewable, Viewable }
all-event-masks, your-event-mask: SETofEVENT
do-not-propagate-mask: SETofDEVICEEVENT
override-redirect: BOOL

Errors: Window
```

This request returns the current attributes of the window. A window is **Unviewable** if it is mapped but some ancestor is unmapped. All-event-masks is the inclusive-OR of all event masks selected on the window by clients. Your-event-mask is the event mask selected by the querying client.

---

**DestroyWindow**

```
window: WINDOW

Errors: Window
```

If the argument window is mapped, an **UnmapWindow** request is performed automatically. The window and all inferiors are then destroyed, and a **DestroyNotify** event is generated for each window. The ordering of the **DestroyNotify** events is such that for any given window,
**DestroyNotify** is generated on all inferiors of the window before being generated on the window itself. The ordering among siblings and across subhierarchies is not otherwise constrained. Normal exposure processing on formerly obscured windows is performed. If the window is a root window, this request has no effect.

**DestroySubwindows**

- **window**: WINDOW
- **Errors**: Window

This request performs a **DestroyWindow** request on all children of the window, in bottom-to-top stacking order.

**ChangeSaveSet**

- **window**: WINDOW
- **mode**: {Insert, Delete}
- **Errors**:
  - Match, Value, Window

This request adds or removes the specified window from the client’s save-set. The window must have been created by some other client (or a **Match** error results). For further information about the use of the save-set, see section 10.

When windows are destroyed, the server automatically removes them from the save-set.

**ReparentWindow**

- **window**, **parent**: WINDOW
- **x**, **y**: INT16
- **Errors**: Match, Window

If the window is mapped, an **UnmapWindow** request is performed automatically first. The window is then removed from its current position in the hierarchy and is inserted as a child of the specified parent. The x and y coordinates are relative to the parent’s origin and specify the new position of the upper-left outer corner of the window. The window is placed on top in the stacking order with respect to siblings. A **ReparentNotify** event is then generated. The override-redirect attribute of the window is passed on in this event; a value of **True** indicates that a window manager should not tamper with this window. Finally, if the window was originally mapped, a **MapWindow** request is performed automatically.

Normal exposure processing on formerly obscured windows is performed. The server might not generate exposure events for regions from the initial unmap that are immediately obscured by the final map.

A **Match** error is generated if:
  - The new parent is not on the same screen as the old parent.
• The new parent is the window itself or an inferior of the window.
• The new parent is **InputOnly**, and the window is not.
• The window has a **ParentRelative** background, and the new parent is not the same depth as the window.

**MapWindow**

*window*: WINDOW

**Errors:** **Window**

If the window is already mapped, this request has no effect.

If the override-redirect attribute of the window is **False** and some other client has selected **SubstructureRedirect** on the parent, then a **MapRequest** event is generated, but the window remains unmapped. Otherwise, the window is mapped, and a **MapNotify** event is generated.

If the window is now viewable and its contents have been discarded, the window is tiled with its background (if no background is defined, the existing screen contents are not altered), and zero or more exposure events are generated. If a backing-store has been maintained while the window was unmapped, no exposure events are generated. If a backing-store will now be maintained, a full-window exposure is always generated. Otherwise, only visible regions may be reported. Similar tiling and exposure take place for any newly viewable inferiors.

**MapSubwindows**

*window*: WINDOW

**Errors:** **Window**

This request performs a **MapWindow** request on all unmapped children of the window, in top-to-bottom stacking order.

**UnmapWindow**

*window*: WINDOW

**Errors:** **Window**

If the window is already unmapped, this request has no effect. Otherwise, the window is unmapped, and an **UnmapNotify** event is generated. Normal exposure processing on formerly obscured windows is performed.

**UnmapSubwindows**

*window*: WINDOW

**Errors:** **Window**
This request performs an **UnmapWindow** request on all mapped children of the window, in bottom-to-top stacking order.

### ConfigureWindow

- **window**: WINDOW
- **value-mask**: BITMASK
- **value-list**: LISTofVALUE

**Errors**: Match, Value, Window

This request changes the configuration of the window. The value-mask and value-list specify which values are to be given. The possible values are:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>INT16</td>
</tr>
<tr>
<td>y</td>
<td>INT16</td>
</tr>
<tr>
<td>width</td>
<td>CARD16</td>
</tr>
<tr>
<td>height</td>
<td>CARD16</td>
</tr>
<tr>
<td>border-width</td>
<td>CARD16</td>
</tr>
<tr>
<td>sibling</td>
<td>WINDOW</td>
</tr>
<tr>
<td>stack-mode</td>
<td>{ Above, Below, TopIf, BottomIf, Opposite }</td>
</tr>
</tbody>
</table>

The x and y coordinates are relative to the parent’s origin and specify the position of the upper-left outer corner of the window. The width and height specify the inside size, not including the border, and must be nonzero (or a **Value** error results). Those values not specified are taken from the existing geometry of the window. Note that changing just the border-width leaves the outer-left corner of the window in a fixed position but moves the absolute position of the window’s origin. It is a **Match** error to attempt to make the border-width of an **InputOnly** window nonzero.

If the override-redirect attribute of the window is **False** and some other client has selected **SubstructureRedirect** on the parent, a **ConfigureRequest** event is generated, and no further processing is performed. Otherwise, the following is performed:

If some other client has selected **ResizeRedirect** on the window and the inside width or height of the window is being changed, a **ResizeRequest** event is generated, and the current inside width and height are used instead. Note that the override-redirect attribute of the window has no effect on **ResizeRedirect** and that **SubstructureRedirect** on the parent has precedence over **ResizeRedirect** on the window.

The geometry of the window is changed as specified, the window is restacked among siblings, and a **ConfigureNotify** event is generated if the state of the window actually changes. If the inside width or height of the window has actually changed, then children of the window are affected, according to their win-gravity. Exposure processing is performed on formerly obscured windows (including the window itself and its inferiors if regions of them were obscured but now are not). Exposure processing is also performed on any new regions of the window (as a result of increasing the width or height) and on any regions where window contents are lost.

If the inside width or height of a window is not changed but the window is moved or its border is changed, then the contents of the window are not lost but move with the window. Changing the inside width or height of the window causes its contents to be moved or lost, depending on the
bit-gravity of the window. It also causes children to be reconfigured, depending on their win-
gravity. For a change of width and height of $W$ and $H$, we define the $[x, y]$ pairs as:

<table>
<thead>
<tr>
<th>Direction</th>
<th>Deltas</th>
</tr>
</thead>
<tbody>
<tr>
<td>NorthWest</td>
<td>[0, 0]</td>
</tr>
<tr>
<td>North</td>
<td>[W/2, 0]</td>
</tr>
<tr>
<td>NorthEast</td>
<td>[W, 0]</td>
</tr>
<tr>
<td>West</td>
<td>[0, H/2]</td>
</tr>
<tr>
<td>Center</td>
<td>[W/2, H/2]</td>
</tr>
<tr>
<td>East</td>
<td>[W, H/2]</td>
</tr>
<tr>
<td>SouthWest</td>
<td>[0, H]</td>
</tr>
<tr>
<td>South</td>
<td>[W/2, H]</td>
</tr>
<tr>
<td>SouthEast</td>
<td>[W, H]</td>
</tr>
</tbody>
</table>

When a window with one of these bit-gravities is resized, the corresponding pair defines the change in position of each pixel in the window. When a window with one of these win-gravities has its parent window resized, the corresponding pair defines the change in position of the window within the parent. This repositioning generates a GravityNotify event. GravityNotify events are generated after the ConfigureNotify event is generated.

A gravity of Static indicates that the contents or origin should not move relative to the origin of the root window. If the change in size of the window is coupled with a change in position of $[X, Y]$, then for bit-gravity the change in position of each pixel is $[-X, -Y]$ and for win-gravity the change in position of a child when its parent is so resized is $[-X, -Y]$. Note that Static gravity still only takes effect when the width or height of the window is changed, not when the window is simply moved.

A bit-gravity of Forget indicates that the window contents are always discarded after a size change, even if backing-store or save-under has been requested. The window is tiled with its background (except, if no background is defined, the existing screen contents are not altered) and zero or more exposure events are generated.

The contents and borders of inferiors are not affected by their parent's bit-gravity. A server is permitted to ignore the specified bit-gravity and use Forget instead.

A win-gravity of Unmap is like NorthWest, but the child is also unmapped when the parent is resized, and an UnmapNotify event is generated. UnmapNotify events are generated after the ConfigureNotify event is generated.

If a sibling and a stack-mode are specified, the window is restacked as follows:

Above       The window is placed just above the sibling.
Below       The window is placed just below the sibling.
TopIf       If the sibling occludes the window, then the window is placed at the top of the stack.
BottomIf    If the window occludes the sibling, then the window is placed at the bottom of the stack.
Opposite    If the sibling occludes the window, then the window is placed at the top of the stack. Otherwise, if the window occludes the sibling, then the window is placed at the bottom of the stack.
If a stack-mode is specified but no sibling is specified, the window is restacked as follows:

- **Above**  
  The window is placed at the top of the stack.

- **Below**  
  The window is placed at the bottom of the stack.

- **TopIf**  
  If any sibling occludes the window, then the window is placed at the top of the stack.

- **BottomIf**  
  If the window occludes any sibling, then the window is placed at the bottom of the stack.

- **Opposite**  
  If any sibling occludes the window, then the window is placed at the top of the stack. Otherwise, if the window occludes any sibling, then the window is placed at the bottom of the stack.

It is a **Match** error if a sibling is specified without a stack-mode or if the window is not actually a sibling.

Note that the computations for **BottomIf**, **TopIf**, and **Opposite** are performed with respect to the window’s final geometry (as controlled by the other arguments to the request), not to its initial geometry.

Attempts to configure a root window have no effect.

---

### CirculateWindow

```plaintext
window: WINDOW
direction: { RaiseLowest, LowerHighest }
```

**Errors:** Value, Window

If some other client has selected **SubstructureRedirect** on the window, then a **CirculateRequest** event is generated, and no further processing is performed. Otherwise, the following is performed, and then a **CirculateNotify** event is generated if the window is actually restacked.

For **RaiseLowest**, **CirculateWindow** raises the lowest mapped child (if any) that is occluded by another child to the top of the stack. For **LowerHighest**, **CirculateWindow** lowers the highest mapped child (if any) that occludes another child to the bottom of the stack. Exposure processing is performed on formerly obscured windows.

---

### GetGeometry

```plaintext
drawable: DRAWABLE
→
root: WINDOW
depth: CARD8
x, y: INT16
width, height, border-width: CARD16
```

**Errors:** Drawable

This request returns the root and current geometry of the drawable. The depth is the number of bits per pixel for the object. The x, y, and border-width will always be zero for pixmaps. For a
window, the \( x \) and \( y \) coordinates specify the upper-left outer corner of the window relative to its parent’s origin, and the width and height specify the inside size, not including the border. It is legal to pass an **InputOnly** window as a drawable to this request.

```plaintext
QueryTree
  window: WINDOW
  →
    root: WINDOW
    parent: WINDOW or None
    children: LIST of WINDOW

Errors: Window
```

This request returns the root, the parent, and the children of the window. The children are listed in bottom-to-top stacking order.

```plaintext
InternAtom
  name: STRING8
  only-if-exists: BOOL
  →
    atom: ATOM or None

Errors: Alloc, Value
```

This request returns the atom for the given name. If only-if-exists is **False**, then the atom is created if it does not exist. The string should use the ISO Latin-1 encoding. Uppercase and lowercase matter.

The lifetime of an atom is not tied to the interning client. Atoms remain defined until server reset (see section 10).

```plaintext
GetAtomName
  atom: ATOM
  →
    name: STRING8

Errors: Atom
```

This request returns the name for the given atom.
ChangeProperty

window: WINDOW
property, type: ATOM
format: {8, 16, 32}
mode: {Replace, Prepend, Append}
data: LISTofINT8 or LISTofINT16 or LISTofINT32

Errors: Alloc, Atom, Match, Value, Window

This request alters the property for the specified window. The type is uninterpreted by the server. The format specifies whether the data should be viewed as a list of 8-bit, 16-bit, or 32-bit quantities so that the server can correctly byte-swap as necessary.

If the mode is Replace, the previous property value is discarded. If the mode is Prepend or Append, then the type and format must match the existing property value (or a Match error results). If the property is undefined, it is treated as defined with the correct type and format with zero-length data. For Prepend, the data is tacked on to the beginning of the existing data, and for Append, it is tacked on to the end of the existing data.

This request generates a PropertyNotify event on the window.

The lifetime of a property is not tied to the storing client. Properties remain until explicitly deleted, until the window is destroyed, or until server reset (see section 10).

The maximum size of a property is server-dependent and may vary dynamically.

DeleteProperty

window: WINDOW
property: ATOM

Errors: Atom, Window

This request deletes the property from the specified window if the property exists and generates a PropertyNotify event on the window unless the property does not exist.

GetProperty

window: WINDOW
property: ATOM
type: ATOM or AnyPropertyType
long-offset, long-length: CARD32
delete: BOOL
→
type: ATOM or None
format: {0, 8, 16, 32}
bytes-after: CARD32
value: LISTofINT8 or LISTofINT16 or LISTofINT32

Errors: Atom, Value, Window
If the specified property does not exist for the specified window, then the return type is None, the format and bytes-after are zero, and the value is empty. The delete argument is ignored in this case. If the specified property exists but its type does not match the specified type, then the return type is the actual type of the property, the format is the actual format of the property (never zero), the bytes-after is the length of the property in bytes (even if the format is 16 or 32), and the value is empty. The delete argument is ignored in this case. If the specified property exists and either AnyPropertyType is specified or the specified type matches the actual type of the property, then the return type is the actual type of the property, the format is the actual format of the property (never zero), and the bytes-after and value are as follows, given:

\[
\begin{align*}
N &= \text{actual length of the stored property in bytes} \\
&\quad \text{(even if the format is 16 or 32)} \\
I &= 4 \times \text{long-offset} \\
T &= N - I \\
L &= \text{MINIMUM}(T, 4 \times \text{long-length}) \\
A &= N - (I + L)
\end{align*}
\]

The returned value starts at byte index I in the property (indexing from 0), and its length in bytes is L. However, it is a Value error if long-offset is given such that L is negative. The value of bytes-after is A, giving the number of trailing unread bytes in the stored property. If delete is True and the bytes-after is zero, the property is also deleted from the window, and a PropertyNotify event is generated on the window.

**RotateProperties**

- window: WINDOW
- delta: INT16
- properties: LISTofATOM

Errors: Atom, Match, Window

If the property names in the list are viewed as being numbered starting from zero, and there are N property names in the list, then the value associated with property name I becomes the value associated with property name \((I + \text{delta}) \mod N\), for all I from zero to \(N - 1\). The effect is to rotate the states by delta places around the virtual ring of property names (right for positive delta, left for negative delta).

If delta mod N is nonzero, a PropertyNotify event is generated for each property in the order listed.

If an atom occurs more than once in the list or no property with that name is defined for the window, a Match error is generated. If an Atom or Match error is generated, no properties are changed.
ListProperties

\[ \text{window: WINDOW} \]
\[ \rightarrow \]
\[ \text{atoms: LISTofATOM} \]

Errors: Window

This request returns the atoms of properties currently defined on the window.

SetSelectionOwner

\[ \text{selection: ATOM} \]
\[ \text{owner: WINDOW or None} \]
\[ \text{time: TIMESTAMP or CurrentTime} \]

Errors: Atom, Window

This request changes the owner, owner window, and last-change time of the specified selection. This request has no effect if the specified time is earlier than the current last-change time of the specified selection or is later than the current server time. Otherwise, the last-change time is set to the specified time with CurrentTime replaced by the current server time. If the owner window is specified as None, then the owner of the selection becomes None (that is, no owner). Otherwise, the owner of the selection becomes the client executing the request. If the new owner (whether a client or None) is not the same as the current owner and the current owner is not None, then the current owner is sent a SelectionClear event.

If the client that is the owner of a selection is later terminated (that is, its connection is closed) or if the owner window it has specified in the request is later destroyed, then the owner of the selection automatically reverts to None, but the last-change time is not affected.

The selection atom is uninterpreted by the server. The owner window is returned by the GetSelectionOwner request and is reported in SelectionRequest and SelectionClear events.

Selections are global to the server.

GetSelectionOwner

\[ \text{selection: ATOM} \]
\[ \rightarrow \]
\[ \text{owner: WINDOW or None} \]

Errors: Atom

This request returns the current owner window of the specified selection, if any. If None is returned, then there is no owner for the selection.
ConvertSelection

selection, target: ATOM
property: ATOM or None
requestor: WINDOW
time: TIMESTAMP or CurrentTime

Errors: Atom, Window

If the specified selection has an owner, the server sends a SelectionRequest event to that owner. If no owner for the specified selection exists, the server generates a SelectionNotify event to the requestor with property None. The arguments are passed on unchanged in either of the events.

SendEvent

destination: WINDOW or PointerWindow or InputFocus
propagate: BOOL
event-mask: SETofEVENT
event: <normal-event-format>

Errors: Value, Window

If PointerWindow is specified, destination is replaced with the window that the pointer is in. If InputFocus is specified and the focus window contains the pointer, destination is replaced with the window that the pointer is in. Otherwise, destination is replaced with the focus window.

If the event-mask is the empty set, then the event is sent to the client that created the destination window. If that client no longer exists, no event is sent.

If propagate is False, then the event is sent to every client selecting on destination any of the event types in event-mask.

If propagate is True and no clients have selected on destination any of the event types in event-mask, then destination is replaced with the closest ancestor of destination for which some client has selected a type in event-mask and no intervening window has that type in its do-not-propagate-mask. If no such window exists or if the window is an ancestor of the focus window and InputFocus was originally specified as the destination, then the event is not sent to any clients. Otherwise, the event is reported to every client selecting on the final destination any of the types specified in event-mask.

The event code must be one of the core events or one of the events defined by an extension (or a Value error results) so that the server can correctly byte-swap the contents as necessary. The contents of the event are otherwise unaltered and unchecked by the server except to force on the most significant bit of the event code and to set the sequence number in the event correctly.

Active grabs are ignored for this request.
GrabPointer

    grab-window: WINDOW
    owner-events: BOOL
    event-mask: SETofPOINTEREVENT
    pointer-mode, keyboard-mode: {Synchronous, Asynchronous}
    confine-to: WINDOW or None
    cursor: CURSOR or None
    time: TIMESTAMP or CurrentTime

→

status: {Success, AlreadyGrabbed, Frozen, InvalidTime, NotViewable}

Errors: Cursor, Value, Window

This request actively grabs control of the pointer. Further pointer events are only reported to the grabbing client. The request overrides any active pointer grab by this client.

If owner-events is False, all generated pointer events are reported with respect to grab-window and are only reported if selected by event-mask. If owner-events is True and a generated pointer event would normally be reported to this client, it is reported normally. Otherwise, the event is reported with respect to the grab-window and is only reported if selected by event-mask. For either value of owner-events, unreported events are simply discarded.

If pointer-mode is Asynchronous, pointer event processing continues normally. If the pointer is currently frozen by this client, then processing of pointer events is resumed. If pointer-mode is Synchronous, the state of the pointer (as seen by means of the protocol) appears to freeze, and no further pointer events are generated by the server until the grabbing client issues a releasing AllowEvents request or until the pointer grab is released. Actual pointer changes are not lost while the pointer is frozen. They are simply queued for later processing.

If keyboard-mode is Asynchronous, keyboard event processing is unaffected by activation of the grab. If keyboard-mode is Synchronous, the state of the keyboard (as seen by means of the protocol) appears to freeze, and no further keyboard events are generated by the server until the grabbing client issues a releasing AllowEvents request or until the pointer grab is released. Actual keyboard changes are not lost while the keyboard is frozen. They are simply queued for later processing.

If a cursor is specified, then it is displayed regardless of what window the pointer is in. If no cursor is specified, then when the pointer is in grab-window or one of its subwindows, the normal cursor for that window is displayed. Otherwise, the cursor for grab-window is displayed.

If a confine-to window is specified, then the pointer will be restricted to stay contained in that window. The confine-to window need have no relationship to the grab-window. If the pointer is not initially in the confine-to window, then it is warped automatically to the closest edge (and enter/leave events are generated normally) just before the grab activates. If the confine-to window is subsequently reconfigured, the pointer will be warped automatically as necessary to keep it contained in the window.

This request generates EnterNotify and LeaveNotify events.

The request fails with status AlreadyGrabbed if the pointer is actively grabbed by some other client. The request fails with status Frozen if the pointer is frozen by an active grab of another client. The request fails with status NotViewable if grab-window or confine-to window is not viewable or if the confine-to window lies completely outside the boundaries of the root window. The request fails with status InvalidTime if the specified time is earlier than the last-pointer-grab time or later than the current server time. Otherwise, the last-pointer-grab time is set to the
specified time, with CurrentTime replaced by the current server time.

**UngrabPointer**

\[ time : \text{T I M E S T A M P} \text{ or } \text{CurrentTime} \]

This request releases the pointer if this client has it actively grabbed (from either **GrabPointer** or **GrabButton** or from a normal button press) and releases any queued events. The request has no effect if the specified time is earlier than the last-pointer-grab time or is later than the current server time.

This request generates **EnterNotify** and **LeaveNotify** events.

An **UngrabPointer** request is performed automatically if the event window or confine-to window for an active pointer grab becomes not viewable or if window reconfiguration causes the confine-to window to lie completely outside the boundaries of the root window.

**GrabButton**

\[ \text{modifiers} : \text{SETofKEYMASK} \text{ or } \text{AnyModifier} \]
\[ \text{button} : \text{BUTTON} \text{ or } \text{AnyButton} \]
\[ \text{grab-window} : \text{WINDOW} \]
\[ \text{owner-events} : \text{BOOL} \]
\[ \text{event-mask} : \text{SETofPOINTEREVENT} \]
\[ \text{pointer-mode}, \text{keyboard-mode} : \{ \text{Synchronous}, \text{Asynchronous} \} \]
\[ \text{confine-to} : \text{WINDOW} \text{ or } \text{None} \]
\[ \text{cursor} : \text{CURSOR} \text{ or } \text{None} \]

**Errors:** Access, Cursor, Value, Window

This request establishes a passive grab. In the future, the pointer is actively grabbed as described in **GrabPointer**, the last-pointer-grab time is set to the time at which the button was pressed (as transmitted in the **ButtonPress** event), and the **ButtonPress** event is reported if all of the following conditions are true:

1. The pointer is not grabbed and the specified button is logically pressed when the specified modifier keys are logically down, and no other buttons or modifier keys are logically down.
2. The grab-window contains the pointer.
3. The confine-to window (if any) is viewable.
4. A passive grab on the same button/key combination does not exist on any ancestor of grab-window.

The interpretation of the remaining arguments is the same as for **GrabPointer**. The active grab is terminated automatically when the logical state of the pointer has all buttons released, independent of the logical state of modifier keys. Note that the logical state of a device (as seen by means of the protocol) may lag the physical state if device event processing is frozen.

This request overrides all previous passive grabs by the same client on the same button/key combinations on the same window. A modifier of **AnyModifier** is equivalent to issuing the request for all possible modifier combinations (including the combination of no modifiers). It is not required that all specified modifiers have currently assigned keycodes. A button of **AnyButton** is equivalent to issuing the request for all possible buttons. Otherwise, it is not required that the
An **Access** error is generated if some other client has already issued a **GrabButton** request with the same button/key combination on the same window. When using **AnyModifier** or **AnyButton**, the request fails completely (no grabs are established), and an **Access** error is generated if there is a conflicting grab for any combination. The request has no effect on an active grab.

**UngrabButton**

- **modifiers**: SETofKEYMASK or **AnyModifier**
- **button**: BUTTON or **AnyButton**
- **grab-window**: WINDOW

Errors: **Value**, **Window**

This request releases the passive button/key combination on the specified window if it was grabbed by this client. A modifiers argument of **AnyModifier** is equivalent to issuing the request for all possible modifier combinations (including the combination of no modifiers). A button of **AnyButton** is equivalent to issuing the request for all possible buttons. The request has no effect on an active grab.

**ChangeActivePointerGrab**

- **event-mask**: SETofPOINTEREVENT
- **cursor**: CURSOR or **None**
- **time**: TIMESTAMP or **CurrentTime**

Errors: **Cursor**, **Value**

This request changes the specified dynamic parameters if the pointer is actively grabbed by the client and the specified time is no earlier than the last-pointer-grab time and no later than the current server time. The interpretation of event-mask and cursor are the same as in **GrabPointer**. This request has no effect on the parameters of any passive grabs established with **GrabButton**.

**GrabKeyboard**

- **grab-window**: WINDOW
- **owner-events**: BOOL
- **pointer-mode**, **keyboard-mode**: {**Synchronous**, **Asynchronous**}
- **time**: TIMESTAMP or **CurrentTime**

→

- **status**: {**Success**, **AlreadyGrabbed**, **Frozen**, **InvalidTime**, **NotViewable**}

Errors: **Value**, **Window**

This request actively grabs control of the keyboard. Further key events are reported only to the grabbing client. This request overrides any active keyboard grab by this client.

If **owner-events** is **False**, all generated key events are reported with respect to **grab-window**. If **owner-events** is **True** and if a generated key event would normally be reported to this client, it is reported normally. Otherwise, the event is reported with respect to the **grab-window**. Both
KeyPress and KeyRelease events are always reported, independent of any event selection made by the client.

If keyboard-mode is Asynchronous, keyboard event processing continues normally. If the keyboard is currently frozen by this client, then processing of keyboard events is resumed. If keyboard-mode is Synchronous, the state of the keyboard (as seen by means of the protocol) appears to freeze. No further keyboard events are generated by the server until the grabbing client issues a releasing AllowEvents request or until the keyboard grab is released. Actual keyboard changes are not lost while the keyboard is frozen. They are simply queued for later processing.

If pointer-mode is Asynchronous, pointer event processing is unaffected by activation of the grab. If pointer-mode is Synchronous, the state of the pointer (as seen by means of the protocol) appears to freeze. No further pointer events are generated by the server until the grabbing client issues a releasing AllowEvents request or until the keyboard grab is released. Actual pointer changes are not lost while the pointer is frozen. They are simply queued for later processing.

This request generates FocusIn and FocusOut events.

The request fails with status AlreadyGrabbed if the keyboard is actively grabbed by some other client. The request fails with status Frozen if the keyboard is frozen by an active grab of another client. The request fails with status NotViewable if grab-window is not viewable. The request fails with status InvalidTime if the specified time is earlier than the last-keyboard-grab time or later than the current server time. Otherwise, the last-keyboard-grab time is set to the specified time with CurrentTime replaced by the current server time.

UngrabKeyboard

time: TIMESTAMP or CurrentTime

This request releases the keyboard if this client has it actively grabbed (as a result of either GrabKeyboard or GrabKey) and releases any queued events. The request has no effect if the specified time is earlier than the last-keyboard-grab time or is later than the current server time.

This request generates FocusIn and FocusOut events.

An UngrabKeyboard is performed automatically if the event window for an active keyboard grab becomes not viewable.

GrabKey

key: KEYCODE or AnyKey
modifiers: SETofKEYMASK or AnyModifier
grab-window: WINDOW
owner-events: BOOL
pointer-mode, keyboard-mode: {Synchronous, Asynchronous}

Errors: Access, Value, Window

This request establishes a passive grab on the keyboard. In the future, the keyboard is actively grabbed as described in GrabKeyboard, the last-keyboard-grab time is set to the time at which the key was pressed (as transmitted in the KeyPress event), and the KeyPress event is reported if all of the following conditions are true:
The keyboard is not grabbed and the specified key (which can itself be a modifier key) is logically pressed when the specified modifier keys are logically down, and no other modifier keys are logically down.

Either the grab-window is an ancestor of (or is) the focus window, or the grab-window is a descendent of the focus window and contains the pointer.

A passive grab on the same key combination does not exist on any ancestor of grab-window.

The interpretation of the remaining arguments is the same as for **GrabKeyboard**. The active grab is terminated automatically when the logical state of the keyboard has the specified key released, independent of the logical state of modifier keys. Note that the logical state of a device (as seen by means of the protocol) may lag the physical state if device event processing is frozen.

This request overrides all previous passive grabs by the same client on the same key combinations on the same window. A modifier of **AnyModifier** is equivalent to issuing the request for all possible modifier combinations (including the combination of no modifiers). It is not required that all modifiers specified have currently assigned keycodes. A key of **AnyKey** is equivalent to issuing the request for all possible keycodes. Otherwise, the key must be in the range specified by min-keycode and max-keycode in the connection setup (or a **Value** error results).

An **Access** error is generated if some other client has issued a **GrabKey** with the same key combination on the same window. When using **AnyModifier** or **AnyKey**, the request fails completely (no grabs are established), and an **Access** error is generated if there is a conflicting grab for any combination.

---

### UngrabKey

*key*: KEYCODE or **AnyKey**  
*modifiers*: SETofKEYMASK or **AnyModifier**  
*grab-window*: WINDOW

**Errors**: **Value**, **Window**

This request releases the key combination on the specified window if it was grabbed by this client. A modifiers argument of **AnyModifier** is equivalent to issuing the request for all possible modifier combinations (including the combination of no modifiers). A key of **AnyKey** is equivalent to issuing the request for all possible keycodes. This request has no effect on an active grab.

### AllowEvents

*mode*:  
  - **AsyncPointer**  
  - **SyncPointer**  
  - **ReplayPointer**  
  - **AsyncKeyboard**  
  - **SyncKeyboard**  
  - **ReplayKeyboard**  
  - **AsyncBoth**  
  - **SyncBoth**

*time*: TIMESTAMP or **CurrentTime**

**Errors**: **Value**

This request releases some queued events if the client has caused a device to freeze. The request has no effect if the specified time is earlier than the last-grab time of the most recent active grab for the client or if the specified time is later than the current server time.

For **AsyncPointer**, if the pointer is frozen by the client, pointer event processing continues normally. If the pointer is frozen twice by the client on behalf of two separate grabs, **AsyncPointer**
thaws for both. **AsyncPointer** has no effect if the pointer is not frozen by the client, but the pointer need not be grabbed by the client.

For **SyncPointer**, if the pointer is frozen and actively grabbed by the client, pointer event processing continues normally until the next **ButtonPress** or **ButtonRelease** event is reported to the client, at which time the pointer again appears to freeze. However, if the reported event causes the pointer grab to be released, then the pointer does not freeze. **SyncPointer** has no effect if the pointer is not frozen by the client or if the pointer is not grabbed by the client.

For **ReplayPointer**, if the pointer is actively grabbed by the client and is frozen as the result of an event having been sent to the client (either from the activation of a **GrabButton** or from a previous **AllowEvents** with mode **SyncPointer** but not from a **GrabPointer**), then the pointer grab is released and that event is completely reprocessed, this time ignoring any passive grabs at or above (towards the root) the grab-window of the grab just released. The request has no effect if the pointer is not grabbed by the client or if the pointer is not frozen as the result of an event.

For **AsyncKeyboard**, if the keyboard is frozen by the client, keyboard event processing continues normally. If the keyboard is frozen twice by the client on behalf of two separate grabs, **AsyncKeyboard** thaws for both. **AsyncKeyboard** has no effect if the keyboard is not frozen by the client, but the keyboard need not be grabbed by the client.

For **SyncKeyboard**, if the keyboard is frozen and actively grabbed by the client, keyboard event processing continues normally until the next **KeyPress** or **KeyRelease** event is reported to the client, at which time the keyboard again appears to freeze. However, if the reported event causes the keyboard grab to be released, then the keyboard does not freeze. **SyncKeyboard** has no effect if the keyboard is not frozen by the client or if the keyboard is not grabbed by the client.

For **ReplayKeyboard**, if the keyboard is actively grabbed by the client and is frozen as the result of an event having been sent to the client (either from the activation of a **GrabKey** or from a previous **AllowEvents** with mode **SyncKeyboard** but not from a **GrabKeyboard**), then the keyboard grab is released and that event is completely reprocessed, this time ignoring any passive grabs at or above (towards the root) the grab-window of the grab just released. The request has no effect if the keyboard is not grabbed by the client or if the keyboard is not frozen as the result of an event.

For **SyncBoth**, if both pointer and keyboard are frozen by the client, event processing (for both devices) continues normally until the next **ButtonPress**, **ButtonRelease**, **KeyPress**, or **KeyRelease** event is reported to the client for a grabbed device (button event for the pointer, key event for the keyboard), at which time the devices again appear to freeze. However, if the reported event causes the grab to be released, then the devices do not freeze (but if the other device is still grabbed, then a subsequent event for it will still cause both devices to freeze). **SyncBoth** has no effect unless both pointer and keyboard are frozen by the client. If the pointer or keyboard is frozen twice by the client on behalf of two separate grabs, **SyncBoth** thaws for both (but a subsequent freeze for **SyncBoth** will only freeze each device once).

For **AsyncBoth**, if the pointer and keyboard are frozen by the client, event processing for both devices continues normally. If a device is frozen twice by the client on behalf of two separate grabs, **AsyncBoth** thaws for both. **AsyncBoth** has no effect unless both pointer and keyboard are frozen by the client.

**AsyncPointer**, **SyncPointer**, and **ReplayPointer** have no effect on processing of keyboard events. **AsyncKeyboard**, **SyncKeyboard**, and **ReplayKeyboard** have no effect on processing of pointer events.

It is possible for both a pointer grab and a keyboard grab to be active simultaneously (by the same or different clients). When a device is frozen on behalf of either grab, no event processing is performed for the device. It is possible for a single device to be frozen because of both grabs. In this
case, the freeze must be released on behalf of both grabs before events can again be processed. If a device is frozen twice by a single client, then a single **AllowEvents** releases both.

**GrabServer**

This request disables processing of requests and close-downs on all connections other than the one this request arrived on.

**UngrabServer**

This request restarts processing of requests and close-downs on other connections.

**QueryPointer**

\[
\text{window: WINDOW}
\]

→

\[
\begin{align*}
\text{root: WINDOW} \\
\text{child: WINDOW or None} \\
\text{same-screen: BOOL} \\
\text{root-x, root-y, win-x, win-y: INT16} \\
\text{mask: SETofKEYBUTMASK}
\end{align*}
\]

**Errors:** **Window**

The root window the pointer is logically on and the pointer coordinates relative to the root’s origin are returned. If same-screen is **False**, then the pointer is not on the same screen as the argument window, child is **None**, and win-x and win-y are zero. If same-screen is **True**, then win-x and win-y are the pointer coordinates relative to the argument window’s origin, and child is the child containing the pointer, if any. The current logical state of the modifier keys and the buttons are also returned. Note that the logical state of a device (as seen by means of the protocol) may lag the physical state if device event processing is frozen.

**GetMotionEvents**

\[
\begin{align*}
\text{start, stop: TIMESTAMP or CurrentTime} \\
\text{window: WINDOW}
\end{align*}
\]

→

\[
\text{events: LISTofTIMECOORD}
\]

**where:**

\[
\begin{align*}
\text{TIMECOORD: } [x, y: INT16] \\
\text{time: TIMESTAMP}
\end{align*}
\]

**Errors:** **Window**
This request returns all events in the motion history buffer that fall between the specified start and stop times (inclusive) and that have coordinates that lie within (including borders) the specified window at its present placement. The x and y coordinates are reported relative to the origin of the window.

If the start time is later than the stop time or if the start time is in the future, no events are returned. If the stop time is in the future, it is equivalent to specifying CurrentTime.

TranslateCoordinates

\[ \text{src-window, dst-window: WINDOW} \]
\[ \text{src-x, src-y: INT16} \]
\[ \rightarrow \]
\[ \text{same-screen: BOOL} \]
\[ \text{child: WINDOW or None} \]
\[ \text{dst-x, dst-y: INT16} \]

Errors: Window

The src-x and src-y coordinates are taken relative to src-window’s origin and are returned as dst-x and dst-y coordinates relative to dst-window’s origin. If same-screen is False, then src-window and dst-window are on different screens, and dst-x and dst-y are zero. If the coordinates are contained in a mapped child of dst-window, then that child is returned.

WarpPointer

\[ \text{src-window: WINDOW or None} \]
\[ \text{dst-window: WINDOW or None} \]
\[ \text{src-x, src-y: INT16} \]
\[ \text{src-width, src-height: CARD16} \]
\[ \text{dst-x, dst-y: INT16} \]

Errors: Window

If dst-window is None, this request moves the pointer by offsets [dst-x, dst-y] relative to the current position of the pointer. If dst-window is a window, this request moves the pointer to [dst-x, dst-y] relative to dst-window’s origin. However, if src-window is not None, the move only takes place if src-window contains the pointer and the pointer is contained in the specified rectangle of src-window.

The src-x and src-y coordinates are relative to src-window’s origin. If src-height is zero, it is replaced with the current height of src-window minus src-y. If src-width is zero, it is replaced with the current width of src-window minus src-x.

This request cannot be used to move the pointer outside the confine-to window of an active pointer grab. An attempt will only move the pointer as far as the closest edge of the confine-to window.

This request will generate events just as if the user had instantaneously moved the pointer.
**SetInputFocus**

- **focus**: WINDOW or PointerRoot or None
- **revert-to**: \{ Parent, PointerRoot, None \}
- **time**: TIMESTAMP or CurrentTime

Errors: Match, Value, Window

This request changes the input focus and the last-focus-change time. The request has no effect if the specified time is earlier than the current last-focus-change time or is later than the current server time. Otherwise, the last-focus-change time is set to the specified time with CurrentTime replaced by the current server time.

If **None** is specified as the focus, all keyboard events are discarded until a new focus window is set. In this case, the revert-to argument is ignored.

If a window is specified as the focus, it becomes the keyboard’s focus window. If a generated keyboard event would normally be reported to this window or one of its inferiors, the event is reported normally. Otherwise, the event is reported with respect to the focus window.

If **PointerRoot** is specified as the focus, the focus window is dynamically taken to be the root window of whatever screen the pointer is on at each keyboard event. In this case, the revert-to argument is ignored.

This request generates **FocusIn** and **FocusOut** events.

The specified focus window must be viewable at the time of the request (or a Match error results). If the focus window later becomes not viewable, the new focus window depends on the revert-to argument. If revert-to is Parent, the focus reverts to the parent (or the closest viewable ancestor) and the new revert-to value is taken to be **None**. If revert-to is PointerRoot or None, the focus reverts to that value. When the focus reverts, **FocusIn** and **FocusOut** events are generated, but the last-focus-change time is not affected.

**GetInputFocus**

→

- **focus**: WINDOW or PointerRoot or None
- **revert-to**: \{ Parent, PointerRoot, None \}

This request returns the current focus state.

**QueryKeymap**

→

- **keys**: LISTofCARD8

This request returns a bit vector for the logical state of the keyboard. Each bit set to 1 indicates that the corresponding key is currently pressed. The vector is represented as 32 bytes. Byte N (from 0) contains the bits for keys 8N to 8N + 7 with the least significant bit in the byte representing key 8N. Note that the logical state of a device (as seen by means of the protocol) may lag the physical state if device event processing is frozen.
OpenFont

\[ fid: \text{FONT} \]
\[ name: \text{STRING8} \]

Errors: Alloc, IDChoice, Name

This request loads the specified font, if necessary, and associates identifier \( fid \) with it. The font name should use the ISO Latin-1 encoding, and uppercase and lowercase do not matter. When the characters ‘?’ and ‘*’ are used in a font name, a pattern match is performed and any matching font is used. In the pattern, the ‘?’ character (octal value 77) will match any single character, and the ‘*’ character (octal value 52) will match any number of characters. A structured format for font names is specified in the X.Org standard \( X \) Logical Font Description Conventions.

Fonts are not associated with a particular screen and can be stored as a component of any graphics context.

CloseFont

\[ font: \text{FONT} \]

Errors: Font

This request deletes the association between the resource ID and the font. The font itself will be freed when no other resource references it.
QueryFont

\(\text{font}: \text{FONTABLE} \rightarrow \)

\(\text{font-info}: \text{FONTINFO} \)

\(\text{char-infos}: \text{LISTofCHARINFO} \)

where:

\[
\text{FONTINFO}: \begin{align*}
\begin{cases}
\text{draw-direction: } \{ \text{LeftToRight, RightToLeft} \} \\
\text{min-char-or-byte2, max-char-or-byte2}: \text{CARD16} \\
\text{min-byte1, max-byte1}: \text{CARD8} \\
\text{allCharsExist}: \text{BOOL} \\
\text{defaultChar}: \text{CARD16} \\
\text{minBounds}: \text{CHARINFO} \\
\text{maxBounds}: \text{CHARINFO} \\
\text{fontAscent}: \text{INT16} \\
\text{fontDescent}: \text{INT16} \\
\text{properties}: \text{LISTofFONTPROP}
\end{cases}
\end{align*}
\]

\[
\text{FONTPROP}: \begin{align*}
\text{name}: \text{ATOM} \\
\text{value}: \langle\text{32-bit-value}\rangle
\end{align*}
\]

\[
\text{CHARINFO}: \begin{align*}
\text{leftSideBearing}: \text{INT16} \\
\text{rightSideBearing}: \text{INT16} \\
\text{characterWidth}: \text{INT16} \\
\text{ascent}: \text{INT16} \\
\text{descent}: \text{INT16} \\
\text{attributes}: \text{CARD16}
\end{align*}
\]

Errors: Font

This request returns logical information about a font. If a gcontext is given for font, the currently contained font is used.

The draw-direction is just a hint and indicates whether most char-infos have a positive, \texttt{LeftToRight}, or a negative, \texttt{RightToLeft}, character-width metric. The core protocol defines no support for vertical text.

If \text{min-byte1} and \text{max-byte1} are both zero, then \text{min-char-or-byte2} specifies the linear character index corresponding to the first element of char-infos, and \text{max-char-or-byte2} specifies the linear character index of the last element. If either \text{min-byte1} or \text{max-byte1} are nonzero, then both \text{min-char-or-byte2} and \text{max-char-or-byte2} will be less than 256, and the 2-byte character index values corresponding to char-infos element \(N\) (counting from 0) are:

\[
\begin{align*}
\text{byte1} &= N/D + \text{min-byte1} \\
\text{byte2} &= N\backslash D + \text{min-char-or-byte2}
\end{align*}
\]

where:

\[
\begin{align*}
D &= \text{max-char-or-byte2} - \text{min-char-or-byte2} + 1 \\
/ &= \text{integer division} \\
\backslash &= \text{integer modulus}
\end{align*}
\]

If char-infos has length zero, then \text{min-bounds} and \text{max-bounds} will be identical, and the effective char-infos is one filled with this char-info, of length:
L = D * (max-byte1 − min-byte1 + 1)

That is, all glyphs in the specified linear or matrix range have the same information, as given by min-bounds (and max-bounds). If all-chars-exist is True, then all characters in char-infos have nonzero bounding boxes.

The default-char specifies the character that will be used when an undefined or nonexistent character is used. Note that default-char is a CARD16, not CHAR2B. For a font using 2-byte matrix format, the default-char has byte1 in the most significant byte and byte2 in the least significant byte. If the default-char itself specifies an undefined or nonexistent character, then no printing is performed for an undefined or nonexistent character.

The min-bounds and max-bounds contain the minimum and maximum values of each individual CHARINFO component over all char-infos (ignoring nonexistent characters). The bounding box of the font (that is, the smallest rectangle enclosing the shape obtained by superimposing all characters at the same origin [x,y]) has its upper-left coordinate at:

\[ [x + \text{min-bounds.left-side-bearing}, y - \text{max-bounds.ascent}] \]

with a width of:

\[ \text{max-bounds.right-side-bearing} - \text{min-bounds.left-side-bearing} \]

and a height of:

\[ \text{max-bounds.ascent} + \text{max-bounds.descent} \]

The font-ascent is the logical extent of the font above the baseline and is used for determining line spacing. Specific characters may extend beyond this. The font-descent is the logical extent of the font at or below the baseline and is used for determining line spacing. Specific characters may extend beyond this. If the baseline is at Y-coordinate y, then the logical extent of the font is inclusive between the Y-coordinate values (y − font-ascent) and (y + font-descent − 1).

A font is not guaranteed to have any properties. The interpretation of the property value (for example, INT32, CARD32) must be derived from a priori knowledge of the property. A basic set of font properties is specified in the X.Org standard X Logical Font Description Conventions.

For a character origin at [x,y], the bounding box of a character (that is, the smallest rectangle enclosing the character’s shape), described in terms of CHARINFO components, is a rectangle with its upper-left corner at:

\[ [x + \text{left-side-bearing}, y - \text{ascent}] \]

with a width of:

\[ \text{right-side-bearing} - \text{left-side-bearing} \]

and a height of:

\[ \text{ascent} + \text{descent} \]

and the origin for the next character is defined to be:

\[ [x + \text{character-width}, y] \]

Note that the baseline is logically viewed as being just below nondescending characters (when descent is zero, only pixels with Y-coordinates less than y are drawn) and that the origin is logically viewed as being coincident with the left edge of a nonkerned character (when left-side-bearing is zero, no pixels with X-coordinate less than x are drawn).
Note that CHARINFO metric values can be negative. A nonexistent character is represented with all CHARINFO components zero. The interpretation of the per-character attributes field is server-dependent.

**QueryTextExtents**

`font: FONTABLE
string: STRING16

→
draw-direction: {LeftToRight, RightToLeft}
font-ascent: INT16
font-descent: INT16
overall-ascent: INT16
overall-descent: INT16
overall-width: INT32
overall-left: INT32
overall-right: INT32

Errors: Font

This request returns the logical extents of the specified string of characters in the specified font. If a gcontext is given for font, the currently contained font is used. The draw-direction, font-ascent, and font-descent are the same as described in **QueryFont**. The overall-ascent is the maximum of the ascent metrics of all characters in the string, and the overall-descent is the maximum of the descent metrics. The overall-width is the sum of the character-width metrics of all characters in the string. For each character in the string, let W be the sum of the character-width metrics of all characters preceding it in the string, let L be the left-side-bearing metric of the character plus W, and let R be the right-side-bearing metric of the character plus W. The overall-left is the minimum L of all characters in the string, and the overall-right is the maximum R.

For fonts defined with linear indexing rather than 2-byte matrix indexing, the server will interpret each CHAR2B as a 16-bit number that has been transmitted most significant byte first (that is, byte1 of the CHAR2B is taken as the most significant byte).

Characters with all zero metrics are ignored. If the font has no defined default-char, then undefined characters in the string are also ignored.

**ListFonts**

`pattern: STRING8
max-names: CARD16

→

names: LISTofSTRING8

This request returns a list of available font names (as controlled by the font search path; see **SetFontPath** request) that match the pattern. At most, max-names names will be returned. The pattern should use the ISO Latin-1 encoding, and uppercase and lowercase do not matter. In the pattern, the “?” character (octal value 77) will match any single character, and the “*” character (octal value 52) will match any number of characters. The returned names are in lowercase.
**ListFontsWithInfo**

- pattern: STRING8
- max-names: CARD16

→

- name: STRING8
- info: FONTINFO
- replies-hint: CARD32

where:

- FONTINFO: <same type definition as in QueryFont>

This request is similar to ListFonts, but it also returns information about each font. The information returned for each font is identical to what QueryFont would return except that the per-character metrics are not returned. Note that this request can generate multiple replies. With each reply, replies-hint may provide an indication of how many more fonts will be returned. This number is a hint only and may be larger or smaller than the number of fonts actually returned. A zero value does not guarantee that no more fonts will be returned. After the font replies, a reply with a zero-length name is sent to indicate the end of the reply sequence.

**SetFontPath**

- path: LISTofSTRING8

Errors: Value

This request defines the search path for font lookup. There is only one search path per server, not one per client. The interpretation of the strings is operating-system-dependent, but the strings are intended to specify directories to be searched in the order listed.

Setting the path to the empty list restores the default path defined for the server.

As a side effect of executing this request, the server is guaranteed to flush all cached information about fonts for which there currently are no explicit resource IDs allocated.

The meaning of an error from this request is system specific.

**GetFontPath**

→

- path: LISTofSTRING8

This request returns the current search path for fonts.
CreatePixmap

\[ pid: \text{PIXMAP} \]
\[ drawable: \text{DRAWABLE} \]
\[ depth: \text{CARD8} \]
\[ width, height: \text{CARD16} \]

Errors: Alloc, Drawable, IDChoice, Value

This request creates a pixmap and assigns the identifier \( pid \) to it. The width and height must be nonzero (or a Value error results). The depth must be one of the depths supported by the root of the specified drawable (or a Value error results). The initial contents of the pixmap are undefined.

It is legal to pass an InputOnly window as a drawable to this request.

FreePixmap

\[ pixmap: \text{PIXMAP} \]

Errors: Pixmap

This request deletes the association between the resource ID and the pixmap. The pixmap storage will be freed when no other resource references it.

CreateGC

\[ cid: \text{GCONTEXT} \]
\[ drawable: \text{DRAWABLE} \]
\[ value-mask: \text{BITMASK} \]
\[ value-list: \text{LISTofVALUE} \]

Errors: Alloc, Drawable, Font, IDChoice, Match, Pixmap, Value

This request creates a graphics context and assigns the identifier \( cid \) to it. The gcontext can be used with any destination drawable having the same root and depth as the specified drawable; use with other drawables results in a Match error.

The value-mask and value-list specify which components are to be explicitly initialized. The context components are:

<table>
<thead>
<tr>
<th>Component</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>function</td>
<td>{ Clear, And, AndReverse, Copy, AndInverted, NoOp, Xor, Or, Nor, Equiv, Invert, OrReverse, CopyInverted, OrInverted, Nand, Set }</td>
</tr>
<tr>
<td>plane-mask</td>
<td>CARD32</td>
</tr>
<tr>
<td>foreground</td>
<td>CARD32</td>
</tr>
<tr>
<td>background</td>
<td>CARD32</td>
</tr>
<tr>
<td>line-width</td>
<td>CARD16</td>
</tr>
<tr>
<td>line-style</td>
<td>{ Solid, OnOffDash, DoubleDash }</td>
</tr>
</tbody>
</table>
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:

\[
((\text{src } \text{FUNC} \text{ dst}) \text{ AND } \text{plane-mask}) \text{ OR } (\text{dst } \text{AND} (\text{NOT } \text{plane-mask}))
\]

Range checking is not performed on the values for foreground, background, or plane-mask. They are simply truncated to the appropriate number of bits.

The meanings of the functions are:

<table>
<thead>
<tr>
<th>Function</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear</td>
<td>0</td>
</tr>
<tr>
<td>And</td>
<td>src AND dst</td>
</tr>
<tr>
<td>AndReverse</td>
<td>src AND (NOT dst)</td>
</tr>
<tr>
<td>Copy</td>
<td>src</td>
</tr>
<tr>
<td>AndInverted</td>
<td>(NOT src) AND dst</td>
</tr>
<tr>
<td>NoOp</td>
<td>dst</td>
</tr>
<tr>
<td>Xor</td>
<td>src XOR dst</td>
</tr>
<tr>
<td>Or</td>
<td>src OR dst</td>
</tr>
<tr>
<td>Nor</td>
<td>(NOT src) AND (NOT dst)</td>
</tr>
<tr>
<td>Equiv</td>
<td>(NOT src) XOR dst</td>
</tr>
<tr>
<td>Invert</td>
<td>NOT dst</td>
</tr>
<tr>
<td>OrReverse</td>
<td>src OR (NOT dst)</td>
</tr>
<tr>
<td>CopyInverted</td>
<td>NOT src</td>
</tr>
<tr>
<td>OrInverted</td>
<td>(NOT src) OR dst</td>
</tr>
<tr>
<td>Nand</td>
<td>(NOT src) OR (NOT dst)</td>
</tr>
<tr>
<td>Set</td>
<td>1</td>
</tr>
</tbody>
</table>
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.

Wide lines are drawn centered on the path described by the graphics request. Unless otherwise specified by the join or cap style, the bounding box of a wide line with endpoints \([x_1, y_1], [x_2, y_2]\) and width \(w\) is a rectangle with vertices at the following real coordinates:

\[
[x_1-(w*sn/2), y_1+(w*cs/2)], [x_1+(w*sn/2), y_1-(w*cs/2)],
[x_2-(w*sn/2), y_2+(w*cs/2)], [x_2+(w*sn/2), y_2-(w*cs/2)]
\]

The \(sn\) is the sine of the angle of the line and \(cs\) is the cosine of the angle of the line. A pixel is part of the line (and hence drawn) if the center of the pixel is fully inside the bounding box, which is viewed as having infinitely thin edges. If the center of the pixel is exactly on the bounding box, it is part of the line if and only if the interior is immediately to its right (x increasing direction). Pixels with centers on a horizontal edge are a special case and are part of the line if and only if the interior or the boundary is immediately below (y increasing direction) and if the interior or the boundary is immediately to the right (x increasing direction). Note that this description is a mathematical model describing the pixels that are drawn for a wide line and does not imply that trigonometry is required to implement such a model. Real or fixed point arithmetic is recommended for computing the corners of the line endpoints for lines greater than one pixel in width.

Thin lines (zero line-width) are nominally one pixel wide lines drawn using an unspecified, device-dependent algorithm. There are only two constraints on this algorithm. First, if a line is drawn unclipped from \([x_1,y_1]\) to \([x_2,y_2]\) and another line is drawn unclipped from \([x_1+dx,y_1+dy]\) to \([x_2+dx,y_2+dy]\), then a point \([x,y]\) is touched by drawing the first line if and only if the point \([x+dx,y+dy]\) is touched by drawing the second line. Second, the effective set of points comprising a line cannot be affected by clipping. Thus, a point is touched in a clipped line if and only if the point lies inside the clipping region and the point would be touched by the line when drawn unclipped.

Note that a wide line drawn from \([x_1,y_1]\) to \([x_2,y_2]\) always draws the same pixels as a wide line drawn from \([x_2,y_2]\) to \([x_1,y_1]\), not counting cap-style and join-style. Implementors are encouraged to make this property true for thin lines, but it is not required. A line-width of zero may differ from a line-width of one in which pixels are drawn. In general, drawing a thin line will be faster than drawing a wide line of width one, but thin lines may not mix well aesthetically with wide lines because of the different drawing algorithms. If it is desirable to obtain precise and uniform results across all displays, a client should always use a line-width of one, rather than a line-width of zero.

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**).

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.

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.

For a line with coincident endpoints (x1=x2, y1=y2), when the cap-style is applied to both endpoints, the semantics depends on the line-width and the cap-style:

**NotLast**  
thin  
This is device-dependent, but the desired effect is that nothing is drawn.

**Butt**  
thin  
This is device-dependent, but the desired effect is that a single pixel is drawn.

**Round**  
thin  
This is the same as **Butt/thin**.

**Projecting**  
thin  
This is the same as **Butt/thin**.

**Butt**  
wide  
Nothing is drawn.

**Round**  
wide  
The closed path is a circle, centered at the endpoint and with a diameter equal to the line-width.

**Projecting**  
wide  
The closed path is a square, aligned with the coordinate axes, centered at the endpoint and with sides equal to the line-width.

For a line with coincident endpoints (x1=x2, y1=y2), when the join-style is applied at one or both endpoints, the effect is as if the line was removed from the overall path. However, if the total path consists of (or is reduced to) a single point joined with itself, the effect is the same as when the cap-style is applied at both endpoints.

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

- \textbf{Solid} \hspace{1cm} Foreground
- \textbf{Tiled} \hspace{1cm} Tile
- \textbf{OpaqueStippled} \hspace{1cm} 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
- \textbf{Stippled} \hspace{1cm} Foreground masked by stipple

For the odd dashes for line requests with line-style \texttt{DoubleDash}:

- \textbf{Solid} \hspace{1cm} Background
- \textbf{Tiled} \hspace{1cm} Same as for even dashes
- \textbf{OpaqueStippled} \hspace{1cm} Same as for even dashes
- \textbf{Stippled} \hspace{1cm} Background masked by stipple

The dashes value allowed here is actually a simplified form of the more general patterns that can be set with \texttt{SetDashes}. Specifying a value of \texttt{N} here is equivalent to specifying the two element list \([N, N]\) in \texttt{SetDashes}. The value must be nonzero (or a \texttt{Value} error results). The meaning of dash-offset and dashes are explained in the \texttt{SetDashes} request.

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 \texttt{Match} error results). If clip-mask is \texttt{None}, then pixels are always drawn, regardless of the clip origin. The clip-mask can also be set with the \texttt{SetClipRectangles} request.

For \texttt{ClipByChildren}, both source and destination windows are additionally clipped by all viewable \texttt{InputOutput} children. For \texttt{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 \texttt{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.

The fill-rule defines what pixels are inside (that is, are drawn) for paths given in \texttt{FillPoly} requests. \texttt{EvenOdd} means a point is inside if an infinite ray with the point as origin crosses the path an odd number of times. For \texttt{Winding}, a point is inside if an infinite ray with the point as origin crosses an unequal number of clockwise and counterclockwise directed path segments. A clockwise directed path segment is one that crosses the ray from left to right as observed from the point. A counter-clockwise segment is one that crosses the ray from right to left as observed from the point. The case where a directed line segment is coincident with the ray is uninteresting because one can simply choose a different ray that is not coincident with a segment.

For both fill rules, a point is infinitely small and the path is an infinitely thin line. A pixel is inside if the center point of the pixel is inside and the center point is not on the boundary.
The center point is on the boundary, the pixel is inside if and only if the polygon interior is immediately to its right (x increasing direction). Pixels with centers along a horizontal edge are a special case and are inside if and only if the polygon interior is immediately below (y increasing direction).

The arc-mode controls filling in the **PolyFillArc** request.

The graphics-exposures flag controls **GraphicsExposure** event generation for **CopyArea** and **CopyPlane** requests (and any similar requests defined by extensions).

The default component values are:

<table>
<thead>
<tr>
<th>Component</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>function</td>
<td>Copy</td>
</tr>
<tr>
<td>plane-mask</td>
<td>all ones</td>
</tr>
<tr>
<td>foreground</td>
<td>0</td>
</tr>
<tr>
<td>background</td>
<td>1</td>
</tr>
<tr>
<td>line-width</td>
<td>0</td>
</tr>
<tr>
<td>line-style</td>
<td>Solid</td>
</tr>
<tr>
<td>cap-style</td>
<td>Butt</td>
</tr>
<tr>
<td>join-style</td>
<td>Miter</td>
</tr>
<tr>
<td>fill-style</td>
<td>Solid</td>
</tr>
<tr>
<td>fill-rule</td>
<td>EvenOdd</td>
</tr>
<tr>
<td>arc-mode</td>
<td>PieSlice</td>
</tr>
<tr>
<td>tile</td>
<td>Pixmap of unspecified size filled with foreground pixel (that is, client specified pixel if any, else 0) (subsequent changes to foreground do not affect this pixmap)</td>
</tr>
<tr>
<td>stipple</td>
<td>Pixmap of unspecified size filled with ones</td>
</tr>
<tr>
<td>tile-stipple-x-origin</td>
<td>0</td>
</tr>
<tr>
<td>tile-stipple-y-origin</td>
<td>0</td>
</tr>
<tr>
<td>font</td>
<td>&lt;server-dependent-font&gt;</td>
</tr>
<tr>
<td>subwindow-mode</td>
<td>ClipByChildren</td>
</tr>
<tr>
<td>graphics-exposures</td>
<td>True</td>
</tr>
<tr>
<td>clip-x-origin</td>
<td>0</td>
</tr>
<tr>
<td>clip-y-origin</td>
<td>0</td>
</tr>
<tr>
<td>clip-mask</td>
<td>None</td>
</tr>
<tr>
<td>dash-offset</td>
<td>0</td>
</tr>
<tr>
<td>dashes</td>
<td>4 (that is, the list [4, 4])</td>
</tr>
</tbody>
</table>

Storing a pixmap in a gccontext might or might not result in a copy being made. If the pixmap is later used as the destination for a graphics request, the change might or might not be reflected in the gccontext. If the pixmap is used simultaneously in a graphics request as both a destination and as a tile or stipple, the results are not defined.

It is quite likely that some amount of gccontext information will be cached in display hardware and that such hardware can only cache a small number of gccontexts. Given the number and complexity of components, clients should view switching between gccontexts with nearly identical state as significantly more expensive than making minor changes to a single gccontext.
ChangeGC

\[ gc: \text{GCONTEXT} \]
\[ \text{value-mask}: \text{BITMASK} \]
\[ \text{value-list}: \text{LISTofVALUE} \]

Errors: Alloc, Font, GContext, Match, Pixmap, Value

This request changes components in gc. The value-mask and value-list specify which components are to be changed. The values and restrictions are the same as for CreateGC.

Changing the clip-mask also overrides any previous SetClipRectangles request on the context.

Changing dash-offset or dashes overrides any previous SetDashes request on the context.

The order in which components are verified and altered is server-dependent. If an error is generated, a subset of the components may have been altered.

CopyGC

\[ \text{src-gc}, \text{dst-gc}: \text{GCONTEXT} \]
\[ \text{value-mask}: \text{BITMASK} \]

Errors: Alloc, GContext, Match, Value

This request copies components from src-gc to dst-gc. The value-mask specifies which components to copy, as for CreateGC. The two gcontexts must have the same root and the same depth (or a Match error results).

SetDashes

\[ gc: \text{GCONTEXT} \]
\[ \text{dash-offset}: \text{CARD16} \]
\[ \text{dashes}: \text{LISTofCARD8} \]

Errors: Alloc, GContext, Value

This request sets dash-offset and dashes in gc for dashed line styles. Dashes cannot be empty (or a Value error results). Specifying an odd-length list is equivalent to specifying the same list concatenated with itself to produce an even-length list. The initial and alternating elements of dashes are the even dashes; the others are the odd dashes. Each element specifies a dash length in pixels. All of the elements must be nonzero (or a Value error results). The dash-offset defines the phase of the pattern, specifying how many pixels into dashes the pattern should actually begin in any single graphics request. Dashing is continuous through path elements combined with a join-style but is reset to the dash-offset between each sequence of joined lines.

The unit of measure for dashes is the same as in the ordinary coordinate system. Ideally, a dash length is measured along the slope of the line, but implementations are only required to match this ideal for horizontal and vertical lines. Failing the ideal semantics, it is suggested that the length be measured along the major axis of the line. The major axis is defined as the x axis for lines drawn at an angle of between \(-45\) and \(+45\) degrees or between \(135\) and \(225\) degrees from the x axis. For all other lines, the major axis is the y axis.

For any graphics primitive, the computation of the endpoint of an individual dash only depends on the geometry of the primitive, the start position of the dash, the direction of the dash, and the
Dash length.

For any graphics primitive, the total set of pixels used to render the primitive (both even and odd numbered dash elements) with DoubleDash line-style is the same as the set of pixels used to render the primitive with Solid line-style.

For any graphics primitive, if the primitive is drawn with OnOffDash or DoubleDash line-style unclipped at position [x,y] and again at position [x+dx,y+dy], then a point [x1,y1] is included in a dash in the first instance if and only if the point [x1+dx,y1+dy] is included in the dash in the second instance. In addition, the effective set of points comprising a dash cannot be affected by clipping. A point is included in a clipped dash if and only if the point lies inside the clipping region and the point would be included in the dash when drawn unclipped.

**SetClipRectangles**

```
gc: GCONTEXT
clip-x-origin, clip-y-origin: INT16
rectangles: LISTofRECTANGLE
ordering: { UnSorted, YSorted, YXSorted, YXBanded }
```

Errors: Alloc, GContext, Match, Value

This request changes clip-mask in gc to the specified list of rectangles and sets the clip origin. Output will be clipped to remain contained within the rectangles. The clip origin is interpreted relative to the origin of whatever destination drawable is specified in a graphics request. The rectangle coordinates are interpreted relative to the clip origin. The rectangles should be nonintersecting, or graphics results will be undefined. Note that the list of rectangles can be empty, which effectively disables output. This is the opposite of passing None as the clip-mask in CreateGC and ChangeGC.

If known by the client, ordering relations on the rectangles can be specified with the ordering argument. This may provide faster operation by the server. If an incorrect ordering is specified, the server may generate a Match error, but it is not required to do so. If no error is generated, the graphics results are undefined. UnSorted means that the rectangles are in arbitrary order. YSorted means that the rectangles are nondecreasing in their Y origin. YXSorted additionally constrains YSorted order in that all rectangles with an equal Y origin are nondecreasing in their X origin. YXBanded additionally constrains YXSorted by requiring that, for every possible Y scanline, all rectangles that include that scanline have identical Y origins and Y extents.

**FreeGC**

```
gc: GCONTEXT
```

Errors: GContext

This request deletes the association between the resource ID and the gcontext and destroys the gcontext.
**ClearArea**

*window*: WINDOW  
*x, y*: INT16  
*width, height*: CARD16  
*exposures*: BOOL

Errors: **Match**, **Value**, **Window**

The x and y coordinates are relative to the window’s origin and specify the upper-left corner of the rectangle. If width is zero, it is replaced with the current width of the window minus x. If height is zero, it is replaced with the current height of the window minus y. If the window has a defined background tile, the rectangle is tiled with a plane-mask of all ones and function of **Copy** and a subwindow-mode of **ClipByChildren**. If the window has background **None**, the contents of the window are not changed. In either case, if exposures is **True**, then one or more exposure events are generated for regions of the rectangle that are either visible or are being retained in a backing store.

It is a **Match** error to use an **InputOnly** window in this request.

**CopyArea**

*src-drawable, dst-drawable*: DRAWABLE  
*gc*: GCOUNTXT  
*src-x, src-y*: INT16  
*width, height*: CARD16  
*dst-x, dst-y*: INT16

Errors: **Drawable**, **GContext**, **Match**

This request combines the specified rectangle of src-drawable with the specified rectangle of dst-drawable. The src-x and src-y coordinates are relative to src-drawable’s origin. The dst-x and dst-y are relative to dst-drawable’s origin, each pair specifying the upper-left corner of the rectangle. The src-drawable must have the same root and the same depth as dst-drawable (or a **Match** error results).

If regions of the source rectangle are obscured and have not been retained in backing store or if regions outside the boundaries of the source drawable are specified, then those regions are not copied, but the following occurs on all corresponding destination regions that are either visible or are retained in backing-store. If the dst-drawable is a window with a background other than **None**, these corresponding destination regions are tiled (with plane-mask of all ones and function **Copy**) with that background. Regardless of tiling and whether the destination is a window or a pixmap, if graphics-exposures in gc is **True**, then **GraphicsExposure** events for all corresponding destination regions are generated.

If graphics-exposures is **True** but no **GraphicsExposure** events are generated, then a **NoExposure** event is generated.

GC components: function, plane-mask, subwindow-mode, graphics-exposures, clip-x-origin, clip-y-origin, clip-mask
CopyPlane

\[ \text{src-drawable}, \text{dst-drawable}: \text{DRAWABLE} \]
\[ \text{gc}: \text{GCONTEXT} \]
\[ \text{src-x}, \text{src-y}: \text{INT16} \]
\[ \text{width}, \text{height}: \text{CARD16} \]
\[ \text{dst-x}, \text{dst-y}: \text{INT16} \]
\[ \text{bit-plane}: \text{CARD32} \]

Errors: Drawable, GContext, Match, Value

The src-drawable must have the same root as dst-drawable (or a Match error results), but it need not have the same depth. The bit-plane must have exactly one bit set to 1 and the value of bit-plane must be less than \(2^n\) where \(n\) is the depth of src-drawable (or a Value error results). Effectively, a pixmap of the same depth as dst-drawable and with size specified by the source region is formed using the foreground/background pixels in gc (foreground everywhere the bit-plane in src-drawable contains a bit set to 1, background everywhere the bit-plane contains a bit set to 0), and the equivalent of a CopyArea is performed, with all the same exposure semantics. This can also be thought of as using the specified region of the source bit-plane as a stipple with a fill-style of OpaqueStippled for filling a rectangular area of the destination.

GC components: function, plane-mask, foreground, background, subwindow-mode, graphics-exposures, clip-x-origin, clip-y-origin, clip-mask

PolyPoint

\[ \text{drawable}: \text{DRAWABLE} \]
\[ \text{gc}: \text{GCONTEXT} \]
\[ \text{coordinate-mode}: \{ \text{Origin, Previous} \} \]
\[ \text{points}: \text{LISTofPOINT} \]

Errors: Drawable, GContext, Match, Value

This request combines the foreground pixel in gc with the pixel at each point in the drawable. The points are drawn in the order listed.

The first point is always relative to the drawable’s origin. The rest are relative either to that origin or the previous point, depending on the coordinate-mode.

GC components: function, plane-mask, foreground, subwindow-mode, clip-x-origin, clip-y-origin, clip-mask

PolyLine

\[ \text{drawable}: \text{DRAWABLE} \]
\[ \text{gc}: \text{GCONTEXT} \]
\[ \text{coordinate-mode}: \{ \text{Origin, Previous} \} \]
\[ \text{points}: \text{LISTofPOINT} \]

Errors: Drawable, GContext, Match, Value

This request draws lines between each pair of points (\(\text{point}[i], \text{point}[i+1]\)). The lines are drawn in the order listed. The lines join correctly at all intermediate points, and if the first and last points
coincide, the first and last lines also join correctly. For any given line, no pixel is drawn more than once. If thin (zero line-width) lines intersect, the intersecting pixels are drawn multiple times. If wide lines intersect, the intersecting pixels are drawn only once, as though the entire PolyLine were a single filled shape.

The first point is always relative to the drawable’s origin. The rest are relative either to that origin or the previous point, depending on the coordinate-mode.

When either of the two lines involved in a Bevel join is neither vertical nor horizontal, then the slope and position of the line segment defining the bevel join edge is implementation dependent. However, the computation of the slope and distance (relative to the join point) only depends on the line width and the slopes of the two lines.

GC components: function, plane-mask, line-width, line-style, cap-style, join-style, fill-style, sub-window-mode, clip-x-origin, clip-y-origin, clip-mask

GC mode-dependent components: foreground, background, tile, stipple, tile-stipple-x-origin, tile-stipple-y-origin, dash-offset, dashes

**PolySegment**

```
drawable: DRAWABLE
gc: GCONTEXT
segments: LISTofSEGMENT
```

where:

```
SEGMENT: [x1, y1, x2, y2: INT16]
```

Errors: Drawable, GContext, Match

For each segment, this request draws a line between [x1, y1] and [x2, y2]. The lines are drawn in the order listed. No joining is performed at coincident endpoints. For any given line, no pixel is drawn more than once. If lines intersect, the intersecting pixels are drawn multiple times.

GC components: function, plane-mask, line-width, line-style, cap-style, join-style, fill-style, sub-window-mode, clip-x-origin, clip-y-origin, clip-mask

GC mode-dependent components: foreground, background, tile, stipple, tile-stipple-x-origin, tile-stipple-y-origin, dash-offset, dashes

**PolyRectangle**

```
drawable: DRAWABLE
gc: GCONTEXT
rectangles: LISTofRECTANGLE
```

Errors: Drawable, GContext, Match

This request draws the outlines of the specified rectangles, as if a five-point PolyLine were specified for each rectangle:

```
[x,y] [x+width,y] [x+width,y+height] [x,y+height] [x,y]
```

The x and y coordinates of each rectangle are relative to the drawable’s origin and define the upper-left corner of the rectangle.
The rectangles are drawn in the order listed. For any given rectangle, no pixel is drawn more than once. If rectangles intersect, the intersecting pixels are drawn multiple times.

GC components: function, plane-mask, line-width, line-style, cap-style, join-style, fill-style, sub-window-mode, clip-x-origin, clip-y-origin, clip-mask

GC mode-dependent components: foreground, background, tile, stipple, tile-stipple-x-origin, tile-stipple-y-origin, dash-offset, dashes

PolyArc

drawable: DRAWABLE
gc: GCONTEXT
arcs: LISTofARC

Errors: Drawable, GContext, Match

This request draws circular or elliptical arcs. Each arc is specified by a rectangle and two angles. The angles are signed integers in degrees scaled by 64, with positive indicating counterclockwise motion and negative indicating clockwise motion. The start of the arc is specified by angle1 relative to the three-o’clock position from the center of the rectangle, and the path and extent of the arc is specified by angle2 relative to the start of the arc. If the magnitude of angle2 is greater than 360 degrees, it is truncated to 360 degrees. The x and y coordinates of the rectangle are relative to the origin of the drawable. For an arc specified as [x,y,w,h,a1,a2], the origin of the major and minor axes is at [x+(w/2),y+(h/2)], and the infinitely thin path describing the entire circle/ellipse intersects the horizontal axis at [x,y+(h/2)] and [x+w,y+(h/2)] and intersects the vertical axis at [x+(w/2),y] and [x+(w/2),y+h]. These coordinates are not necessarily integral; that is, they are not truncated to discrete coordinates.

For a wide line with line-width lw, the ideal bounding outlines for filling are given by the two infinitely thin paths consisting of all points whose perpendicular distance from a tangent to the path of the circle/ellipse is equal to lw/2 (which may be a fractional value). When the width and height of the arc are not equal and both are nonzero, then the actual bounding outlines are implementation dependent. However, the computation of the shape and position of the bounding outlines (relative to the center of the arc) only depends on the width and height of the arc and the line-width.

The cap-style is applied the same as for a line corresponding to the tangent of the circle/ellipse at the endpoint. When the angle of an arc face is not an integral multiple of 90 degrees, and the width and height of the arc are both are nonzero, then the shape and position of the cap at that face is implementation dependent. However, for a Butt cap, the face is defined by a straight line, and the computation of the position (relative to the center of the arc) and the slope of the line only depends on the width and height of the arc and the angle of the arc face. For other cap styles, the computation of the position (relative to the center of the arc) and the shape of the cap only depends on the width and height of the arc, the line-width, the angle of the arc face, and the direction (clockwise or counter clockwise) of the arc from the endpoint.

The join-style is applied the same as for two lines corresponding to the tangents of the circles/ellipses at the join point. When the width and height of both arcs are nonzero, and the angle of either arc face is not an integral multiple of 90 degrees, then the shape of the join is implementation dependent. However, the computation of the shape only depends on the width and height of each arc, the line-width, the angles of the two arc faces, the direction (clockwise or counter clockwise) of the arcs from the join point, and the relative orientation of the two arc center points.
For an arc specified as \([x, y, w, h, a1, a2]\), the angles must be specified in the effectively skewed coordinate system of the ellipse (for a circle, the angles and coordinate systems are identical). The relationship between these angles and angles expressed in the normal coordinate system of the screen (as measured with a protractor) is as follows:

\[
\text{skewed-angle} = \arctan(\tan(\text{normal-angle}) \times \frac{w}{h}) + \text{adjust}
\]

The skewed-angle and normal-angle are expressed in radians (rather than in degrees scaled by 64) in the range \([0, 2\pi]\). The \(\arctan\) returns a value in the range \([-\pi/2, \pi/2]\). The adjust is:

- 0 for normal-angle in the range \([0, \pi/2]\)
- \(\pi\) for normal-angle in the range \([\pi/2, (3\pi)/2]\)
- \(2\pi\) for normal-angle in the range \([(3\pi)/2, 2\pi]\)

The arcs are drawn in the order listed. If the last point in one arc coincides with the first point in the following arc, the two arcs will join correctly. If the first point in the first arc coincides with the last point in the last arc, the two arcs will join correctly. For any given arc, no pixel is drawn more than once. If two arcs join correctly and the line-width is greater than zero and the arcs intersect, no pixel is drawn more than once. Otherwise, the intersecting pixels of intersecting arcs are drawn multiple times. Specifying an arc with one endpoint and a clockwise extent draws the same pixels as specifying the other endpoint and an equivalent counterclockwise extent, except as it affects joins.

By specifying one axis to be zero, a horizontal or vertical line can be drawn.

Angles are computed based solely on the coordinate system, ignoring the aspect ratio.

GC components: function, plane-mask, line-width, line-style, cap-style, join-style, fill-style, sub-window-mode, clip-x-origin, clip-y-origin, clip-mask

GC mode-dependent components: foreground, background, tile, stipple, tile-stipple-x-origin, tile-stipple-y-origin, dash-offset, dashes

---

**FillPoly**

\[
\text{drawable}: \text{DRAWABLE} \\
\text{gc}: \text{GCONTEXT} \\
\text{shape}: \{ \text{Complex}, \text{Nonconvex}, \text{Convex} \} \\
\text{coordinate-mode}: \{ \text{Origin}, \text{Previous} \} \\
\text{points}: \text{LISTofPOINT}
\]

Errors: **Drawable**, **GContext**, **Match**, **Value**

This request fills the region closed by the specified path. The path is closed automatically if the last point in the list does not coincide with the first point. No pixel of the region is drawn more than once.

The first point is always relative to the drawable’s origin. The rest are relative either to that origin or the previous point, depending on the coordinate-mode.

The shape parameter may be used by the server to improve performance. **Complex** means the path may self-intersect. Contiguous coincident points in the path are not treated as self-intersection.

**Nonconvex** means the path does not self-intersect, but the shape is not wholly convex. If known by the client, specifying **Nonconvex** over **Complex** may improve performance. If **Nonconvex** is
specified for a self-intersecting path, the graphics results are undefined.

**Convex** means that for every pair of points inside the polygon, the line segment connecting them does not intersect the path. If known by the client, specifying **Convex** can improve performance. If **Convex** is specified for a path that is not convex, the graphics results are undefined.

GC components: function, plane-mask, fill-style, fill-rule, subwindow-mode, clip-x-origin, clip-y-origin, clip-mask

GC mode-dependent components: foreground, background, tile, stipple, tile-stipple-x-origin, tile-stipple-y-origin

---

**PolyFillRectangle**

- **drawable**: DRAWABLE
- **gc**: GCONTEXT
- **rectangles**: LISTofRECTANGLE

Errors: **Drawable**, **GContext**, **Match**

This request fills the specified rectangles, as if a four-point **FillPoly** were specified for each rectangle:

\[
[x,y] [x+width,y] [x+width,y+height] [x,y+height]
\]

The x and y coordinates of each rectangle are relative to the drawable’s origin and define the upper-left corner of the rectangle.

The rectangles are drawn in the order listed. For any given rectangle, no pixel is drawn more than once. If rectangles intersect, the intersecting pixels are drawn multiple times.

GC components: function, plane-mask, fill-style, subwindow-mode, clip-x-origin, clip-y-origin, clip-mask

GC mode-dependent components: foreground, background, tile, stipple, tile-stipple-x-origin, tile-stipple-y-origin

---

**PolyFillArc**

- **drawable**: DRAWABLE
- **gc**: GCONTEXT
- **arcs**: LISTofARC

Errors: **Drawable**, **GContext**, **Match**

For each arc, this request fills the region closed by the infinitely thin path described by the specified arc and one or two line segments, depending on the arc-mode. For **Chord**, the single line segment joining the endpoints of the arc is used. For **PieSlice**, the two line segments joining the endpoints of the arc with the center point are used.

For an arc specified as \([x,y,w,h,a1,a2]\), the origin of the major and minor axes is at \([x+(w/2),y+(h/2)]\), and the infinitely thin path describing the entire circle/ellipse intersects the horizontal axis at \([x,y+(h/2)]\) and \([x+w,y+(h/2)]\) and intersects the vertical axis at \([x+(w/2),y]\) and \([x+(w/2),y+h]\). These coordinates are not necessarily integral; that is, they are not truncated to discrete coordinates.
The arc angles are interpreted as specified in the **PolyArc** request. When the angle of an arc face is not an integral multiple of 90 degrees, then the precise endpoint on the arc is implementation dependent. However, for **Chord** arc-mode, the computation of the pair of endpoints (relative to the center of the arc) only depends on the width and height of the arc and the angles of the two arc faces. For **PieSlice** arc-mode, the computation of an endpoint only depends on the angle of the arc face for that endpoint and the ratio of the arc width to arc height.

The arcs are filled in the order listed. For any given arc, no pixel is drawn more than once. If regions intersect, the intersecting pixels are drawn multiple times.

**GC components:** function, plane-mask, fill-style, arc-mode, subwindow-mode, clip-x-origin, clip-y-origin, clip-mask

**GC mode-dependent components:** foreground, background, tile, stipple, tile-stipple-x-origin, tile-stipple-y-origin

---

**PutImage**

```
drawable: DRAWABLE
gc: GCONTEXT
depth: CARD8
width, height: CARD16
dst-x, dst-y: INT16
left-pad: CARD8
format: { Bitmap, XYPixmap, ZPixmap }
data: LISTofBYTE
```

**Errors:** Drawable, GContext, Match, Value

This request combines an image with a rectangle of the drawable. The dst-x and dst-y coordinates are relative to the drawable’s origin.

If **Bitmap** format is used, then depth must be one (or a **Match** error results), and the image must be in XY format. The foreground pixel in gc defines the source for bits set to 1 in the image, and the background pixel defines the source for the bits set to 0.

For **XYPixmap** and **ZPixmap**, the depth must match the depth of the drawable (or a **Match** error results). For **XYPixmap**, the image must be sent in XY format. For **ZPixmap**, the image must be sent in the Z format defined for the given depth.

The left-pad must be zero for **ZPixmap** format (or a **Match** error results). For **Bitmap** and **XYPixmap** format, left-pad must be less than bitmap-scanline-pad as given in the server connection setup information (or a **Match** error results). 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 width argument defines the width of the actual image and does not include left-pad.

**GC components:** function, plane-mask, subwindow-mode, clip-x-origin, clip-y-origin, clip-mask

**GC mode-dependent components:** foreground, background
GetImage

drawable: DRAWABLE
x, y: INT16
width, height: CARD16
plane-mask: CARD32
format: { XYPixmap, ZPixmap }

→
depth: CARD8
visual: VISUALID or None
data: LISTofBYTE

Errors: Drawable, Match, Value

This request returns the contents of the given rectangle of the drawable in the given format. The x and y coordinates are relative to the drawable’s origin and define the upper-left corner of the rectangle. If XYPixmap is specified, only the bit planes specified in plane-mask are transmitted, with the planes appearing from most significant to least significant in bit order. If ZPixmap is specified, then bits in all planes not specified in plane-mask are transmitted as zero. Range checking is not performed on plane-mask; extraneous bits are simply ignored. The returned depth is as specified when the drawable was created and is the same as a depth component in a FORMAT structure (in the connection setup), not a bits-per-pixel component. If the drawable is a window, its visual type is returned. If the drawable is a pixmap, the visual is None.

If the drawable is a pixmap, then the given rectangle must be wholly contained within the pixmap (or a Match error results). If the drawable is a window, the window must be viewable, and it must be the case that, if there were no inferiors or overlapping windows, the specified rectangle of the window would be fully visible on the screen and wholly contained within the outside edges of the window (or a Match error results). Note that the borders of the window can be included and read with this request. If the window has a backing store, then the backing-store contents are returned for regions of the window that are obscured by noninferior windows; otherwise, the returned contents of such obscured regions are undefined. Also undefined are the returned contents of visible regions of inferiors of different depth than the specified window. The pointer cursor image is not included in the contents returned.

This request is not general-purpose in the same sense as other graphics-related requests. It is intended specifically for rudimentary hardcopy support.
PolyText8

`drawable`: DRAWABLE
`gc`: GCONTEXT
`x, y`: INT16
`items`: LISTofTEXTITEM8

where:

TEXTITEM8: TEXTELT8 or FONT
TEXTELT8: [delta: INT8
            string: STRING8]

Errors: Drawable, Font, GContext, Match

The x and y coordinates are relative to the drawable’s origin and specify the baseline starting position (the initial character origin). Each text item is processed in turn. A font item causes the font to be stored in gc and to be used for subsequent text. Switching among fonts does not affect the next character origin. A text element delta specifies an additional change in the position along the x axis before the string is drawn; the delta is always added to the character origin. Each character image, as defined by the font in gc, is treated as an additional mask for a fill operation on the drawable.

All contained FONTs are always transmitted most significant byte first.

If a Font error is generated for an item, the previous items may have been drawn.

For fonts defined with 2-byte matrix indexing, each STRING8 byte is interpreted as a byte2 value of a CHAR2B with a byte1 value of zero.

GC components: function, plane-mask, fill-style, font, subwindow-mode, clip-x-origin, clip-y-origin, clip-mask

GC mode-dependent components: foreground, background, tile, stipple, tile-stipple-x-origin, tile-stipple-y-origin

PolyText16

`drawable`: DRAWABLE
`gc`: GCONTEXT
`x, y`: INT16
`items`: LISTofTEXTITEM16

where:

TEXTITEM16: TEXTELT16 or FONT
TEXTELT16: [delta: INT8
            string: STRING16]

Errors: Drawable, Font, GContext, Match

This request is similar to PolyText8, except 2-byte (or 16-bit) characters are used. For fonts defined with linear indexing rather than 2-byte matrix indexing, the server will interpret each CHAR2B as a 16-bit number that has been transmitted most significant byte first (that is, byte1 of the CHAR2B is taken as the most significant byte).
ImageText8

drawable: DRAWWABLE
gc: GCONTEXT
x, y: INT16
string: STRING8

Errors: Drawable, GContext, Match

The x and y coordinates are relative to the drawable’s origin and specify the baseline starting position (the initial character origin). The effect is first to fill a destination rectangle with the background pixel defined in gc and then to paint the text with the foreground pixel. The upper-left corner of the filled rectangle is at:

[x, y - font-ascent]

the width is:

overall-width

and the height is:

font-ascent + font-descent

The overall-width, font-ascent, and font-descent are as they would be returned by a QueryTextExtents call using gc and string.

The function and fill-style defined in gc are ignored for this request. The effective function is Copy, and the effective fill-style Solid.

For fonts defined with 2-byte matrix indexing, each STRING8 byte is interpreted as a byte2 value of a CHAR2B with a byte1 value of zero.

GC components: plane-mask, foreground, background, font, subwindow-mode, clip-x-origin, clip-y-origin, clip-mask

ImageText16

drawable: DRAWWABLE
gc: GCONTEXT
x, y: INT16
string: STRING16

Errors: Drawable, GContext, Match

This request is similar to ImageText8, except 2-byte (or 16-bit) characters are used. For fonts defined with linear indexing rather than 2-byte matrix indexing, the server will interpret each CHAR2B as a 16-bit number that has been transmitted most significant byte first (that is, byte1 of the CHAR2B is taken as the most significant byte).
CreateColormap

mid: COLORMAP
visual: VISUALID
window: WINDOW
alloc: { None, All }

Errors: Alloc, IDChoice, Match, Value, Window

This request creates a colormap of the specified visual type for the screen on which the window resides and associates the identifier mid with it. The visual type must be one supported by the screen (or a Match error results). The initial values of the colormap entries are undefined for classes GrayScale, PseudoColor, and DirectColor. For StaticGray, StaticColor, and TrueColor, the entries will have defined values, but those values are specific to the visual and are not defined by the core protocol. For StaticGray, StaticColor, and TrueColor, alloc must be specified as None (or a Match error results). For the other classes, if alloc is None, the colormap initially has no allocated entries, and clients can allocate entries.

If alloc is All, then the entire colormap is allocated writable. The initial values of all allocated entries are undefined. For GrayScale and PseudoColor, the effect is as if an AllocColorCells request returned all pixel values from zero to N – 1, where N is the colormap-entries value in the specified visual. For DirectColor, the effect is as if an AllocColorPlanes request returned a pixel value of zero and red-mask, green-mask, and blue-mask values containing the same bits as the corresponding masks in the specified visual. However, in all cases, none of these entries can be freed with FreeColors.

FreeColormap
cmap: COLORMAP

Errors: Colormap

This request deletes the association between the resource ID and the colormap and frees the colormap storage. If the colormap is an installed map for a screen, it is uninstalled (see UninstallColormap request). If the colormap is defined as the colormap for a window (by means of CreateWindow or ChangeWindowAttributes), the colormap for the window is changed to None, and a ColormapNotify event is generated. The protocol does not define the colors displayed for a window with a colormap of None.

This request has no effect on a default colormap for a screen.

CopyColormapAndFree

mid, src-cmap: COLORMAP

Errors: Alloc, Colormap, IDChoice

This request creates a colormap of the same visual type and for the same screen as src-cmap, and it associates identifier mid with it. It also moves all of the client’s existing allocations from src-cmap to the new colormap with their color values intact and their read-only or writable characteristics intact, and it frees those entries in src-cmap. Color values in other entries in the new colormap are undefined. If src-cmap was created by the client with alloc All (see CreateColormap
request), then the new colormap is also created with alloc All, all color values for all entries are copied from src-cmap, and then all entries in src-cmap are freed. If src-cmap was not created by the client with alloc All, then the allocations to be moved are all those pixels and planes that have been allocated by the client using either AllocColor, AllocNamedColor, AllocColorCells, or AllocColorPlanes and that have not been freed since they were allocated.

InstallColormap
cmap: COLORMAP

Errors: Colormap

This request makes this colormap an installed map for its screen. All windows associated with this colormap immediately display with true colors. As a side effect, additional colormaps might be implicitly installed or uninstalled by the server. Which other colormaps get installed or uninstalled is server-dependent except that the required list must remain installed.

If cmap is not already an installed map, a ColormapNotify event is generated on every window having cmap as an attribute. In addition, for every other colormap that is installed or uninstalled as a result of the request, a ColormapNotify event is generated on every window having that colormap as an attribute.

At any time, there is a subset of the installed maps that are viewed as an ordered list and are called the required list. The length of the required list is at most M, where M is the min-installed-maps specified for the screen in the connection setup. The required list is maintained as follows. When a colormap is an explicit argument to InstallColormap, it is added to the head of the list; the list is truncated at the tail, if necessary, to keep the length of the list to at most M. When a colormap is an explicit argument to UninstallColormap and it is in the required list, it is removed from the list. A colormap is not added to the required list when it is installed implicitly by the server, and the server cannot implicitly uninstall a colormap that is in the required list.

Initially the default colormap for a screen is installed (but is not in the required list).

UninstallColormap
cmap: COLORMAP

Errors: Colormap

If cmap is on the required list for its screen (see InstallColormap request), it is removed from the list. As a side effect, cmap might be uninstalled, and additional colormaps might be implicitly installed or uninstalled. Which colormaps get installed or uninstalled is server-dependent except that the required list must remain installed.

If cmap becomes uninstalled, a ColormapNotify event is generated on every window having cmap as an attribute. In addition, for every other colormap that is installed or uninstalled as a result of the request, a ColormapNotify event is generated on every window having that colormap as an attribute.
**ListInstalledColormaps**

\[
\text{\textit{window}: \text{WINDOW}}
\]

\[
\rightarrow
\]

\[
\text{\textit{cmaps}: \text{LISTofCOLORMAP}}
\]

**Errors:** \text{Window}

This request returns a list of the currently installed colormaps for the screen of the specified window. The order of colormaps is not significant, and there is no explicit indication of the required list (see \text{InstallColormap} request).

**AllocColor**

\[
\text{\textit{cmap}: \text{COLORMAP}}
\]

\[
\text{\textit{red}, \textit{green}, \textit{blue}: \text{CARD16}}
\]

\[
\rightarrow
\]

\[
\text{\textit{pixel}: \text{CARD32}}
\]

\[
\text{\textit{red}, \textit{green}, \textit{blue}: \text{CARD16}}
\]

**Errors:** \text{Alloc, Colormap}

This request allocates a read-only colormap entry corresponding to the closest RGB values provided by the hardware. It also returns the pixel and the RGB values actually used. Multiple clients requesting the same effective RGB values can be assigned the same read-only entry, allowing entries to be shared.

**AllocNamedColor**

\[
\text{\textit{cmap}: \text{COLORMAP}}
\]

\[
\text{\textit{name}: \text{STRING8}}
\]

\[
\rightarrow
\]

\[
\text{\textit{pixel}: \text{CARD32}}
\]

\[
\text{\textit{exact-red}, \textit{exact-green}, \textit{exact-blue}: \text{CARD16}}
\]

\[
\text{\textit{visual-red}, \textit{visual-green}, \textit{visual-blue}: \text{CARD16}}
\]

**Errors:** \text{Alloc, Colormap, Name}

This request looks up the named color with respect to the screen associated with the colormap. Then, it does an \text{AllocColor} on \textit{cmap}. The name should use the ISO Latin-1 encoding, and uppercase and lowercase do not matter. The exact RGB values specify the true values for the color, and the visual values specify the values actually used in the colormap.
AllocColorCells

cmap: COLORMAP
colors, planes: CARD16
contiguous: BOOL

→

pixels, masks: LISTofCARD32

Errors: Alloc, Colormap, Value

The number of colors must be positive, and the number of planes must be nonnegative (or a Value error results). If C colors and P planes are requested, then C pixels and P masks are returned. No mask will have any bits in common with any other mask or with any of the pixels. By ORing together masks and pixels, C*2^P distinct pixels can be produced; all of these are allocated writable by the request. For GrayScale or PseudoColor, each mask will have exactly one bit set to 1; for DirectColor, each will have exactly three bits set to 1. If contiguous is True and if all masks are ORed together, a single contiguous set of bits will be formed for GrayScale or PseudoColor, and three contiguous sets of bits (one within each pixel subfield) for DirectColor. The RGB values of the allocated entries are undefined.

AllocColorPlanes

cmap: COLORMAP
colors, reds, greens, blues: CARD16
contiguous: BOOL

→

pixels: LISTofCARD32
red-mask, green-mask, blue-mask: CARD32

Errors: Alloc, Colormap, Value

The number of colors must be positive, and the reds, greens, and blues must be nonnegative (or a Value error results). If C colors, R reds, G greens, and B blues are requested, then C pixels are returned, and the masks have R, G, and B bits set, respectively. If contiguous is True, then each mask will have a contiguous set of bits. No mask will have any bits in common with any other mask or with any of the pixels. For DirectColor, each mask will lie within the corresponding pixel subfield. By ORing together subsets of masks with pixels, C*2^{R+G+B} distinct pixels can be produced; all of these are allocated writable by the request. The initial RGB values of the allocated entries are undefined. In the colormap, there are only C*2^R independent red entries, C*2^G independent green entries, and C*2^B independent blue entries. This is true even for PseudoColor. When the colormap entry for a pixel value is changed using StoreColors or StoreNamedColor, the pixel is decomposed according to the masks and the corresponding independent entries are updated.
FreeColors

cmap: COLORMAP
pixels: LISTofCARD32
plane-mask: CARD32

Errors: Access, Colormap, Value

The plane-mask should not have any bits in common with any of the pixels. The set of all pixels is produced by ORing together subsets of plane-mask with the pixels. The request frees all of these pixels that were allocated by the client (using AllocColor, AllocNamedColor, AllocColorCells, and AllocColorPlanes). Note that freeing an individual pixel obtained from AllocColorPlanes may not actually allow it to be reused until all of its related pixels are also freed. Similarly, a read-only entry is not actually freed until it has been freed by all clients, and if a client allocates the same read-only entry multiple times, it must free the entry that many times before the entry is actually freed.

All specified pixels that are allocated by the client in cmap are freed, even if one or more pixels produce an error. A Value error is generated if a specified pixel is not a valid index into cmap. An Access error is generated if a specified pixel is not allocated by the client (that is, is unallocated or is only allocated by another client) or if the colormap was created with all entries writable (using an alloc value of All in CreateColormap). If more than one pixel is in error, it is arbitrary as to which pixel is reported.

StoreColors

cmap: COLORMAP
items: LISTofCOLORITEM

where:

COLORITEM: [pixel: CARD32
do-red, do-green, do-blue: BOOL
red, green, blue: CARD16]

Errors: Access, Colormap, Value

This request changes the colormap entries of the specified pixels. The do-red, do-green, and do-blue fields indicate which components should actually be changed. If the colormap is an installed map for its screen, the changes are visible immediately.

All specified pixels that are allocated writable in cmap (by any client) are changed, even if one or more pixels produce an error. A Value error is generated if a specified pixel is not a valid index into cmap, and an Access error is generated if a specified pixel is unallocated or is allocated read-only. If more than one pixel is in error, it is arbitrary as to which pixel is reported.
StoreNamedColor

cmap: COLORMAP
pixel: CARD32
name: STRING8
do-red, do-green, do-blue: BOOL

Errors: Access, Colormap, Name, Value

This request looks up the named color with respect to the screen associated with cmap and then does a StoreColors in cmap. The name should use the ISO Latin-1 encoding, and uppercase and lowercase do not matter. The Access and Value errors are the same as in StoreColors.

QueryColors

cmap: COLORMAP
pixels: LISTofCARD32
→
colors: LISTofRGB
where:
RGB: [red, green, blue: CARD16]

Errors: Colormap, Value

This request returns the hardware-specific color values stored in cmap for the specified pixels. The values returned for an unallocated entry are undefined. A Value error is generated if a pixel is not a valid index into cmap. If more than one pixel is in error, it is arbitrary as to which pixel is reported.

LookupColor

cmap: COLORMAP
name: STRING8
→
exact-red, exact-green, exact-blue: CARD16
visual-red, visual-green, visual-blue: CARD16

Errors: Colormap, Name

This request looks up the string name of a color with respect to the screen associated with cmap and returns both the exact color values and the closest values provided by the hardware with respect to the visual type of cmap. The name should use the ISO Latin-1 encoding, and uppercase and lowercase do not matter.
CreateCursor

\[cid: CURSOR\]
\[source: PIXMAP\]
\[mask: PIXMAP or None\]
\[fore-red, fore-green, fore-blue: CARD16\]
\[back-red, back-green, back-blue: CARD16\]
\[x, y: CARD16\]

**Errors:** Alloc, IDChoice, Match, Pixmap

This request creates a cursor and associates identifier \(cid\) with it. The foreground and background RGB values must be specified, even if the server only has a StaticGray or GrayScale screen. The foreground is used for the bits set to 1 in the source, and the background is used for the bits set to 0. Both source and mask (if specified) must have depth one (or a Match error results), but they can have any root. The mask pixmap defines the shape of the cursor. That is, the bits set to 1 in the mask define which source pixels will be displayed, and where the mask has bits set to 0, the corresponding bits of the source pixmap are ignored. If no mask is given, all pixels of the source are displayed. The mask, if present, must be the same size as the source (or a Match error results). The \(x\) and \(y\) coordinates define the hotspot relative to the source’s origin and must be a point within the source (or a Match error results).

The components of the cursor may be transformed arbitrarily to meet display limitations.

The pixmaps can be freed immediately if no further explicit references to them are to be made. Subsequent drawing in the source or mask pixmap has an undefined effect on the cursor. The server might or might not make a copy of the pixmap.

CreateGlyphCursor

\[cid: CURSOR\]
\[source-font: FONT\]
\[mask-font: FONT or None\]
\[source-char, mask-char: CARD16\]
\[fore-red, fore-green, fore-blue: CARD16\]
\[back-red, back-green, back-blue: CARD16\]

**Errors:** Alloc, Font, IDChoice, Value

This request is similar to CreateCursor, except the source and mask bitmaps are obtained from the specified font glyphs. The source-char must be a defined glyph in source-font, and if mask-font is given, mask-char must be a defined glyph in mask-font (or a Value error results). The mask font and character are optional. The origins of the source and mask (if it is defined) glyphs are positioned coincidently and define the hotspot. The source and mask need not have the same bounding box metrics, and there is no restriction on the placement of the hotspot relative to the bounding boxes. If no mask is given, all pixels of the source are displayed. Note that source-char and mask-char are CARD16, not CHAR2B. For 2-byte matrix fonts, the 16-bit value should be formed with byte1 in the most significant byte and byte2 in the least significant byte.

The components of the cursor may be transformed arbitrarily to meet display limitations.

The fonts can be freed immediately if no further explicit references to them are to be made.
FreeCursor

cursor: CURSOR

Errors: Cursor

This request deletes the association between the resource ID and the cursor. The cursor storage will be freed when no other resource references it.

RecolorCursor

cursor: CURSOR
text-red, text-green, text-blue: CARD16

Errors: Cursor

This request changes the color of a cursor. If the cursor is being displayed on a screen, the change is visible immediately.

QueryBestSize

class: {Cursor, Tile, Stipple}
drawable: DRAWABLE

width, height: CARD16

→

width, height: CARD16

Errors: Drawable, Match, Value

This request returns the best size that is closest to the argument size. For Cursor, this is the largest size that can be fully displayed. For Tile, this is the size that can be tiled fastest. For Stipple, this is the size that can be stippled fastest.

For Cursor, the drawable indicates the desired screen. For Tile and Stipple, the drawable indicates the screen and also possibly the window class and depth. An InputOnly window cannot be used as the drawable for Tile or Stipple (or a Match error results).

QueryExtension

name: STRING8

→

present: BOOL
major-opcode: CARD8
first-event: CARD8
first-error: CARD8

This request determines if the named extension is present. If so, the major opcode for the extension is returned, if it has one. Otherwise, zero is returned. Any minor opcode and the request formats are specific to the extension. If the extension involves additional event types, the base event
type code is returned. Otherwise, zero is returned. The format of the events is specific to the
extension. If the extension involves additional error codes, the base error code is returned. Other-
wise, zero is returned. The format of additional data in the errors is specific to the extension.
The extension name should use the ISO Latin-1 encoding, and uppercase and lowercase matter.

ListExtensions
→
  names: LISTofSTRING8

This request returns a list of all extensions supported by the server.

SetModifierMapping
  keycodes-per-modifier: CARD8
  keycodes: LISTofKEYCODE
→
  status: {Success, Busy, Failed}
  Errors: Alloc, Value

This request specifies the keycodes (if any) of the keys to be used as modifiers. The number of
keycodes in the list must be 8*keycodes-per-modifier (or a Length error results). The keycodes
are divided into eight sets, with each set containing keycodes-per-modifier elements. The sets are
assigned to the modifiers Shift, Lock, Control, Mod1, Mod2, Mod3, Mod4, and Mod5, in
order. Only nonzero keycode values are used within each set; zero values are ignored. All of the
nonzero keycodes must be in the range specified by min-keycode and max-keycode in the connec-
tion setup (or a Value error results). The order of keycodes within a set does not matter. If no
nonzero values are specified in a set, the use of the corresponding modifier is disabled, and the
modifier bit will always be zero. Otherwise, the modifier bit will be one whenever at least one of
the keys in the corresponding set is in the down position.

A server can impose restrictions on how modifiers can be changed (for example, if certain keys do
not generate up transitions in hardware, if auto-repeat cannot be disabled on certain keys, or if
multiple keys per modifier are not supported). The status reply is Failed if some such restriction
is violated, and none of the modifiers is changed.

If the new nonzero keycodes specified for a modifier differ from those currently defined and any
(current or new) keys for that modifier are logically in the down state, then the status reply is
Busy, and none of the modifiers is changed.

This request generates a MappingNotify event on a Success status.

GetModifierMapping
→
  keycodes-per-modifier: CARD8
  keycodes: LISTofKEYCODE
This request returns the keycodes of the keys being used as modifiers. The number of keycodes in the list is 8*keycodes-per-modifier. The keycodes are divided into eight sets, with each set containing keycodes-per-modifier elements. The sets are assigned to the modifiers Shift, Lock, Control, Mod1, Mod2, Mod3, Mod4, and Mod5, in order. The keycodes-per-modifier value is chosen arbitrarily by the server; zeroes are used to fill in unused elements within each set. If only zero values are given in a set, the use of the corresponding modifier has been disabled. The order of keycodes within each set is chosen arbitrarily by the server.

**ChangeKeyboardMapping**

first-keycode : KEYCODE
delsyms-per-keycode : CARD8
delsyms : LISTofKEYSYM

Errors: Alloc, Value

This request defines the symbols for the specified number of keycodes, starting with the specified keycode. The symbols for keycodes outside this range remained unchanged. The number of elements in the delsyms list must be a multiple of delsyms-per-keycode (or a Length error results). The first-keycode must be greater than or equal to min-keycode as returned in the connection setup (or a Value error results) and:

\[
\text{first-keycode} + \left( \frac{\text{delsyms-length}}{\text{delsyms-per-keycode}} \right) - 1
\]

must be less than or equal to max-keycode as returned in the connection setup (or a Value error results). KEYSYM number N (counting from zero) for keycode K has an index (counting from zero) of:

\[
(K - \text{first-keycode}) \times \text{delsyms-per-keycode} + N
\]

in delsyms. The delsyms-per-keycode can be chosen arbitrarily by the client to be large enough to hold all desired symbols. A special KEYSYM value of NoSymbol should be used to fill in unused elements for individual keycodes. It is legal for NoSymbol to appear in nontrailing positions of the effective list for a keycode.

This request generates a MappingNotify event.

There is no requirement that the server interpret this mapping; it is merely stored for reading and writing by clients (see section 5).

**GetKeyboardMapping**

first-keycode : KEYCODE
count : CARD8

\[
\rightarrow
\text{delsyms-per-keycode} : \text{CARD8}
delsyms : \text{LISTofKEYSYM}
\]

Errors: Value

This request returns the symbols for the specified number of keycodes, starting with the specified keycode. The first-keycode must be greater than or equal to min-keycode as returned in the
connection setup (or a Value error results), and:

\[ \text{first-keycode + count} - 1 \]

must be less than or equal to max-keycode as returned in the connection setup (or a Value error results). The number of elements in the keysyms list is:

\[ \text{count} \times \text{keysyms-per-keycode} \]

and KEYSYM number \( N \) (counting from zero) for keycode \( K \) has an index (counting from zero) of:

\[ (K - \text{first-keycode}) \times \text{keysyms-per-keycode} + N \]

in keysyms. The keysyms-per-keycode value is chosen arbitrarily by the server to be large enough to report all requested symbols. A special KEYSYM value of NoSymbol is used to fill in unused elements for individual keycodes.

---

**ChangeKeyboardControl**

\[ \text{value-mask}: \text{BITMASK} \]

\[ \text{value-list}: \text{LISTofVALUE} \]

Errors: Match, Value

This request controls various aspects of the keyboard. The value-mask and value-list specify which controls are to be changed. The possible values are:

<table>
<thead>
<tr>
<th>Control</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>key-click-percent</td>
<td>INT8</td>
</tr>
<tr>
<td>bell-percent</td>
<td>INT8</td>
</tr>
<tr>
<td>bell-pitch</td>
<td>INT16</td>
</tr>
<tr>
<td>bell-duration</td>
<td>INT16</td>
</tr>
<tr>
<td>led</td>
<td>CARD8</td>
</tr>
<tr>
<td>led-mode</td>
<td>{ On, Off }</td>
</tr>
<tr>
<td>key</td>
<td>KEYCODE</td>
</tr>
<tr>
<td>auto-repeat-mode</td>
<td>{ On, Off, Default }</td>
</tr>
</tbody>
</table>

The key-click-percent sets the volume for key clicks between 0 (off) and 100 (loud) inclusive, if possible. Setting to \(-1\) restores the default. Other negative values generate a Value error.

The bell-percent sets the base volume for the bell between 0 (off) and 100 (loud) inclusive, if possible. Setting to \(-1\) restores the default. Other negative values generate a Value error.

The bell-pitch sets the pitch (specified in Hz) of the bell, if possible. Setting to \(-1\) restores the default. Other negative values generate a Value error.

The bell-duration sets the duration of the bell (specified in milliseconds), if possible. Setting to \(-1\) restores the default. Other negative values generate a Value error.

If both led-mode and led are specified, then the state of that LED is changed, if possible. If only led-mode is specified, then the state of all LEDs are changed, if possible. At most 32 LEDs, numbered from one, are supported. No standard interpretation of LEDs is defined. It is a Match
error if an led is specified without an led-mode.

If both auto-repeat-mode and key are specified, then the auto-repeat mode of that key is changed, if possible. If only auto-repeat-mode is specified, then the global auto-repeat mode for the entire keyboard is changed, if possible, without affecting the per-key settings. It is a Match error if a key is specified without an auto-repeat-mode. Each key has an individual mode of whether or not it should auto-repeat and a default setting for that mode. In addition, there is a global mode of whether auto-repeat should be enabled or not and a default setting for that mode. When the global mode is On, keys should obey their individual auto-repeat modes. When the global mode is Off, no keys should auto-repeat. An auto-repeating key generates alternating KeyPress and KeyRelease events. When a key is used as a modifier, it is desirable for the key not to auto-repeat, regardless of the auto-repeat setting for that key.

A bell generator connected with the console but not directly on the keyboard is treated as if it were part of the keyboard.

The order in which controls are verified and altered is server-dependent. If an error is generated, a subset of the controls may have been altered.

GetKeyboardControl

\[
\text{GetKeyboardControl} \\
\rightarrow \\
\text{key-click-percent: CARD8} \;
\text{bell-percent: CARD8} \\
\text{bell-pitch: CARD16} \;
\text{bell-duration: CARD16} \\
\text{led-mask: CARD32} \;
\text{global-auto-repeat: \{On, Off\}} \\
\text{auto-repeats: LISTofCARD8}
\]

This request returns the current control values for the keyboard. For the LEDs, the least significant bit of led-mask corresponds to LED one, and each one bit in led-mask indicates an LED that is lit. The auto-repeats is a bit vector; each one bit indicates that auto-repeat is enabled for the corresponding key. The vector is represented as 32 bytes. Byte N (from 0) contains the bits for keys 8N to 8N + 7, with the least significant bit in the byte representing key 8N.

Bell

\[
\text{percent: INT8}
\]

Errors: Value

This request rings the bell on the keyboard at a volume relative to the base volume for the keyboard, if possible. Percent can range from –100 to 100 inclusive (or a Value error results). The volume at which the bell is rung when percent is nonnegative is:

\[
\text{base} - \lfloor(\text{base} \times \text{percent}) / 100\rfloor + \text{percent}
\]

When percent is negative, it is:

\[
\text{base} + \lfloor(\text{base} \times \text{percent}) / 100\rfloor
\]
SetPointerMapping

\[ \text{map}: \text{LISTofCARD8} \]

\[ \rightarrow \]

\[ \text{status}: \{ \text{Success}, \text{Busy} \} \]

Errors: Value

This request sets the mapping of the pointer. Elements of the list are indexed starting from one. The length of the list must be the same as GetPointerMapping would return (or a Value error results). The index is a core button number, and the element of the list defines the effective number.

A zero element disables a button. Elements are not restricted in value by the number of physical buttons, but no two elements can have the same nonzero value (or a Value error results).

If any of the buttons to be altered are logically in the down state, the status reply is Busy, and the mapping is not changed.

This request generates a MappingNotify event on a Success status.

GetPointerMapping

\[ \rightarrow \]

\[ \text{map}: \text{LISTofCARD8} \]

This request returns the current mapping of the pointer. Elements of the list are indexed starting from one. The length of the list indicates the number of physical buttons.

The nominal mapping for a pointer is the identity mapping: map[i]=i.

ChangePointerControl

\[ \text{do-acceleration}, \text{do-threshold}: \text{BOOL} \]

\[ \text{acceleration-numerator}, \text{acceleration-denominator}: \text{INT16} \]

\[ \text{threshold}: \text{INT16} \]

Errors: Value

This request defines how the pointer moves. The acceleration is a multiplier for movement expressed as a fraction. For example, specifying 3/1 means the pointer moves three times as fast as normal. The fraction can be rounded arbitrarily by the server. Acceleration only takes effect if the pointer moves more than threshold number of pixels at once and only applies to the amount beyond the threshold. Setting a value to −1 restores the default. Other negative values generate a Value error, as does a zero value for acceleration-denominator.
GetPointerControl

acceleration-numerator, acceleration-denominator: CARD16
threshold: CARD16

This request returns the current acceleration and threshold for the pointer.

SetScreenSaver

timeout, interval: INT16
prefer-blanking: { Yes, No, Default }
allow-exposures: { Yes, No, Default }

Errors: Value

The timeout and interval are specified in seconds; setting a value to –1 restores the default. Other negative values generate a Value error. If the timeout value is zero, screen-saver is disabled (but an activated screen-saver is not deactivated). If the timeout value is nonzero, screen-saver is enabled. Once screen-saver is enabled, if no input from the keyboard or pointer is generated for timeout seconds, screen-saver is activated. For each screen, if blanking is preferred and the hardware supports video blanking, the screen will simply go blank. Otherwise, if either exposures are allowed or the screen can be regenerated without sending exposure events to clients, the screen is changed in a server-dependent fashion to avoid phosphor burn. Otherwise, the state of the screens does not change, and screen-saver is not activated. At the next keyboard or pointer input or at the next ForceScreenSaver with mode Reset, screen-saver is deactivated, and all screen states are restored.

If the server-dependent screen-saver method is amenable to periodic change, interval serves as a hint about how long the change period should be, with zero hinting that no periodic change should be made. Examples of ways to change the screen include scrambling the color map periodically, moving an icon image about the screen periodically, or tiling the screen with the root window background tile, randomly reorigined periodically.

GetScreenSaver

timeout, interval: CARD16
prefer-blanking: { Yes, No } 
allow-exposures: { Yes, No } 

This request returns the current screen-saver control values.

ForceScreenSaver

mode: { Activate, Reset } 

Errors: Value
If the mode is **Activate** and screen-saver is currently deactivated, then screen-saver is activated (even if screen-saver has been disabled with a timeout value of zero). If the mode is **Reset** and screen-saver is currently enabled, then screen-saver is deactivated (if it was activated), and the activation timer is reset to its initial state as if device input had just been received.

### ChangeHosts

- **mode:** \{ Insert, Delete \}
- **host:** HOST

**Errors:** Access, Value

This request adds or removes the specified host from the access control list. When the access control mechanism is enabled and a client attempts to establish a connection to the server, the host on which the client resides must be in the access control list, or the client must have been granted permission by a server-dependent method, or the server will refuse the connection.

The client must reside on the same host as the server and/or have been granted permission by a server-dependent method to execute this request (or an Access error results).

An initial access control list can usually be specified, typically by naming a file that the server reads at startup and reset.

The following address families are defined. A server is not required to support these families and may support families not listed here. Use of an unsupported family, an improper address format, or an improper address length within a supported family results in a Value error.

For the Internet family, the address must be four bytes long. The address bytes are in standard IP order; the server performs no automatic swapping on the address bytes. The Internet family supports IP version 4 addresses only.

For the InternetV6 family, the address must be sixteen bytes long. The address bytes are in standard IP order; the server performs no automatic swapping on the address bytes. The InternetV6 family supports IP version 6 addresses only.

For the DECnet family, the server performs no automatic swapping on the address bytes. A Phase IV address is two bytes long: the first byte contains the least significant eight bits of the node number, and the second byte contains the most significant two bits of the node number in the least significant two bits of the byte and the area in the most significant six bits of the byte.

For the Chaos family, the address must be two bytes long. The host number is always the first byte in the address, and the subnet number is always the second byte. The server performs no automatic swapping on the address bytes.

For the ServerInterpreted family, the address may be of any length up to 65535 bytes. The address consists of two strings of ASCII characters, separated by a byte with a value of 0. The first string represents the type of address, and the second string contains the address value. Address types and the syntax for their associated values will be registered via the X.Org Registry. Implementors who wish to add implementation specific types may register a unique prefix with the X.Org registry to prevent namespace collisions.

Use of a host address in the ChangeHosts request is deprecated. It is only useful when a host has a unique, constant address, a requirement that is increasingly unmet as sites adopt dynamically assigned addresses, network address translation gateways, IPv6 link local addresses, and various other technologies. It also assumes all users of a host share equivalent access rights, and as such has never been suitable for many multi-user machine environments. Instead, more secure forms
of authentication, such as those based on shared secrets or public key encryption, are recom-
mended.

**ListHosts**

```
→
  mode: {Enabled, Disabled}
  hosts: LISTofHOST
```

This request returns the hosts on the access control list and whether use of the list at connection setup is currently enabled or disabled.

Each HOST is padded to a multiple of four bytes.

**SetAccessControl**

```
mode: {Enable, Disable}
```

Errors: Access, Value

This request enables or disables the use of the access control list at connection setups.

The client must reside on the same host as the server and/or have been granted permission by a server-dependent method to execute this request (or an Access error results).

**SetCloseDownMode**

```
mode: {Destroy, RetainPermanent, RetainTemporary}
```

Errors: Value

This request defines what will happen to the client’s resources at connection close. A connection starts in Destroy mode. The meaning of the close-down mode is described in section 10.

**KillClient**

```
resource: CARD32 or AllTemporary
```

Errors: Value

If a valid resource is specified, **KillClient** forces a close-down of the client that created the resource. If the client has already terminated in either RetainPermanent or RetainTemporary mode, all of the client’s resources are destroyed (see section 10). If AllTemporary is specified, then the resources of all clients that have terminated in RetainTemporary are destroyed.

**NoOperation**
This request has no arguments and no results, but the request length field allows the request to be any multiple of four bytes in length. The bytes contained in the request are uninterpreted by the server.

This request can be used in its minimum four byte form as padding where necessary by client libraries that find it convenient to force requests to begin on 64-bit boundaries.

10. Connection Close

At connection close, all event selections made by the client are discarded. If the client has the pointer actively grabbed, an UngrabPointer is performed. If the client has the keyboard actively grabbed, an UngrabKeyboard is performed. All passive grabs by the client are released. If the client has the server grabbed, an UngrabServer is performed. All selections (see SetSelectionOwner request) owned by the client are disowned. If close-down mode (see SetCloseDownMode request) is RetainPermanent or RetainTemporary, then all resources (including colormap entries) allocated by the client are marked as permanent or temporary, respectively (but this does not prevent other clients from explicitly destroying them). If the mode is Destroy, all of the client’s resources are destroyed.

When a client’s resources are destroyed, for each window in the client’s save-set, if the window is an inferior of a window created by the client, the save-set window is reparented to the closest ancestor such that the save-set window is not an inferior of a window created by the client. If the save-set window is unmapped, a MapWindow request is performed on it (even if it was not an inferior of a window created by the client). The reparenting leaves unchanged the absolute coordinates (with respect to the root window) of the upper-left outer corner of the save-set window. After save-set processing, all windows created by the client are destroyed. For each nonwindow resource created by the client, the appropriate Free request is performed. All colors and colormap entries allocated by the client are freed.

A server goes through a cycle of having no connections and having some connections. At every transition to the state of having no connections as a result of a connection closing with a Destroy close-down mode, the server resets its state as if it had just been started. This starts by destroying all lingering resources from clients that have terminated in RetainPermanent or RetainTemporary mode. It additionally includes deleting all but the predefined atom identifiers, deleting all properties on all root windows, resetting all device maps and attributes (key click, bell volume, acceleration), resetting the access control list, restoring the standard root tiles and cursors, restoring the default font path, and restoring the input focus to state PointerRoot.

Note that closing a connection with a close-down mode of RetainPermanent or RetainTemporary will not cause the server to reset.

11. Events

When a button press is processed with the pointer in some window W and no active pointer grab is in progress, the ancestors of W are searched from the root down, looking for a passive grab to activate. If no matching passive grab on the button exists, then an active grab is started automatically for the client receiving the event, and the last-pointer-grab time is set to the current server time. The effect is essentially equivalent to a GrabButton with arguments:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>event-window</td>
<td>Event window</td>
</tr>
<tr>
<td>event-mask</td>
<td>Client’s selected pointer events on the event window</td>
</tr>
</tbody>
</table>
Argument | Value
--- | ---
pointer-mode and keyboard-mode | Asynchronous
owner-events | True if the client has OwnerGrabButton selected on the event window, otherwise False
confineto | None
cursor | None

The grab is terminated automatically when the logical state of the pointer has all buttons released. **UngrabPointer** and **ChangeActivePointerGrab** can both be used to modify the active grab.

**KeyPress**, **KeyRelease**, **ButtonPress**, **ButtonRelease**, **MotionNotify**

*root, event*: WINDOW
*child*: WINDOW or **None**
*same-screen*: BOOL
*root-x, root-y, event-x, event-y*: INT16
*detail*: <see below>
*state*: SETofKEYBUTMASK
*time*: TIMESTAMP

These events are generated either when a key or button logically changes state or when the pointer logically moves. The generation of these logical changes may lag the physical changes if device event processing is frozen. Note that **KeyPress** and **KeyRelease** are generated for all keys, even those mapped to modifier bits. The source of the event is the window the pointer is in. The window the event is reported with respect to is called the event window. The event window is found by starting with the source window and looking up the hierarchy for the first window on which any client has selected interest in the event (provided no intervening window prohibits event generation by including the event type in its do-not-propagate-mask). The actual window used for reporting can be modified by active grabs and, in the case of keyboard events, can be modified by the focus window.

The root is the root window of the source window, and root-x and root-y are the pointer coordinates relative to root’s origin at the time of the event. Event is the event window. If the event window is on the same screen as root, then event-x and event-y are the pointer coordinates relative to the event window’s origin. Otherwise, event-x and event-y are zero. If the source window is an inferior of the event window, then child is set to the child of the event window that is an ancestor of (or is) the source window. Otherwise, it is set to **None**. The state component gives the logical state of the buttons and modifier keys just before the event. The detail component type varies with the event type:

<table>
<thead>
<tr>
<th>Event</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KeyPress</strong>, <strong>KeyRelease</strong></td>
<td><strong>KEYCODE</strong></td>
</tr>
</tbody>
</table>
MotionNotify events are only generated when the motion begins and ends in the window. The granularity of motion events is not guaranteed, but a client selecting for motion events is guaranteed to get at least one event when the pointer moves and comes to rest. Selecting PointerMotion receives events independent of the state of the pointer buttons. By selecting some subset of Button[1-5]Motion instead, MotionNotify events will only be received when one or more of the specified buttons are pressed. By selecting ButtonMotion, MotionNotify events will be received only when at least one button is pressed. The events are always of type MotionNotify, independent of the selection. If PointerMotionHint is selected, the server is free to send only one MotionNotify event (with detail Hint) to the client for the event window until either the key or button state changes, the pointer leaves the event window, or the client issues a QueryPointer or GetMotionEvents request.

EnterNotify
LeaveNotify

<table>
<thead>
<tr>
<th>Event</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>ButtonPress, ButtonRelease</td>
<td>BUTTON</td>
</tr>
<tr>
<td>MotionNotify</td>
<td>{ Normal, Hint }</td>
</tr>
</tbody>
</table>

If pointer motion or window hierarchy change causes the pointer to be in a different window than before, EnterNotify and LeaveNotify events are generated instead of a MotionNotify event. Only clients selecting EnterWindow on a window receive EnterNotify events, and only clients selecting LeaveWindow receive LeaveNotify events. The pointer position reported in the event is always the final position, not the initial position of the pointer. The root is the root window for this position, and root-x and root-y are the pointer coordinates relative to root’s origin at the time of the event. Event is the event window. Event is the event window. If the event window is on the same screen as root, then event-x and event-y are the pointer coordinates relative to the event window’s origin. Otherwise, event-x and event-y are zero. In a LeaveNotify event, if a child of the event window contains the initial position of the pointer, then the child component is set to that child. Otherwise, it is None. For an EnterNotify event, if a child of the event window contains the final pointer position, then the child component is set to that child. Otherwise, it is None. If the event window is the focus window or an inferior of the focus window, then focus is True. Otherwise, focus is False.

Normal pointer motion events have mode Normal. Pseudo-motion events when a grab activates have mode Grab, and pseudo-motion events when a grab deactivates have mode Ungrab.

All EnterNotify and LeaveNotify events caused by a hierarchy change are generated after any hierarchy event caused by that change (that is, UnmapNotify, MapNotify, ConfigureNotify, GravityNotify, CirculateNotify), but the ordering of EnterNotify and LeaveNotify events
with respect to **FocusOut**, **VisibilityNotify**, and **Expose** events is not constrained.

Normal events are generated as follows:

When the pointer moves from window A to window B and A is an inferior of B:

- **LeaveNotify** with detail **Ancestor** is generated on A.
- **LeaveNotify** with detail **Virtual** is generated on each window between A and B exclusive (in that order).
- **EnterNotify** with detail **Inferior** is generated on B.

When the pointer moves from window A to window B and B is an inferior of A:

- **LeaveNotify** with detail **Inferior** is generated on A.
- **EnterNotify** with detail **Virtual** is generated on each window between A and B exclusive (in that order).
- **EnterNotify** with detail **Ancestor** is generated on B.

When the pointer moves from window A to window B and window C is their least common ancestor:

- **LeaveNotify** with detail **Nonlinear** is generated on A.
- **LeaveNotify** with detail **NonlinearVirtual** is generated on each window between A and C exclusive (in that order).
- **EnterNotify** with detail **NonlinearVirtual** is generated on each window between C and B exclusive (in that order).
- **EnterNotify** with detail **Nonlinear** is generated on B.

When the pointer moves from window A to window B on different screens:

- **LeaveNotify** with detail **Nonlinear** is generated on A.
- If A is not a root window, **LeaveNotify** with detail **NonlinearVirtual** is generated on each window above A up to and including its root (in order).
- If B is not a root window, **EnterNotify** with detail **NonlinearVirtual** is generated on each window from B’s root down to but not including B (in order).
- **EnterNotify** with detail **Nonlinear** is generated on B.

When a pointer grab activates (but after any initial warp into a confine-to window and before generating any actual **ButtonPress** event that activates the grab), G is the grab-window for the grab, and P is the window the pointer is in:

- **EnterNotify** and **LeaveNotify** events with mode **Grab** are generated (as for **Normal** above) as if the pointer were to suddenly warp from its current position in P to some position in G. However, the pointer does not warp, and the pointer position is used as both the initial and final positions for the events.

When a pointer grab deactivates (but after generating any actual **ButtonRelease** event that deactivates the grab), G is the grab-window for the grab, and P is the window the pointer is in:

- **EnterNotify** and **LeaveNotify** events with mode **Ungrab** are generated (as for **Normal** above) as if the pointer were to suddenly warp from some position in G to its current position in P. However, the pointer does not warp, and the current pointer position is used as both the initial and final positions for the events.
FocusIn
FocusOut

event: WINDOW
mode: \{Normal, WhileGrabbed, Grab, Ungrab\}
detail: \{Ancestor, Virtual, Inferior, Nonlinear, NonlinearVirtual, Pointer, PointerRoot, None\}

These events are generated when the input focus changes and are reported to clients selecting FocusChange on the window. Events generated by SetInputFocus when the keyboard is not grabbed have mode Normal. Events generated by SetInputFocus when the keyboard is grabbed have mode WhileGrabbed. Events generated when a keyboard grab activates have mode Grab, and events generated when a keyboard grab deactivates have mode Ungrab.

All FocusOut events caused by a window unmap are generated after any UnmapNotify event, but the ordering of FocusOut with respect to generated EnterNotify, LeaveNotify, VisibilityNotify, and Expose events is not constrained.

Normal and WhileGrabbed events are generated as follows:

When the focus moves from window A to window B, A is an inferior of B, and the pointer is in window P:

- FocusOut with detail Ancestor is generated on A.
- FocusOut with detail Virtual is generated on each window between A and B exclusive (in order).
- FocusIn with detail Inferior is generated on B.
- If P is an inferior of B but P is not A or an inferior of A or an ancestor of A, FocusIn with detail Pointer is generated on each window below B down to and including P (in order).

When the focus moves from window A to window B, B is an inferior of A, and the pointer is in window P:

- If P is an inferior of A but P is not an inferior of B or an ancestor of B, FocusOut with detail Pointer is generated on each window from P up to but not including A (in order).
- FocusOut with detail Inferior is generated on A.
- FocusIn with detail Virtual is generated on each window between A and B exclusive (in order).
- FocusIn with detail Ancestor is generated on B.

When the focus moves from window A to window B, window C is their least common ancestor, and the pointer is in window P:

- If P is an inferior of A, FocusOut with detail Pointer is generated on each window from P up to but not including A (in order).
- FocusOut with detail Nonlinear is generated on A.
- FocusOut with detail NonlinearVirtual is generated on each window between A and C exclusive (in order).
- FocusIn with detail NonlinearVirtual is generated on each window between C and B exclusive (in order).
- FocusIn with detail Nonlinear is generated on B.
- If P is an inferior of B, FocusIn with detail Pointer is generated on each window below B down to and including P (in order).
When the focus moves from window A to window B on different screens and the pointer is in window P:

- If P is an inferior of A, **FocusOut** with detail **Pointer** is generated on each window from P up to but not including A (in order).
- **FocusOut** with detail **Nonlinear** is generated on A.
- If A is not a root window, **FocusOut** with detail **NonlinearVirtual** is generated on each window above A up to and including its root (in order).
- If B is not a root window, **FocusIn** with detail **NonlinearVirtual** is generated on each window from B’s root down to but not including B (in order).
- **FocusIn** with detail **Nonlinear** is generated on B.
- If P is an inferior of B, **FocusIn** with detail **Pointer** is generated on each window below B down to and including P (in order).

When the focus moves from window A to **PointerRoot** (or **None**) and the pointer is in window P:

- If P is an inferior of A, **FocusOut** with detail **Pointer** is generated on each window from P up to but not including A (in order).
- **FocusOut** with detail **Nonlinear** is generated on A.
- If A is not a root window, **FocusOut** with detail **NonlinearVirtual** is generated on each window above A up to and including its root (in order).
- **FocusIn** with detail **PointerRoot** (or **None**) is generated on all root windows.
- If the new focus is **PointerRoot**, **FocusIn** with detail **Pointer** is generated on each window from P’s root down to and including P (in order).

When the focus moves from **PointerRoot** (or **None**) to window A and the pointer is in window P:

- If the old focus is **PointerRoot**, **FocusOut** with detail **Pointer** is generated on each window from P up to and including P’s root (in order).
- **FocusOut** with detail **PointerRoot** (or **None**) is generated on all root windows.
- If A is not a root window, **FocusIn** with detail **NonlinearVirtual** is generated on each window from A’s root down to but not including A (in order).
- **FocusIn** with detail **Nonlinear** is generated on A.
- If P is an inferior of A, **FocusIn** with detail **Pointer** is generated on each window below A down to and including P (in order).

When the focus moves from **PointerRoot** to **None** (or vice versa) and the pointer is in window P:

- If the old focus is **PointerRoot**, **FocusOut** with detail **Pointer** is generated on each window from P up to and including P’s root (in order).
- **FocusOut** with detail **PointerRoot** (or **None**) is generated on all root windows.
- **FocusIn** with detail **None** (or **PointerRoot**) is generated on all root windows.
- If the new focus is **PointerRoot**, **FocusIn** with detail **Pointer** is generated on each window from P’s root down to and including P (in order).

When a keyboard grab activates (but before generating any actual **KeyPress** event that activates the grab), G is the grab-window for the grab, and F is the current focus:
FocusIn and FocusOut events with mode Grab are generated (as for Normal above) as if the focus were to change from F to G.

When a keyboard grab deactivates (but after generating any actual KeyRelease event that deactivates the grab), G is the grab-window for the grab, and F is the current focus:

FocusIn and FocusOut events with mode Ungrab are generated (as for Normal above) as if the focus were to change from G to F.

KeymapNotify

keys: LISTofCARD8

The value is a bit vector as described in QueryKeymap. This event is reported to clients selecting KeymapState on a window and is generated immediately after every EnterNotify and FocusIn.

Expose

window: WINDOW
x, y, width, height: CARD16

This event is reported to clients selecting Exposure on the window. It is generated when no valid contents are available for regions of a window, and either the regions are visible, the regions are viewable and the server is (perhaps newly) maintaining backing store on the window, or the window is not viewable but the server is (perhaps newly) honoring window’s backing-store attribute of Always or WhenMapped. The regions are decomposed into an arbitrary set of rectangles, and an Expose event is generated for each rectangle.

For a given action causing exposure events, the set of events for a given window are guaranteed to be reported contiguously. If count is zero, then no more Expose events for this window follow. If count is nonzero, then at least that many more Expose events for this window follow (and possibly more).

The x and y coordinates are relative to window’s origin and specify the upper-left corner of a rectangle. The width and height specify the extent of the rectangle.

Expose events are never generated on InputOnly windows.

All Expose events caused by a hierarchy change are generated after any hierarchy event caused by that change (for example, UnmapNotify, MapNotify, ConfigureNotify, GravityNotify, CirculateNotify). All Expose events on a given window are generated after any VisibilityNotify event on that window, but it is not required that all Expose events on all windows be generated after all Visibility events on all windows. The ordering of Expose events with respect to FocusOut, EnterNotify, and LeaveNotify events is not constrained.
GraphicsExposure

drawable: DRAWABLE
x, y, width, height: CARD16
count: CARD16
major-opcode: CARD8
minor-opcode: CARD16

This event is reported to a client using a graphics context with graphics-exposures selected and is generated when a destination region could not be computed due to an obscured or out-of-bounds source region. All of the regions exposed by a given graphics request are guaranteed to be reported contiguously. If count is zero then no more GraphicsExposure events for this window follow. If count is nonzero, then at least that many more GraphicsExposure events for this window follow (and possibly more).

The x and y coordinates are relative to drawable’s origin and specify the upper-left corner of a rectangle. The width and height specify the extent of the rectangle.

The major and minor opcodes identify the graphics request used. For the core protocol, major-opcode is always CopyArea or CopyPlane, and minor-opcode is always zero.

NoExposure

drawable: DRAWABLE
major-opcode: CARD8
minor-opcode: CARD16

This event is reported to a client using a graphics context with graphics-exposures selected and is generated when a graphics request that might produce GraphicsExposure events does not produce any. The drawable specifies the destination used for the graphics request.

The major and minor opcodes identify the graphics request used. For the core protocol, major-opcode is always CopyArea or CopyPlane, and the minor-opcode is always zero.

VisibilityNotify

window: WINDOW
state: { Unobscured, PartiallyObscured, FullyObscured }

This event is reported to clients selecting VisibilityChange on the window. In the following, the state of the window is calculated ignoring all of the window’s subwindows. When a window changes state from partially or fully obscured or not viewable to viewable and completely unobscured, an event with Unobscured is generated. When a window changes state from viewable and completely unobscured, from viewable and completely obscured, or from not viewable, to viewable and partially obscured, an event with PartiallyObscured is generated. When a window changes state from viewable and completely unobscured, from viewable and partially obscured, or from not viewable to viewable and fully obscured, an event with FullyObscured is generated.

VisibilityNotify events are never generated on InputOnly windows.

All VisibilityNotify events caused by a hierarchy change are generated after any hierarchy event caused by that change (for example, UnmapNotify, MapNotify, ConfigureNotify).
GravityNotify, CirculateNotify). Any VisibilityNotify event on a given window is generated before any Expose events on that window, but it is not required that all VisibilityNotify events on all windows be generated before all Expose events on all windows. The ordering of VisibilityNotify events with respect to FocusOut, EnterNotify, and LeaveNotify events is not constrained.

CreateNotify

\[\text{parent, window: WINDOW} \]
\[x, y: \text{INT16} \]
\[width, height, border-width: \text{CARD16} \]
\[override-redirect: \text{BOOL} \]

This event is reported to clients selecting SubstructureNotify on the parent and is generated when the window is created. The arguments are as in the CreateWindow request.

DestroyNotify

\[\text{event, window: WINDOW} \]

This event is reported to clients selecting StructureNotify on the window and to clients selecting SubstructureNotify on the parent. It is generated when the window is destroyed. The event is the window on which the event was generated, and the window is the window that is destroyed. The ordering of the DestroyNotify events is such that for any given window, DestroyNotify is generated on all inferiors of the window before being generated on the window itself. The ordering among siblings and across subhierarchies is not otherwise constrained.

UnmapNotify

\[\text{event, window: WINDOW} \]
\[from-configure: \text{BOOL} \]

This event is reported to clients selecting StructureNotify on the window and to clients selecting SubstructureNotify on the parent. It is generated when the window changes state from mapped to unmapped. The event is the window on which the event was generated, and the window is the window that is unmapped. The from-configure flag is True if the event was generated as a result of the window’s parent being resized when the window itself had a win-gravity of Unmap.

MapNotify

\[\text{event, window: WINDOW} \]
\[override-redirect: \text{BOOL} \]

This event is reported to clients selecting StructureNotify on the window and to clients selecting SubstructureNotify on the parent. It is generated when the window changes state from unmapped to mapped. The event is the window on which the event was generated, and the
window is the window that is mapped. The override-redirect flag is from the window’s attribute.

**MapRequest**

\[
\text{parent}, \text{window}: \text{WINDOW}
\]

This event is reported to the client selecting **SubstructureRedirect** on the parent and is generated when a **MapWindow** request is issued on an unmapped window with an override-redirect attribute of **False**.

**ReparentNotify**

\[
\text{event}, \text{window}, \text{parent}: \text{WINDOW} \\
x, y: \text{INT16} \\
\text{override-redirect}: \text{BOOL}
\]

This event is reported to clients selecting **SubstructureNotify** on either the old or the new parent and to clients selecting **StructureNotify** on the window. It is generated when the window is reparented. The event is the window on which the event was generated. The window is the window that has been rerooted. The parent specifies the new parent. The x and y coordinates are relative to the new parent’s origin and specify the position of the upper-left outer corner of the window. The override-redirect flag is from the window’s attribute.

**ConfigureNotify**

\[
\text{event}, \text{window}: \text{WINDOW} \\
x, y: \text{INT16} \\
width, height, \text{border-width}: \text{CARD16} \\
\text{above-sibling}: \text{WINDOW or None} \\
\text{override-redirect}: \text{BOOL}
\]

This event is reported to clients selecting **StructureNotify** on the window and to clients selecting **SubstructureNotify** on the parent. It is generated when a **ConfigureWindow** request actually changes the state of the window. The event is the window on which the event was generated, and the window is the window that is changed. The x and y coordinates are relative to the new parent’s origin and specify the position of the upper-left outer corner of the window. The width and height specify the inside size, not including the border. If above-sibling is **None**, then the window is on the bottom of the stack with respect to siblings. Otherwise, the window is immediately on top of the specified sibling. The override-redirect flag is from the window’s attribute.

**GravityNotify**

\[
\text{event}, \text{window}: \text{WINDOW} \\
x, y: \text{INT16}
\]

This event is reported to clients selecting **SubstructureNotify** on the parent and to clients selecting **StructureNotify** on the window. It is generated when a window is moved because of a
change in size of the parent. The event is the window on which the event was generated, and the window is the window that is moved. The x and y coordinates are relative to the new parent’s origin and specify the position of the upper-left outer corner of the window.

**ResizeRequest**
- **window**: WINDOW
- **width, height**: CARD16

This event is reported to the client selecting **ResizeRedirect** on the window and is generated when a **ConfigureWindow** request by some other client on the window attempts to change the size of the window. The width and height are the requested inside size, not including the border.

**ConfigureRequest**
- **parent, window**: WINDOW
- **x, y**: INT16
- **width, height, border-width**: CARD16
- **sibling**: WINDOW or None
- **stack-mode**: {Above, Below, TopIf, BottomIf, Opposite}
- **value-mask**: BITMASK

This event is reported to the client selecting **SubstructureRedirect** on the parent and is generated when a **ConfigureWindow** request is issued on the window by some other client. The value-mask indicates which components were specified in the request. The value-mask and the corresponding values are reported as given in the request. The remaining values are filled in from the current geometry of the window, except in the case of sibling and stack-mode, which are reported as None and Above (respectively) if not given in the request.

**CirculateNotify**
- **event, window**: WINDOW
- **place**: {Top, Bottom}

This event is reported to clients selecting **StructureNotify** on the window and to clients selecting **SubstructureNotify** on the parent. It is generated when the window is actually restacked from a **CirculateWindow** request. The event is the window on which the event was generated, and the window is the window that is restacked. If place is Top, the window is now on top of all siblings. Otherwise, it is below all siblings.

**CirculateRequest**
- **parent, window**: WINDOW
- **place**: {Top, Bottom}

This event is reported to the client selecting **SubstructureRedirect** on the parent and is generated when a **CirculateWindow** request is issued on the parent and a window actually needs to be
restacked. The window specifies the window to be restacked, and the place specifies what the new position in the stacking order should be.

**PropertyNotify**

- `window`: WINDOW
- `atom`: ATOM
- `state`: `{ NewValue, Deleted }
- `time`: TIMESTAMP

This event is reported to clients selecting PropertyChange on the window and is generated with state NewValue when a property of the window is changed using ChangeProperty or RotateProperties, even when adding zero-length data using ChangeProperty and when replacing all or part of a property with identical data using ChangeProperty or RotateProperties. It is generated with state Deleted when a property of the window is deleted using request DeleteProperty or GetProperty. The timestamp indicates the server time when the property was changed.

**SelectionClear**

- `owner`: WINDOW
- `selection`: ATOM
- `time`: TIMESTAMP

This event is reported to the current owner of a selection and is generated when a new owner is being defined by means of SetSelectionOwner. The timestamp is the last-change time recorded for the selection. The owner argument is the window that was specified by the current owner in its SetSelectionOwner request.

**SelectionRequest**

- `owner`: WINDOW
- `selection`: ATOM
- `target`: ATOM
- `property`: ATOM or None
- `requestor`: WINDOW
- `time`: TIMESTAMP or CurrentTime

This event is reported to the owner of a selection and is generated when a client issues a ConvertSelection request. The owner argument is the window that was specified in the SetSelectionOwner request. The remaining arguments are as in the ConvertSelection request.

The owner should convert the selection based on the specified target type and send a SelectionNotify back to the requestor. A complete specification for using selections is given in the X.Org standard Inter-Client Communication Conventions Manual.
SelectionNotify
    requestor: WINDOW
    selection, target: ATOM
    property: ATOM or None
    time: TIMESTAMP or CurrentTime

This event is generated by the server in response to a ConvertSelection request when there is no owner for the selection. When there is an owner, it should be generated by the owner using SendEvent. The owner of a selection should send this event to a requestor either when a selection has been converted and stored as a property or when a selection conversion could not be performed (indicated with property None).

ColormapNotify
    window: WINDOW
    colormap: COLORMAP or None
    new: BOOL
    state: { Installed, Uninstalled }

This event is reported to clients selecting ColormapChange on the window. It is generated with value True for new when the colormap attribute of the window is changed and is generated with value False for new when the colormap of a window is installed or uninstalled. In either case, the state indicates whether the colormap is currently installed.

MappingNotify
    request: { Modifier, Keyboard, Pointer }
    first-keycode, count: CARD8

This event is sent to all clients. There is no mechanism to express disinterest in this event. The detail indicates the kind of change that occurred: Modifiers for a successful SetModifierMapping, Keyboard for a successful ChangeKeyboardMapping, and Pointer for a successful SetPointerMapping. If the detail is Keyboard, then first-keycode and count indicate the range of altered keycodes.

ClientMessage
    window: WINDOW
    type: ATOM
    format: { 8, 16, 32 }
    data: LISTofINT8 or LISTofINT16 or LISTofINT32

This event is only generated by clients using SendEvent. The type specifies how the data is to be interpreted by the receiving client; the server places no interpretation on the type or the data. The format specifies whether the data should be viewed as a list of 8-bit, 16-bit, or 32-bit quantities, so that the server can correctly byte-swap, as necessary. The data always consists of either 20 8-bit values or 10 16-bit values or 5 32-bit values, although particular message types might not
make use of all of these values.

12. Flow Control and Concurrency

Whenever the server is writing to a given connection, it is permissible for the server to stop reading from that connection (but if the writing would block, it must continue to service other connections). The server is not required to buffer more than a single request per connection at one time. For a given connection to the server, a client can block while reading from the connection but should undertake to read (events and errors) when writing would block. Failure on the part of a client to obey this rule could result in a deadlocked connection, although deadlock is probably unlikely unless either the transport layer has very little buffering or the client attempts to send large numbers of requests without ever reading replies or checking for errors and events.

Whether or not a server is implemented with internal concurrency, the overall effect must be as if individual requests are executed to completion in some serial order, and requests from a given connection must be executed in delivery order (that is, the total execution order is a shuffle of the individual streams). The execution of a request includes validating all arguments, collecting all data for any reply, and generating and queueing all required events. However, it does not include the actual transmission of the reply and the events. In addition, the effect of any other cause that can generate multiple events (for example, activation of a grab or pointer motion) must effectively generate and queue all required events indivisibly with respect to all other causes and requests. For a request from a given client, any events destined for that client that are caused by executing the request must be sent to the client before any reply or error is sent.
Appendix A

KEYSYM Encoding

KEYSYM values are 32-bit integers that encode the symbols on the keycaps of a keyboard. The three most significant bits are always zero, which leaves a 29-bit number space. For convenience, KEYSYM values can be viewed as split into four bytes:

- Byte 1 is the most significant eight bits (three zero bits and the most-significant five bits of the 29-bit effective value)
- Byte 2 is the next most-significant eight bits
- Byte 3 is the next most-significant eight bits
- Byte 4 is the least-significant eight bits

There are six categories of KEYSYM values.

A.1 Special KEYSYM

There are two special values: NoSymbol and VoidSymbol. They are used to indicate the absence of symbols (see section 5).

<table>
<thead>
<tr>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
<th>Byte 4</th>
<th>Hex. value</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>#x00000000</td>
<td>NoSymbol</td>
</tr>
<tr>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>#x00FFFFFF</td>
<td>VoidSymbol</td>
</tr>
</tbody>
</table>

A.2 Latin-1 KEYSYM

The Latin-1 KEYSYMs occupy the range #x0020 to #x007E and #x00A0 to #00FF and represent the ISO 10646 / Unicode characters U+0020 to U+007E and U+00A0 to U+00FF, respectively.

A.3 Unicode KEYSYM

These occupy the range #x01000100 to #x0110FFFF and represent the ISO 10646 / Unicode characters U+0020 to U+007E and U+00A0 to U+00FF, respectively. The numeric value of a Unicode KEYSYM is the Unicode position of the corresponding character plus #x01000000. In the interest of backwards compatibility, clients should be able to process both the Unicode KEYSYM and the Legacy KEYSYM for those characters where both exist.

Dead keys, which place an accent on the next character entered, shall be encoded as Function KEYSYMs, and not as the Unicode KEYSYM corresponding to an equivalent combining character. Where a keycap indicates a specific function with a graphical symbol that is also available in Unicode (e.g., an upwards arrow for the cursor up function), the appropriate Function KEYSYM should be used, and not the Unicode KEYSYM corresponding to the depicted symbol.

A.4 Function KEYSYM

These represent keycap symbols that do not directly represent elements of a coded character set. Instead, they typically identify a software function, mode, or operation (e.g., cursor up, caps lock, insert) that can be activated using a dedicated key. Function KEYSYMs have zero values for
Bytes 1 and 2. Byte 3 distinguishes between several 8-bit sets within which byte 4 identifies the individual function key.

<table>
<thead>
<tr>
<th>Byte 3</th>
<th>Byte 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>Keyboard</td>
</tr>
<tr>
<td>254</td>
<td>Keyboard (XKB) Extension</td>
</tr>
<tr>
<td>253</td>
<td>3270</td>
</tr>
</tbody>
</table>

Within a national market, keyboards tend to be comparatively standard with respect to the character keys, but they can differ significantly on the miscellaneous function keys. Some have function keys left over from early timesharing days, others were designed for a specific application, such as text processing, web browsing, or accessing audiovisual data. The symbols on the keycaps can differ significantly between manufacturers and national markets, even where they denote the same software function (e.g., Ctrl in the U.S. versus Strg in Germany).

There are two ways of thinking about how to define KEYSYMs for such a world:

- The Engraving approach
- The Common approach

The Engraving approach is to create a KEYSYM for every unique key engraving. This is effectively taking the union of all key engravings on all keyboards. For example, some keyboards label function keys across the top as F1 through Fn, and others label them as PF1 through PFn. These would be different keys under the Engraving approach. Likewise, Lock would differ from Shift Lock, which is different from the up-arrow symbol that has the effect of changing lowercase to uppercase. There are lots of other aliases such as Del, DEL, Delete, Remove, and so forth. The Engraving approach makes it easy to decide if a new entry should be added to the KEYSYM set: if it does not exactly match an existing one, then a new one is created.

The Common approach tries to capture all of the keys present on an interesting number of keyboards, folding likely aliases into the same KEYSYM. For example, Del, DEL, and Delete are all merged into a single KEYSYM. Vendors can augment the KEYSYM set (using the vendor-specific encoding space) to include all of their unique keys that were not included in the standard set. Each vendor decides which of its keys map into the standard KEYSYMs, which presumably can be overridden by a user. It is more difficult to implement this approach, because judgment is required about when a sufficient set of keyboards implements an engraving to justify making it a KEYSYM in the standard set and about which engravings should be merged into a single KEYSYM.

Although neither scheme is perfect or elegant, the Common approach has been selected because it makes it easier to write a portable application. Having the Delete functionality merged into a single KEYSYM allows an application to implement a deletion function and expect reasonable bindings on a wide set of workstations. Under the Common approach, application writers are still free to look for and interpret vendor-specific KEYSYMs, but because they are in the extended set, the application developer is more conscious that they are writing the application in a nonportable fashion.

The Keyboard set is a miscellaneous collection of commonly occurring keys on keyboards. Within this set, the numeric keypad symbols are generally duplicates of symbols found on keys on the main part of the keyboard, but they are distinguished here because they often have a distinguishable semantics associated with them.
<table>
<thead>
<tr>
<th>KEYSYM value</th>
<th>Name</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>#xFF08</td>
<td>BACKSPACE, BACK SPACE, BACK CHAR</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF09</td>
<td>TAB</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF0A</td>
<td>LINEFEED, LF</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF0B</td>
<td>CLEAR</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF0D</td>
<td>RETURN, ENTER</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF13</td>
<td>PAUSE, HOLD</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF14</td>
<td>SCROLL LOCK</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF15</td>
<td>SYS REQ, SYSTEM REQUEST</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF1B</td>
<td>ESCAPE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF20</td>
<td>MULTI-KEY CHARACTER PREFACE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF21</td>
<td>KANJI, KANJI CONVERT</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF22</td>
<td>MUHENKAN</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF23</td>
<td>HENKAN MODE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF24</td>
<td>ROMAJI</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF25</td>
<td>HIRAGANA</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF26</td>
<td>KATAKANA</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF27</td>
<td>HIRAGANA/KATAKANA TOGGLE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF28</td>
<td>ZENKAKU</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF29</td>
<td>HANKAKU</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF2A</td>
<td>ZENKAKU/HANKAKU TOGGLE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF2B</td>
<td>TOUROKU</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF2C</td>
<td>MASSYO</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF2D</td>
<td>KANA LOCK</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF2E</td>
<td>KANA SHIFT</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF2F</td>
<td>EISU SHIFT</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF30</td>
<td>EISU TOGGLE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF31</td>
<td>HANGUL START/STOP (TOGGLE)</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF32</td>
<td>HANGUL START</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF33</td>
<td>HANGUL END, ENGLISH START</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF34</td>
<td>START HANGUL/HANJA CONVERSION</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF35</td>
<td>HANGUL JAMO MODE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF36</td>
<td>HANGUL ROMAJA MODE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF37</td>
<td>HANGUL CODE INPUT</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF38</td>
<td>HANGUL JEONJA MODE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF39</td>
<td>HANGUL BANJA MODE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF3A</td>
<td>HANGUL PREHANJA CONVERSION</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF3B</td>
<td>HANGUL POSTHANJA CONVERSION</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF3C</td>
<td>HANGUL SINGLE CANDIDATE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF3D</td>
<td>HANGUL MULTIPLE CANDIDATE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF3E</td>
<td>HANGUL PREVIOUS CANDIDATE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF3F</td>
<td>HANGUL SPECIAL SYMBOLS</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF50</td>
<td>HOME</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF51</td>
<td>LEFT, MOVE LEFT, LEFT ARROW</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF52</td>
<td>UP, MOVE UP, UP ARROW</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF53</td>
<td>RIGHT, MOVE RIGHT, RIGHT ARROW</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF54</td>
<td>DOWN, MOVE DOWN, DOWN ARROW</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF55</td>
<td>PRIOR, PREVIOUS, PAGE UP</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF56</td>
<td>NEXT, PAGE DOWN</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF57</td>
<td>END, EOL</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF58</td>
<td>BEGIN, BOL</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF60</td>
<td>SELECT, MARK</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF61</td>
<td>PRINT</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF62</td>
<td>EXECUTE, RUN, DO</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF63</td>
<td>INSERT, INSERT HERE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF65</td>
<td>UNDO, OOPS</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF66</td>
<td>REDO, AGAIN</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF67</td>
<td>MENU</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF68</td>
<td>FIND, SEARCH</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF69</td>
<td>CANCEL, STOP, ABORT, EXIT</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF6A</td>
<td>HELP</td>
<td>Keyboard</td>
</tr>
<tr>
<td>KEYSYM value</td>
<td>Name</td>
<td>Set</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>#xFF6B</td>
<td>BREAK</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF7E</td>
<td>MODE SWITCH, SCRIPT SWITCH, CHARACTER SET SWITCH</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF7F</td>
<td>NUM LOCK</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF80</td>
<td>KEYPAD SPACE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF89</td>
<td>KEYPAD TAB</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF8D</td>
<td>KEYPAD ENTER</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF91</td>
<td>KEYPAD F1, PF1, A</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF92</td>
<td>KEYPAD F2, PF2, B</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF93</td>
<td>KEYPAD F3, PF3, C</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF94</td>
<td>KEYPAD F4, PF4, D</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF95</td>
<td>KEYPAD HOME</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF96</td>
<td>KEYPAD LEFT</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF97</td>
<td>KEYPAD UP</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF98</td>
<td>KEYPAD RIGHT</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF99</td>
<td>KEYPAD DOWN</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF9A</td>
<td>KEYPAD PRIOR, PAGE UP</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF9B</td>
<td>KEYPAD NEXT, PAGE DOWN</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF9C</td>
<td>KEYPAD END</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF9D</td>
<td>KEYPAD BEGIN</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF9E</td>
<td>KEYPAD INSERT</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFF9F</td>
<td>KEYPAD DELETE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFAA</td>
<td>KEYPAD MULTIPLICATION SIGN, ASTERISK</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFAB</td>
<td>KEYPAD PLUS SIGN</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFAC</td>
<td>KEYPAD SEPARATOR, COMMA</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFAD</td>
<td>KEYPAD MINUS SIGN, HYPHEN</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFAE</td>
<td>KEYPAD DECIMAL POINT, FULL STOP</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFAF</td>
<td>KEYPAD DIVISION SIGN, SOLIDUS</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFB0</td>
<td>KEYPAD DIGIT ZERO</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFB1</td>
<td>KEYPAD DIGIT ONE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFB2</td>
<td>KEYPAD DIGIT TWO</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFB3</td>
<td>KEYPAD DIGIT THREE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFB4</td>
<td>KEYPAD DIGIT FOUR</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFB5</td>
<td>KEYPAD DIGIT FIVE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFB6</td>
<td>KEYPAD DIGIT SIX</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFB7</td>
<td>KEYPAD DIGIT SEVEN</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFB8</td>
<td>KEYPAD DIGIT EIGHT</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFB9</td>
<td>KEYPAD DIGIT NINE</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFBD</td>
<td>KEYPAD EQUALS SIGN</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFBEE</td>
<td>F1</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFBF</td>
<td>F2</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFC0</td>
<td>F3</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFC1</td>
<td>F4</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFC2</td>
<td>F5</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFC3</td>
<td>F6</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFC4</td>
<td>F7</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFC5</td>
<td>F8</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFC6</td>
<td>F9</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFC7</td>
<td>F10</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFC8</td>
<td>F11, L1</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFC9</td>
<td>F12, L2</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFCA</td>
<td>F13, L3</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFCB</td>
<td>F14, L4</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFCC</td>
<td>F15, L5</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFCD</td>
<td>F16, L6</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFCE</td>
<td>F17, L7</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFCF</td>
<td>F18, L8</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFD0</td>
<td>F19, L9</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFD1</td>
<td>F20, L10</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFD2</td>
<td>F21, R1</td>
<td>Keyboard</td>
</tr>
<tr>
<td>#xFFD3</td>
<td>F22, R2</td>
<td>Keyboard</td>
</tr>
</tbody>
</table>
The Keyboard (XKB) Extension set, which provides among other things a range of dead keys, is defined in “The X Keyboard Extension: Protocol Specification”, Appendix C.

The 3270 set defines additional keys that are specific to IBM 3270 terminals.
<table>
<thead>
<tr>
<th>KEYSYM value</th>
<th>Name</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>#xFD16</td>
<td>3270 PLAY</td>
<td>3270</td>
</tr>
<tr>
<td>#xFD17</td>
<td>3270 SETUP</td>
<td>3270</td>
</tr>
<tr>
<td>#xFD18</td>
<td>3270 RECORD</td>
<td>3270</td>
</tr>
<tr>
<td>#xFD19</td>
<td>3270 CHANGESCREEN</td>
<td>3270</td>
</tr>
<tr>
<td>#xFD1A</td>
<td>3270 DELETEWORD</td>
<td>3270</td>
</tr>
<tr>
<td>#xFD1B</td>
<td>3270 EXSELECT</td>
<td>3270</td>
</tr>
<tr>
<td>#xFD1C</td>
<td>3270 CURSORSELECT</td>
<td>3270</td>
</tr>
<tr>
<td>#xFD1D</td>
<td>3270 PRINTSCREEN</td>
<td>3270</td>
</tr>
<tr>
<td>#xFD1E</td>
<td>3270 ENTER</td>
<td>3270</td>
</tr>
</tbody>
</table>

### A.5 Vendor KEYSYMs

The KEYSYM number range #x10000000 to #x1FFFFFFF is available for vendor-specific extensions. Among these, the range #x11000000 to #x1100FFFF is designated for keypad KEYSYMs.

### A.6 Legacy KEYSYMs

These date from the time before ISO 10646 / Unicode was available. They represent characters from a number of different older 8-bit coded character sets and have zero values for bytes 1 and 2. Byte 3 indicates a coded character set and byte 4 is the 8-bit value of the particular character within that set.

<table>
<thead>
<tr>
<th>Byte 3</th>
<th>Byte 4</th>
<th>Byte 3</th>
<th>Byte 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Latin-2</td>
<td>11</td>
<td>APL</td>
</tr>
<tr>
<td>2</td>
<td>Latin-3</td>
<td>12</td>
<td>Hebrew</td>
</tr>
<tr>
<td>3</td>
<td>Latin-4</td>
<td>13</td>
<td>Thai</td>
</tr>
<tr>
<td>4</td>
<td>Kana</td>
<td>14</td>
<td>Korean</td>
</tr>
<tr>
<td>5</td>
<td>Arabic</td>
<td>15</td>
<td>Latin-5</td>
</tr>
<tr>
<td>6</td>
<td>Cyrillic</td>
<td>16</td>
<td>Latin-6</td>
</tr>
<tr>
<td>7</td>
<td>Greek</td>
<td>17</td>
<td>Latin-7</td>
</tr>
<tr>
<td>8</td>
<td>Technical</td>
<td>18</td>
<td>Latin-8</td>
</tr>
<tr>
<td>9</td>
<td>Special</td>
<td>19</td>
<td>Latin-9</td>
</tr>
<tr>
<td>10</td>
<td>Publishing</td>
<td>32</td>
<td>Currency</td>
</tr>
</tbody>
</table>

Each character set contains gaps where codes have been removed that were duplicates with codes in previous character sets (that is, character sets with lesser byte 3 value).

The Latin, Arabic, Cyrillic, Greek, Hebrew, and Thai sets were taken from the early drafts of the relevant ISO 8859 parts available at the time. However, in the case of the Cyrillic and Greek sets, these turned out differently in the final versions of the ISO standard. The Technical, Special, and Publishing sets are based on Digital Equipment Corporation standards, as no equivalent international standards were available at the time.

The table below lists all standardized Legacy KEYSYMs, along with the name used in the source document. Where there exists an unambiguous equivalent in Unicode, as it is the case with all ISO 8859 characters, it is given in the second column as a cross reference. Where there is no Unicode number provided, the exact semantics of the KEYSYM may have been lost and a Unicode KEYSYM should be used instead, if available.

As support of Unicode KEYSYMs increases, some or all of the Legacy KEYSYMs may be phased out and withdrawn in future versions of this standard. Most KEYSYMs in the sets Technical, Special, Publishing, APL and Currency (with the exception of #x20AC) were probably never...
used in practice, and were not supported by pre-Unicode fonts. In particular, the Currency set, which was copied from Unicode, has already been deprecated by the introduction of the Unicode KEYSYMs.

<table>
<thead>
<tr>
<th>KEYSYM value</th>
<th>Unicode value</th>
<th>Name</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>#x01A1</td>
<td>U+0104</td>
<td>LATIN CAPITAL LETTER A WITH OGONEK</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01A2</td>
<td>U+02D8</td>
<td>BREVE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01A3</td>
<td>U+0141</td>
<td>LATIN CAPITAL LETTER L WITH STROKE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01A5</td>
<td>U+013D</td>
<td>LATIN CAPITAL LETTER L WITH CARON</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01A6</td>
<td>U+015A</td>
<td>LATIN CAPITAL LETTER S WITH ACUTE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01A9</td>
<td>U+0160</td>
<td>LATIN CAPITAL LETTER S WITH CARON</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01AA</td>
<td>U+015E</td>
<td>LATIN CAPITAL LETTER S WITH CEDILLA</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01AB</td>
<td>U+0164</td>
<td>LATIN CAPITAL LETTER T WITH CARON</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01AC</td>
<td>U+0179</td>
<td>LATIN CAPITAL LETTER Z WITH ACUTE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01AE</td>
<td>U+017D</td>
<td>LATIN CAPITAL LETTER Z WITH CARON</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01AF</td>
<td>U+017B</td>
<td>LATIN CAPITAL LETTER Z WITH DOT ABOVE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01B1</td>
<td>U+0105</td>
<td>LATIN SMALL LETTER A WITH OGONEK</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01B2</td>
<td>U+02DB</td>
<td>OGONEK</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01B3</td>
<td>U+0142</td>
<td>LATIN SMALL LETTER L WITH STROKE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01B5</td>
<td>U+013E</td>
<td>LATIN SMALL LETTER L WITH CARON</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01B6</td>
<td>U+015B</td>
<td>LATIN SMALL LETTER S WITH ACUTE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01B7</td>
<td>U+02C7</td>
<td>CARON</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01B9</td>
<td>U+0161</td>
<td>LATIN SMALL LETTER S WITH CARON</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01BA</td>
<td>U+015F</td>
<td>LATIN SMALL LETTER S WITH CEDILLA</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01BB</td>
<td>U+0165</td>
<td>LATIN SMALL LETTER T WITH CARON</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01BC</td>
<td>U+017A</td>
<td>LATIN SMALL LETTER Z WITH ACUTE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01BD</td>
<td>U+02DD</td>
<td>DOUBLE ACUTE ACCENT</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01BE</td>
<td>U+017E</td>
<td>LATIN SMALL LETTER Z WITH CARON</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01BF</td>
<td>U+017C</td>
<td>LATIN SMALL LETTER Z WITH DOT ABOVE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01C0</td>
<td>U+0154</td>
<td>LATIN CAPITAL LETTER R WITH ACUTE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01C3</td>
<td>U+0102</td>
<td>LATIN CAPITAL LETTER A WITH BREVE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01C5</td>
<td>U+0139</td>
<td>LATIN CAPITAL LETTER L WITH ACUTE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01C6</td>
<td>U+0106</td>
<td>LATIN CAPITAL LETTER C WITH ACUTE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01C8</td>
<td>U+010C</td>
<td>LATIN CAPITAL LETTER C WITH CARON</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01CA</td>
<td>U+0118</td>
<td>LATIN CAPITAL LETTER E WITH OGONEK</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01CC</td>
<td>U+011A</td>
<td>LATIN CAPITAL LETTER E WITH CARON</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01CF</td>
<td>U+010E</td>
<td>LATIN CAPITAL LETTER D WITH CARON</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01D0</td>
<td>U+0110</td>
<td>LATIN CAPITAL LETTER D WITH STROKE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01D1</td>
<td>U+0143</td>
<td>LATIN CAPITAL LETTER N WITH ACUTE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01D2</td>
<td>U+0147</td>
<td>LATIN CAPITAL LETTER N WITH CARON</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01D5</td>
<td>U+0150</td>
<td>LATIN CAPITAL LETTER O WITH DOUBLE ACUTE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01D8</td>
<td>U+0158</td>
<td>LATIN CAPITAL LETTER R WITH CARON</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01D9</td>
<td>U+016E</td>
<td>LATIN CAPITAL LETTER U WITH RING ABOVE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01DB</td>
<td>U+0170</td>
<td>LATIN CAPITAL LETTER U WITH DOUBLE ACUTE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01DE</td>
<td>U+0162</td>
<td>LATIN CAPITAL LETTER T WITH CEDILLA</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01E0</td>
<td>U+0155</td>
<td>LATIN SMALL LETTER R WITH ACUTE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01E3</td>
<td>U+0103</td>
<td>LATIN SMALL LETTER A WITH BREVE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01E5</td>
<td>U+013A</td>
<td>LATIN SMALL LETTER L WITH ACUTE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01E6</td>
<td>U+0107</td>
<td>LATIN SMALL LETTER C WITH ACUTE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01E8</td>
<td>U+010D</td>
<td>LATIN SMALL LETTER C WITH CARON</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01EA</td>
<td>U+0119</td>
<td>LATIN SMALL LETTER E WITH OGONEK</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01EC</td>
<td>U+011B</td>
<td>LATIN SMALL LETTER E WITH CARON</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01EF</td>
<td>U+010F</td>
<td>LATIN SMALL LETTER D WITH CARON</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01F0</td>
<td>U+0111</td>
<td>LATIN SMALL LETTER D WITH STROKE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01F1</td>
<td>U+0144</td>
<td>LATIN SMALL LETTER N WITH ACUTE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01F2</td>
<td>U+0148</td>
<td>LATIN SMALL LETTER N WITH CARON</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01F5</td>
<td>U+0151</td>
<td>LATIN SMALL LETTER O WITH DOUBLE ACUTE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01F8</td>
<td>U+0159</td>
<td>LATIN SMALL LETTER R WITH CARON</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01F9</td>
<td>U+016F</td>
<td>LATIN SMALL LETTER U WITH RING ABOVE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01FB</td>
<td>U+0171</td>
<td>LATIN SMALL LETTER U WITH DOUBLE ACUTE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>KEYSYM value</td>
<td>Unicode value</td>
<td>Name</td>
<td>Set</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td>#x01FE</td>
<td>U+0163</td>
<td>LATIN SMALL LETTER T WITH CEDILLA</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x01FF</td>
<td>U+02D9</td>
<td>DOT ABOVE</td>
<td>Latin-2</td>
</tr>
<tr>
<td>#x02A1</td>
<td>U+0126</td>
<td>LATIN CAPITAL LETTER H WITH STROKE</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02A6</td>
<td>U+0124</td>
<td>LATIN CAPITAL LETTER H WITH CIRCUMFLEX</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02A9</td>
<td>U+0130</td>
<td>LATIN CAPITAL LETTER I WITH DOT ABOVE</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02AB</td>
<td>U+011E</td>
<td>LATIN CAPITAL LETTER G WITH BREVE</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02AC</td>
<td>U+0134</td>
<td>LATIN CAPITAL LETTER J WITH CIRCUMFLEX</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02B1</td>
<td>U+0127</td>
<td>LATIN SMALL LETTER H WITH STROKE</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02B6</td>
<td>U+0125</td>
<td>LATIN SMALL LETTER H WITH CIRCUMFLEX</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02B9</td>
<td>U+0131</td>
<td>LATIN SMALL LETTER DOTLESS I</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02BB</td>
<td>U+011F</td>
<td>LATIN SMALL LETTER G WITH BREVE</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02BC</td>
<td>U+0135</td>
<td>LATIN SMALL LETTER J WITH CIRCUMFLEX</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02CC</td>
<td>U+010A</td>
<td>LATIN CAPITAL LETTER C WITH DOT ABOVE</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02CD</td>
<td>U+0108</td>
<td>LATIN CAPITAL LETTER C WITH CIRCUMFLEX</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02DE</td>
<td>U+0120</td>
<td>LATIN CAPITAL LETTER G WITH DOT ABOVE</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02DF</td>
<td>U+011C</td>
<td>LATIN CAPITAL LETTER G WITH CIRCUMFLEX</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02DD</td>
<td>U+016C</td>
<td>LATIN CAPITAL LETTER U WITH BREVE</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02DE</td>
<td>U+015C</td>
<td>LATIN CAPITAL LETTER S WITH CIRCUMFLEX</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02E5</td>
<td>U+010B</td>
<td>LATIN SMALL LETTER C WITH DOT ABOVE</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02E6</td>
<td>U+0109</td>
<td>LATIN SMALL LETTER C WITH CIRCUMFLEX</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02F5</td>
<td>U+0121</td>
<td>LATIN SMALL LETTER G WITH DOT ABOVE</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02F8</td>
<td>U+011D</td>
<td>LATIN SMALL LETTER G WITH CIRCUMFLEX</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02FD</td>
<td>U+016D</td>
<td>LATIN SMALL LETTER U WITH BREVE</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x02FE</td>
<td>U+015D</td>
<td>LATIN SMALL LETTER S WITH CIRCUMFLEX</td>
<td>Latin-3</td>
</tr>
<tr>
<td>#x03A2</td>
<td>U+0138</td>
<td>LATIN SMALL LETTER KRA</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03A3</td>
<td>U+0156</td>
<td>LATIN CAPITAL LETTER R WITH CEDILLA</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03A5</td>
<td>U+0128</td>
<td>LATIN CAPITAL LETTER I WITH TILDE</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03A6</td>
<td>U+013B</td>
<td>LATIN CAPITAL LETTER L WITH CEDILLA</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03AA</td>
<td>U+0112</td>
<td>LATIN CAPITAL LETTER E WITH MACRON</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03AB</td>
<td>U+0122</td>
<td>LATIN CAPITAL LETTER G WITH CEDILLA</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03AC</td>
<td>U+0166</td>
<td>LATIN CAPITAL LETTER T WITH STROKE</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03B3</td>
<td>U+0157</td>
<td>LATIN SMALL LETTER R WITH CEDILLA</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03B5</td>
<td>U+0129</td>
<td>LATIN SMALL LETTER I WITH TILDE</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03B6</td>
<td>U+013C</td>
<td>LATIN SMALL LETTER L WITH CEDILLA</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03BA</td>
<td>U+0113</td>
<td>LATIN SMALL LETTER E WITH MACRON</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03BB</td>
<td>U+0123</td>
<td>LATIN SMALL LETTER G WITH CEDILLA</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03BC</td>
<td>U+0167</td>
<td>LATIN SMALL LETTER T WITH STROKE</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03BD</td>
<td>U+014A</td>
<td>LATIN CAPITAL LETTER ENG</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03BF</td>
<td>U+014B</td>
<td>LATIN SMALL LETTER ENG</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03C0</td>
<td>U+0100</td>
<td>LATIN CAPITAL LETTER A WITH MACRON</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03C7</td>
<td>U+012E</td>
<td>LATIN CAPITAL LETTER I WITH OGONEK</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03CC</td>
<td>U+0116</td>
<td>LATIN CAPITAL LETTER E WITH DOT ABOVE</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03CF</td>
<td>U+012A</td>
<td>LATIN CAPITAL LETTER I WITH MACRON</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03D1</td>
<td>U+0145</td>
<td>LATIN CAPITAL LETTER N WITH CEDILLA</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03D2</td>
<td>U+014C</td>
<td>LATIN CAPITAL LETTER O WITH MACRON</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03D3</td>
<td>U+0136</td>
<td>LATIN CAPITAL LETTER K WITH CEDILLA</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03D9</td>
<td>U+0172</td>
<td>LATIN CAPITAL LETTER U WITH OGONEK</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03DD</td>
<td>U+0168</td>
<td>LATIN CAPITAL LETTER U WITH TILDE</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03DE</td>
<td>U+016A</td>
<td>LATIN CAPITAL LETTER U WITH MACRON</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03E0</td>
<td>U+0101</td>
<td>LATIN SMALL LETTER A WITH MACRON</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03E7</td>
<td>U+012F</td>
<td>LATIN SMALL LETTER I WITH OGONEK</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03EC</td>
<td>U+0117</td>
<td>LATIN SMALL LETTER E WITH DOT ABOVE</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03EF</td>
<td>U+012B</td>
<td>LATIN SMALL LETTER I WITH MACRON</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03F1</td>
<td>U+0146</td>
<td>LATIN SMALL LETTER N WITH CEDILLA</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03F2</td>
<td>U+014D</td>
<td>LATIN SMALL LETTER O WITH MACRON</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03F3</td>
<td>U+0137</td>
<td>LATIN SMALL LETTER K WITH CEDILLA</td>
<td>Latin-4</td>
</tr>
<tr>
<td>KEYSYM value</td>
<td>Unicode value</td>
<td>Name</td>
<td>Set</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td>#x03F9</td>
<td>U+0173</td>
<td>LATIN SMALL LETTER U WITH OGONEK</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03FD</td>
<td>U+0169</td>
<td>LATIN SMALL LETTER U WITH TILDE</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x03FE</td>
<td>U+016B</td>
<td>LATIN SMALL LETTER U WITH MACRON</td>
<td>Latin-4</td>
</tr>
<tr>
<td>#x047E</td>
<td>U+203E</td>
<td>OVERLINE</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04A1</td>
<td>U+3002</td>
<td>KANA FULL STOP</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04A2</td>
<td>U+300C</td>
<td>KANA OPENING BRACKET</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04A3</td>
<td>U+300D</td>
<td>KANA CLOSING BRACKET</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04A4</td>
<td>U+3001</td>
<td>KANA COMMA</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04A5</td>
<td>U+30FB</td>
<td>KANA CONJUNCTIVE</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04A6</td>
<td>U+30F2</td>
<td>KANA LETTER WO</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04A7</td>
<td>U+30A1</td>
<td>KANA LETTER SMALL A</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04A8</td>
<td>U+30A3</td>
<td>KANA LETTER SMALL I</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04A9</td>
<td>U+30A5</td>
<td>KANA LETTER SMALL U</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04AA</td>
<td>U+30A7</td>
<td>KANA LETTER SMALL E</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04AB</td>
<td>U+30A9</td>
<td>KANA LETTER SMALL O</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04AC</td>
<td>U+30E3</td>
<td>KANA LETTER SMALL YA</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04AD</td>
<td>U+30E5</td>
<td>KANA LETTER SMALL YU</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04AE</td>
<td>U+30E7</td>
<td>KANA LETTER SMALL YO</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04AF</td>
<td>U+30C3</td>
<td>KANA LETTER SMALL TSU</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04B0</td>
<td>U+30FC</td>
<td>PROLONGED SOUND SYMBOL</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04B1</td>
<td>U+30A2</td>
<td>KANA LETTER A</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04B2</td>
<td>U+30A4</td>
<td>KANA LETTER I</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04B3</td>
<td>U+30A6</td>
<td>KANA LETTER U</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04B4</td>
<td>U+30A8</td>
<td>KANA LETTER E</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04B5</td>
<td>U+30A9</td>
<td>KANA LETTER O</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04B6</td>
<td>U+30AB</td>
<td>KANA LETTER KA</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04B7</td>
<td>U+30AD</td>
<td>KANA LETTER KI</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04B8</td>
<td>U+30AF</td>
<td>KANA LETTER KU</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04B9</td>
<td>U+30B1</td>
<td>KANA LETTER KE</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04BA</td>
<td>U+30B3</td>
<td>KANA LETTER KO</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04BB</td>
<td>U+30B5</td>
<td>KANA LETTER SA</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04BC</td>
<td>U+30B7</td>
<td>KANA LETTER SHI</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04BD</td>
<td>U+30B9</td>
<td>KANA LETTER SU</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04BE</td>
<td>U+30BB</td>
<td>KANA LETTER SE</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04BF</td>
<td>U+30BD</td>
<td>KANA LETTER SO</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04C0</td>
<td>U+30BF</td>
<td>KANA LETTER TA</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04C1</td>
<td>U+30C1</td>
<td>KANA LETTER CHI</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04C2</td>
<td>U+30C4</td>
<td>KANA LETTER TSU</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04C3</td>
<td>U+30C6</td>
<td>KANA LETTER TE</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04C4</td>
<td>U+30C8</td>
<td>KANA LETTER TO</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04C5</td>
<td>U+30CA</td>
<td>KANA LETTER NA</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04C6</td>
<td>U+30CB</td>
<td>KANA LETTER NI</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04C7</td>
<td>U+30CC</td>
<td>KANA LETTER NU</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04C8</td>
<td>U+30CD</td>
<td>KANA LETTER NE</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04C9</td>
<td>U+30CE</td>
<td>KANA LETTER NO</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04CA</td>
<td>U+30CF</td>
<td>KANA LETTER HA</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04CB</td>
<td>U+30D2</td>
<td>KANA LETTER HI</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04CC</td>
<td>U+30D5</td>
<td>KANA LETTER FU</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04CD</td>
<td>U+30D8</td>
<td>KANA LETTER HE</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04CE</td>
<td>U+30DB</td>
<td>KANA LETTER HO</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04CF</td>
<td>U+30DE</td>
<td>KANA LETTER MA</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04D0</td>
<td>U+30DF</td>
<td>KANA LETTER MI</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04D1</td>
<td>U+30E0</td>
<td>KANA LETTER MU</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04D2</td>
<td>U+30E1</td>
<td>KANA LETTER ME</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04D3</td>
<td>U+30E2</td>
<td>KANA LETTER MO</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04D4</td>
<td>U+30E4</td>
<td>KANA LETTER YA</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04D5</td>
<td>U+30E6</td>
<td>KANA LETTER YU</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04D6</td>
<td>U+30E8</td>
<td>KANA LETTER YO</td>
<td>Kana</td>
</tr>
<tr>
<td>KEYSYM value</td>
<td>Unicode value</td>
<td>Name</td>
<td>Set</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td>#x04D7</td>
<td>U+30E9</td>
<td>KANA LETTER RA</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04D8</td>
<td>U+30EA</td>
<td>KANA LETTER RI</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04D9</td>
<td>U+30EB</td>
<td>KANA LETTER RU</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04DA</td>
<td>U+30EC</td>
<td>KANA LETTER RE</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04DB</td>
<td>U+30ED</td>
<td>KANA LETTER RO</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04DC</td>
<td>U+30EF</td>
<td>KANA LETTER WA</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04DD</td>
<td>U+30F3</td>
<td>KANA LETTER N</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04DE</td>
<td>U+309B</td>
<td>VOICED SOUND SYMBOL</td>
<td>Kana</td>
</tr>
<tr>
<td>#x04DF</td>
<td>U+309C</td>
<td>SEMIVOICED SOUND SYMBOL</td>
<td>Kana</td>
</tr>
<tr>
<td>#x05AC</td>
<td>U+060C</td>
<td>ARABIC COMMA</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05BB</td>
<td>U+061B</td>
<td>ARABIC SEMICOLON</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05BF</td>
<td>U+061F</td>
<td>ARABIC QUESTION MARK</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05C1</td>
<td>U+0621</td>
<td>ARABIC LETTER HAMZA</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05C2</td>
<td>U+0622</td>
<td>ARABIC LETTER ALEF WITH MADDA ABOVE</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05C3</td>
<td>U+0623</td>
<td>ARABIC LETTER ALEF WITH HAMZA ABOVE</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05C4</td>
<td>U+0624</td>
<td>ARABIC LETTER WAW WITH HAMZA ABOVE</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05C5</td>
<td>U+0625</td>
<td>ARABIC LETTER ALEF WITH HAMZA BELOW</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05C6</td>
<td>U+0626</td>
<td>ARABIC LETTER YEH WITH HAMZA ABOVE</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05C7</td>
<td>U+0627</td>
<td>ARABIC LETTER ALEF</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05C8</td>
<td>U+0628</td>
<td>ARABIC LETTER BEH</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05C9</td>
<td>U+0629</td>
<td>ARABIC LETTER TEH MARBUTA</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05CA</td>
<td>U+062A</td>
<td>ARABIC LETTER TEH</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05CB</td>
<td>U+062B</td>
<td>ARABIC LETTER THEH</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05CC</td>
<td>U+062C</td>
<td>ARABIC LETTER JEEM</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05CD</td>
<td>U+062D</td>
<td>ARABIC LETTER HAH</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05CE</td>
<td>U+062E</td>
<td>ARABIC LETTER KHAH</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05CF</td>
<td>U+062F</td>
<td>ARABIC LETTER DAL</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05D0</td>
<td>U+0630</td>
<td>ARABIC LETTER THAL</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05D1</td>
<td>U+0631</td>
<td>ARABIC LETTER REH</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05D2</td>
<td>U+0632</td>
<td>ARABIC LETTER ZAIN</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05D3</td>
<td>U+0633</td>
<td>ARABIC LETTER SEEN</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05D4</td>
<td>U+0634</td>
<td>ARABIC LETTER SHEEN</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05D5</td>
<td>U+0635</td>
<td>ARABIC LETTER SAD</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05D6</td>
<td>U+0636</td>
<td>ARABIC LETTER DAD</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05D7</td>
<td>U+0637</td>
<td>ARABIC LETTER TAH</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05D8</td>
<td>U+0638</td>
<td>ARABIC LETTER ZAH</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05D9</td>
<td>U+0639</td>
<td>ARABIC LETTER AIN</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05DA</td>
<td>U+063A</td>
<td>ARABIC LETTER GHAIN</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05E0</td>
<td>U+0640</td>
<td>ARABIC TATWEEL</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05E1</td>
<td>U+0641</td>
<td>ARABIC LETTER FEH</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05E2</td>
<td>U+0642</td>
<td>ARABIC LETTER QAF</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05E3</td>
<td>U+0643</td>
<td>ARABIC LETTER KAF</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05E4</td>
<td>U+0644</td>
<td>ARABIC LETTER LAM</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05E5</td>
<td>U+0645</td>
<td>ARABIC LETTER MEEM</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05E6</td>
<td>U+0646</td>
<td>ARABIC LETTER NOON</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05E7</td>
<td>U+0647</td>
<td>ARABIC LETTER HEH</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05E8</td>
<td>U+0648</td>
<td>ARABIC LETTER WAW</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05E9</td>
<td>U+0649</td>
<td>ARABIC LETTER ALEF MAKURA</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05EA</td>
<td>U+064A</td>
<td>ARABIC LETTER YEH</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05EB</td>
<td>U+064B</td>
<td>ARABIC FATHATAN</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05EC</td>
<td>U+064C</td>
<td>ARABIC DAMMATAN</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05ED</td>
<td>U+064D</td>
<td>ARABIC KASRATAN</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05EE</td>
<td>U+064E</td>
<td>ARABIC FATHA</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05EF</td>
<td>U+064F</td>
<td>ARABIC DAMMA</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05F0</td>
<td>U+0650</td>
<td>ARABIC KASRA</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05F1</td>
<td>U+0651</td>
<td>ARABIC SHADDA</td>
<td>Arabic</td>
</tr>
<tr>
<td>#x05F2</td>
<td>U+0652</td>
<td>ARABIC SUKUN</td>
<td>Arabic</td>
</tr>
<tr>
<td>KEYSYM value</td>
<td>Unicode value</td>
<td>Name</td>
<td>Set</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>#x06A1</td>
<td>U+0452</td>
<td>CYRILLIC SMALL LETTER DJE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06A2</td>
<td>U+0453</td>
<td>CYRILLIC SMALL LETTER GJE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06A3</td>
<td>U+0451</td>
<td>CYRILLIC SMALL LETTER IO</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06A4</td>
<td>U+0454</td>
<td>CYRILLIC SMALL LETTER UKRAINIAN IE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06A5</td>
<td>U+0455</td>
<td>CYRILLIC SMALL LETTER DZE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06A6</td>
<td>U+0456</td>
<td>CYRILLIC SMALL LETTER BYELORUSSIAN-UKRAINIAN I</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06A7</td>
<td>U+0457</td>
<td>CYRILLIC SMALL LETTER YI</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06A8</td>
<td>U+0458</td>
<td>CYRILLIC SMALL LETTER JE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06A9</td>
<td>U+0459</td>
<td>CYRILLIC SMALL LETTER LJE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06AA</td>
<td>U+045A</td>
<td>CYRILLIC SMALL LETTER NJE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06AB</td>
<td>U+045B</td>
<td>CYRILLIC SMALL LETTER TSHE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06AC</td>
<td>U+045C</td>
<td>CYRILLIC SMALL LETTER KJE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06AD</td>
<td>U+0491</td>
<td>CYRILLIC SMALL LETTER GHE WITH UPTURN</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06AE</td>
<td>U+045E</td>
<td>CYRILLIC SMALL LETTER SHORT U</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06AF</td>
<td>U+045F</td>
<td>CYRILLIC SMALL LETTER DZHE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06B0</td>
<td>U+2116</td>
<td>NUMERO SIGN</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06B1</td>
<td>U+0402</td>
<td>CYRILLIC CAPITAL LETTER DJE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06B2</td>
<td>U+0403</td>
<td>CYRILLIC CAPITAL LETTER GJE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06B3</td>
<td>U+0401</td>
<td>CYRILLIC CAPITAL LETTER IO</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06B4</td>
<td>U+0404</td>
<td>CYRILLIC CAPITAL LETTER UKRAINIAN IE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06B5</td>
<td>U+0405</td>
<td>CYRILLIC CAPITAL LETTER DZE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06B6</td>
<td>U+0406</td>
<td>CYRILLIC CAPITAL LETTER BYELORUSSIAN-UKRAINIAN I</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06B7</td>
<td>U+0407</td>
<td>CYRILLIC CAPITAL LETTER YI</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06B8</td>
<td>U+0408</td>
<td>CYRILLIC CAPITAL LETTER JE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06B9</td>
<td>U+0409</td>
<td>CYRILLIC CAPITAL LETTER LJE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06BA</td>
<td>U+040A</td>
<td>CYRILLIC CAPITAL LETTER NJE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06BB</td>
<td>U+040B</td>
<td>CYRILLIC CAPITAL LETTER TSHE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06BC</td>
<td>U+040C</td>
<td>CYRILLIC CAPITAL LETTER KJE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06BD</td>
<td>U+0490</td>
<td>CYRILLIC CAPITAL LETTER GHE WITH UPTURN</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06BE</td>
<td>U+040E</td>
<td>CYRILLIC CAPITAL LETTER SHORT U</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06BF</td>
<td>U+040F</td>
<td>CYRILLIC CAPITAL LETTER DZHE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06C0</td>
<td>U+044E</td>
<td>CYRILLIC SMALL LETTER YU</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06C1</td>
<td>U+0430</td>
<td>CYRILLIC SMALL LETTER A</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06C2</td>
<td>U+0431</td>
<td>CYRILLIC SMALL LETTER BE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06C3</td>
<td>U+0446</td>
<td>CYRILLIC SMALL LETTER TSE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06C4</td>
<td>U+0434</td>
<td>CYRILLIC SMALL LETTER DE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06C5</td>
<td>U+0435</td>
<td>CYRILLIC SMALL LETTER IE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06C6</td>
<td>U+0444</td>
<td>CYRILLIC SMALL LETTER EF</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06C7</td>
<td>U+0433</td>
<td>CYRILLIC SMALL LETTER GHE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06C8</td>
<td>U+0445</td>
<td>CYRILLIC SMALL LETTER HA</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06C9</td>
<td>U+0438</td>
<td>CYRILLIC SMALL LETTER I</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06CA</td>
<td>U+0439</td>
<td>CYRILLIC SMALL LETTER SHORT I</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06CB</td>
<td>U+043A</td>
<td>CYRILLIC SMALL LETTER KA</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06CC</td>
<td>U+043B</td>
<td>CYRILLIC SMALL LETTER EL</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06CD</td>
<td>U+043C</td>
<td>CYRILLIC SMALL LETTER EM</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06CE</td>
<td>U+043D</td>
<td>CYRILLIC SMALL LETTER EN</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06CF</td>
<td>U+043E</td>
<td>CYRILLIC SMALL LETTER O</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06D0</td>
<td>U+043F</td>
<td>CYRILLIC SMALL LETTER PE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06D1</td>
<td>U+044F</td>
<td>CYRILLIC SMALL LETTER YA</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06D2</td>
<td>U+0440</td>
<td>CYRILLIC SMALL LETTER ER</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06D3</td>
<td>U+0441</td>
<td>CYRILLIC SMALL LETTER ES</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06D4</td>
<td>U+0442</td>
<td>CYRILLIC SMALL LETTER TE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06D5</td>
<td>U+0443</td>
<td>CYRILLIC SMALL LETTER U</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06D6</td>
<td>U+0436</td>
<td>CYRILLIC SMALL LETTER ZHE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06D7</td>
<td>U+0432</td>
<td>CYRILLIC SMALL LETTER VE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06D8</td>
<td>U+044C</td>
<td>CYRILLIC SMALL LETTER SOFT SIGN</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06D9</td>
<td>U+044B</td>
<td>CYRILLIC SMALL LETTER YERU</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06DA</td>
<td>U+0437</td>
<td>CYRILLIC SMALL LETTER ZE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06DB</td>
<td>U+0448</td>
<td>CYRILLIC SMALL LETTER SHA</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06DC</td>
<td>U+044D</td>
<td>CYRILLIC SMALL LETTER E</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>KEYSYM value</td>
<td>Unicode value</td>
<td>Name</td>
<td>Set</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>-------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>#x06DD</td>
<td>U+0449</td>
<td>CYRILLIC SMALL LETTER SHCHA</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06DE</td>
<td>U+0447</td>
<td>CYRILLIC SMALL LETTER CHE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06DF</td>
<td>U+044A</td>
<td>CYRILLIC SMALL LETTER HARD SIGN</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06E0</td>
<td>U+042E</td>
<td>CYRILLIC CAPITAL LETTER YU</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06E1</td>
<td>U+0410</td>
<td>CYRILLIC CAPITAL LETTER A</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06E2</td>
<td>U+0411</td>
<td>CYRILLIC CAPITAL LETTER BE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06E3</td>
<td>U+0426</td>
<td>CYRILLIC CAPITAL LETTER TSE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06E4</td>
<td>U+0414</td>
<td>CYRILLIC CAPITAL LETTER DE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06E5</td>
<td>U+0415</td>
<td>CYRILLIC CAPITAL LETTER IE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06E6</td>
<td>U+0424</td>
<td>CYRILLIC CAPITAL LETTER EF</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06E7</td>
<td>U+0413</td>
<td>CYRILLIC CAPITAL LETTER GHE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06E8</td>
<td>U+0425</td>
<td>CYRILLIC CAPITAL LETTER HA</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06E9</td>
<td>U+0418</td>
<td>CYRILLIC CAPITAL LETTER I</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06EA</td>
<td>U+0419</td>
<td>CYRILLIC CAPITAL LETTER SHORT I</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06EB</td>
<td>U+041A</td>
<td>CYRILLIC CAPITAL LETTER KA</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06EC</td>
<td>U+041B</td>
<td>CYRILLIC CAPITAL LETTER EL</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06ED</td>
<td>U+041C</td>
<td>CYRILLIC CAPITAL LETTER EM</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06EE</td>
<td>U+041D</td>
<td>CYRILLIC CAPITAL LETTER EN</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06EF</td>
<td>U+041E</td>
<td>CYRILLIC CAPITAL LETTER O</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06F0</td>
<td>U+041F</td>
<td>CYRILLIC CAPITAL LETTER PE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06F1</td>
<td>U+0420</td>
<td>CYRILLIC CAPITAL LETTER ER</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06F2</td>
<td>U+0421</td>
<td>CYRILLIC CAPITAL LETTER ES</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06F3</td>
<td>U+0422</td>
<td>CYRILLIC CAPITAL LETTER TE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06F4</td>
<td>U+0423</td>
<td>CYRILLIC CAPITAL LETTER U</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06F5</td>
<td>U+0416</td>
<td>CYRILLIC CAPITAL LETTER ZHE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06F6</td>
<td>U+0412</td>
<td>CYRILLIC CAPITAL LETTER VE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06F8</td>
<td>U+042C</td>
<td>CYRILLIC CAPITAL LETTER SOFT SIGN</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06F9</td>
<td>U+042B</td>
<td>CYRILLIC CAPITAL LETTER YERU</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06FA</td>
<td>U+0417</td>
<td>CYRILLIC CAPITAL LETTER ZE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06FB</td>
<td>U+0428</td>
<td>CYRILLIC CAPITAL LETTER SHA</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06FC</td>
<td>U+042D</td>
<td>CYRILLIC CAPITAL LETTER E</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06FD</td>
<td>U+0429</td>
<td>CYRILLIC CAPITAL LETTER SHCHA</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06FE</td>
<td>U+0427</td>
<td>CYRILLIC CAPITAL LETTER CHE</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x06FF</td>
<td>U+042A</td>
<td>CYRILLIC CAPITAL LETTER HARD SIGN</td>
<td>Cyrillic</td>
</tr>
<tr>
<td>#x07A1</td>
<td>U+0386</td>
<td>GREEK CAPITAL LETTER ALPHA WITH TONOS</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07A2</td>
<td>U+0388</td>
<td>GREEK CAPITAL LETTER EPSILON WITH TONOS</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07A3</td>
<td>U+0389</td>
<td>GREEK CAPITAL LETTER ETA WITH TONOS</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07A4</td>
<td>U+038A</td>
<td>GREEK CAPITAL LETTER IOTA WITH TONOS</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07A5</td>
<td>U+038C</td>
<td>GREEK CAPITAL LETTER OMICRON WITH TONOS</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07A7</td>
<td>U+038E</td>
<td>GREEK CAPITAL LETTER UPSILON WITH TONOS</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07A9</td>
<td>U+03AB</td>
<td>GREEK CAPITAL LETTER UPSILON WITH DIA LYTIKA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07AA</td>
<td>U+038F</td>
<td>GREEK CAPITAL LETTER OMEGA WITH TONOS</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07AE</td>
<td>U+0385</td>
<td>GREEK DIALYTIKA TONOS</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07AF</td>
<td>U+2015</td>
<td>HORIZONTAL BAR</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07B1</td>
<td>U+03AC</td>
<td>GREEK SMALL LETTER ALPHA WITH TONOS</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07B2</td>
<td>U+03AD</td>
<td>GREEK SMALL LETTER EPSILON WITH TONOS</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07B3</td>
<td>U+03AE</td>
<td>GREEK SMALL LETTER ETA WITH TONOS</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07B4</td>
<td>U+03AF</td>
<td>GREEK SMALL LETTER IOTA WITH TONOS</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07B5</td>
<td>U+03CA</td>
<td>GREEK SMALL LETTER IOTA WITH DIA LYTIKA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07B6</td>
<td>U+0390</td>
<td>GREEK SMALL LETTER IOTA WITH DIALYTIKA AND TONOS</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07B7</td>
<td>U+03CC</td>
<td>GREEK SMALL LETTER OMICRON WITH TONOS</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07B8</td>
<td>U+03CD</td>
<td>GREEK SMALL LETTER UPSILON WITH TONOS</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07B9</td>
<td>U+03CB</td>
<td>GREEK SMALL LETTER UPSILON WITH DIALYTIKA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07BA</td>
<td>U+03B0</td>
<td>GREEK SMALL LETTER UPSILON WITH DIALYTIKA AND TONOS</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07BB</td>
<td>U+03CE</td>
<td>GREEK SMALL LETTER OMEGA WITH TONOS</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07C1</td>
<td>U+0391</td>
<td>GREEK CAPITAL LETTER ALPHA</td>
<td>Greek</td>
</tr>
<tr>
<td>KEYSYM value</td>
<td>Unicode value</td>
<td>Name</td>
<td>Set</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------</td>
<td>-------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>#x07C2</td>
<td>U+0392</td>
<td>GREEK CAPITAL LETTER BETA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07C3</td>
<td>U+0393</td>
<td>GREEK CAPITAL LETTER GAMMA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07C4</td>
<td>U+0394</td>
<td>GREEK CAPITAL LETTER DELTA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07C5</td>
<td>U+0395</td>
<td>GREEK CAPITAL LETTER EPSILON</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07C6</td>
<td>U+0396</td>
<td>GREEK CAPITAL LETTER ZETA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07C7</td>
<td>U+0397</td>
<td>GREEK CAPITAL LETTER ETA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07C8</td>
<td>U+0398</td>
<td>GREEK CAPITAL LETTER THETA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07C9</td>
<td>U+0399</td>
<td>GREEK CAPITAL LETTER IOTA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07CA</td>
<td>U+039A</td>
<td>GREEK CAPITAL LETTER KAPPA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07CB</td>
<td>U+039B</td>
<td>GREEK CAPITAL LETTER LAMDA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07CC</td>
<td>U+039C</td>
<td>GREEK CAPITAL LETTER MU</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07CD</td>
<td>U+039D</td>
<td>GREEK CAPITAL LETTER NU</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07CE</td>
<td>U+039E</td>
<td>GREEK CAPITAL LETTER XI</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07CF</td>
<td>U+039F</td>
<td>GREEK CAPITAL LETTER OMICRON</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07D0</td>
<td>U+03A0</td>
<td>GREEK CAPITAL LETTER PI</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07D1</td>
<td>U+03A1</td>
<td>GREEK CAPITAL LETTER RHO</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07D2</td>
<td>U+03A2</td>
<td>GREEK CAPITAL LETTER SIGMA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07D3</td>
<td>U+03A3</td>
<td>GREEK CAPITAL LETTER TAU</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07D4</td>
<td>U+03A4</td>
<td>GREEK CAPITAL LETTER UPSILON</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07D5</td>
<td>U+03A5</td>
<td>GREEK CAPITAL LETTER PSI</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07D6</td>
<td>U+03A6</td>
<td>GREEK CAPITAL LETTER PHI</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07D7</td>
<td>U+03A7</td>
<td>GREEK CAPITAL LETTER ETA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07D8</td>
<td>U+03A8</td>
<td>GREEK CAPITAL LETTER ETA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07D9</td>
<td>U+03A9</td>
<td>GREEK CAPITAL LETTER OMEGA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07E1</td>
<td>U+03B1</td>
<td>GREEK SMALL LETTER ALPHA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07E2</td>
<td>U+03B2</td>
<td>GREEK SMALL LETTER BETA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07E3</td>
<td>U+03B3</td>
<td>GREEK SMALL LETTER GAMMA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07E4</td>
<td>U+03B4</td>
<td>GREEK SMALL LETTER DELTA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07E5</td>
<td>U+03B5</td>
<td>GREEK SMALL LETTER EPSILON</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07E6</td>
<td>U+03B6</td>
<td>GREEK SMALL LETTER ZETA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07E7</td>
<td>U+03B7</td>
<td>GREEK SMALL LETTER ETA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07E8</td>
<td>U+03B8</td>
<td>GREEK SMALL LETTER THETA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07E9</td>
<td>U+03B9</td>
<td>GREEK SMALL LETTER IOTA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07EA</td>
<td>U+03BA</td>
<td>GREEK SMALL LETTER KAPPA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07EB</td>
<td>U+03BB</td>
<td>GREEK SMALL LETTER LAMDA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07EC</td>
<td>U+03BC</td>
<td>GREEK SMALL LETTER MU</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07ED</td>
<td>U+03BD</td>
<td>GREEK SMALL LETTER NU</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07EE</td>
<td>U+03BE</td>
<td>GREEK SMALL LETTER XI</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07EF</td>
<td>U+03BF</td>
<td>GREEK SMALL LETTER OMICRON</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07F0</td>
<td>U+03C0</td>
<td>GREEK SMALL LETTER PI</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07F1</td>
<td>U+03C1</td>
<td>GREEK SMALL LETTER RHO</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07F2</td>
<td>U+03C2</td>
<td>GREEK SMALL LETTER SIGMA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07F3</td>
<td>U+03C3</td>
<td>GREEK SMALL LETTER FINAL SIGMA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07F4</td>
<td>U+03C4</td>
<td>GREEK SMALL LETTER TAU</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07F5</td>
<td>U+03C5</td>
<td>GREEK SMALL LETTER UPSILON</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07F6</td>
<td>U+03C6</td>
<td>GREEK SMALL LETTER PHI</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07F7</td>
<td>U+03C7</td>
<td>GREEK SMALL LETTER CHI</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07F8</td>
<td>U+03C8</td>
<td>GREEK SMALL LETTER PSI</td>
<td>Greek</td>
</tr>
<tr>
<td>#x07F9</td>
<td>U+03C9</td>
<td>GREEK SMALL LETTER OMEGA</td>
<td>Greek</td>
</tr>
<tr>
<td>#x08A1</td>
<td>U+23B7</td>
<td>LEFT RADICAL</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08A2</td>
<td>–</td>
<td>TOP LEFT RADICAL</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08A3</td>
<td>–</td>
<td>HORIZONTAL CONNECTOR</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08A4</td>
<td>U+2320</td>
<td>TOP INTEGRAL</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08A5</td>
<td>U+2321</td>
<td>BOTTOM INTEGRAL</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08A6</td>
<td>–</td>
<td>VERTICAL CONNECTOR</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08A7</td>
<td>U+23A1</td>
<td>TOP LEFT SQUARE BRACKET</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08A8</td>
<td>U+23A3</td>
<td>BOTTOM LEFT SQUARE BRACKET</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08A9</td>
<td>U+23A4</td>
<td>TOP RIGHT SQUARE BRACKET</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08AA</td>
<td>U+23A6</td>
<td>BOTTOM RIGHT SQUARE BRACKET</td>
<td>Technical</td>
</tr>
<tr>
<td>KEYSYM value</td>
<td>Unicode value</td>
<td>Name</td>
<td>Set</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------</td>
<td>------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>#x08AB</td>
<td>U+239B</td>
<td>TOP LEFT PARENTHESIS</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08AC</td>
<td>U+239D</td>
<td>BOTTOM LEFT PARENTHESIS</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08AD</td>
<td>U+239E</td>
<td>TOP RIGHT PARENTHESIS</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08AE</td>
<td>U+23A0</td>
<td>BOTTOM RIGHT PARENTHESIS</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08AF</td>
<td>U+23A8</td>
<td>LEFT MIDDLE CURLY BRACE</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08B0</td>
<td>U+23AC</td>
<td>RIGHT MIDDLE CURLY BRACE</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08B1</td>
<td></td>
<td>TOP LEFT SUMMATION</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08B2</td>
<td></td>
<td>BOTTOM LEFT SUMMATION</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08B3</td>
<td></td>
<td>TOP VERTICAL SUMMATION Connector</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08B4</td>
<td></td>
<td>BOTTOM VERTICAL SUMMATION Connector</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08B5</td>
<td></td>
<td>TOP RIGHT SUMMATION</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08B6</td>
<td></td>
<td>BOTTOM RIGHT SUMMATION</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08B7</td>
<td></td>
<td>RIGHT MIDDLE SUMMATION</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08BC</td>
<td>U+2264</td>
<td>LESS THAN OR EQUAL SIGN</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08BD</td>
<td>U+2260</td>
<td>NOT EQUAL SIGN</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08BE</td>
<td>U+2265</td>
<td>GREATER THAN OR EQUAL SIGN</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08BF</td>
<td>U+222B</td>
<td>INTEGRAL</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08C0</td>
<td>U+2234</td>
<td>THEREFORE</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08C1</td>
<td>U+221D</td>
<td>VARIATION, PROPORTIONAL TO</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08C2</td>
<td>U+221E</td>
<td>INFINITY</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08C5</td>
<td>U+2207</td>
<td>NABLA, DEL</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08C8</td>
<td>U+223C</td>
<td>IS APPROXIMATE TO</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08C9</td>
<td>U+2243</td>
<td>SIMILAR OR EQUAL TO</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08CD</td>
<td>U+21D4</td>
<td>IF AND ONLY IF</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08CE</td>
<td>U+21D2</td>
<td>IMPLIES</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08CF</td>
<td>U+2261</td>
<td>IDENTICAL TO</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08D6</td>
<td>U+221A</td>
<td>RADICAL</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08DA</td>
<td>U+2282</td>
<td>IS INCLUDED IN</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08DB</td>
<td>U+2283</td>
<td>INCLUDES</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08DC</td>
<td>U+2229</td>
<td>INTERSECTION</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08DD</td>
<td>U+222A</td>
<td>UNION</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08DE</td>
<td>U+2227</td>
<td>LOGICAL AND</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08DF</td>
<td>U+2228</td>
<td>LOGICAL OR</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08EF</td>
<td>U+2202</td>
<td>PARTIAL DERIVATIVE</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08F6</td>
<td>U+0192</td>
<td>FUNCTION</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08FB</td>
<td>U+2190</td>
<td>LEFT ARROW</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08FC</td>
<td>U+2191</td>
<td>UPWARD ARROW</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08FD</td>
<td>U+2192</td>
<td>RIGHT ARROW</td>
<td>Technical</td>
</tr>
<tr>
<td>#x08FE</td>
<td>U+2193</td>
<td>DOWNWARD ARROW</td>
<td>Technical</td>
</tr>
<tr>
<td>#x09DF</td>
<td></td>
<td>BLANK</td>
<td>Special</td>
</tr>
<tr>
<td>#x09E0</td>
<td>U+25C6</td>
<td>SOLID DIAMOND</td>
<td>Special</td>
</tr>
<tr>
<td>#x09E1</td>
<td>U+2592</td>
<td>CHECKERBOARD</td>
<td>Special</td>
</tr>
<tr>
<td>#x09E2</td>
<td>U+2409</td>
<td>“HT”</td>
<td>Special</td>
</tr>
<tr>
<td>#x09E3</td>
<td>U+240C</td>
<td>“FF”</td>
<td>Special</td>
</tr>
<tr>
<td>#x09E4</td>
<td>U+240D</td>
<td>“CR”</td>
<td>Special</td>
</tr>
<tr>
<td>#x09E5</td>
<td>U+240A</td>
<td>“LF”</td>
<td>Special</td>
</tr>
<tr>
<td>#x09E8</td>
<td>U+2424</td>
<td>“NL”</td>
<td>Special</td>
</tr>
<tr>
<td>#x09E9</td>
<td>U+240B</td>
<td>“VT”</td>
<td>Special</td>
</tr>
<tr>
<td>#x09EA</td>
<td>U+2518</td>
<td>LOWER-RIGHT CORNER</td>
<td>Special</td>
</tr>
<tr>
<td>#x09EB</td>
<td>U+2510</td>
<td>UPPER-RIGHT CORNER</td>
<td>Special</td>
</tr>
<tr>
<td>#x09EC</td>
<td>U+250C</td>
<td>UPPER-LEFT CORNER</td>
<td>Special</td>
</tr>
<tr>
<td>#x09ED</td>
<td>U+2514</td>
<td>LOWER-LEFT CORNER</td>
<td>Special</td>
</tr>
<tr>
<td>#x09EE</td>
<td>U+253C</td>
<td>CROSSING-LINES</td>
<td>Special</td>
</tr>
<tr>
<td>#x09EF</td>
<td>U+23BA</td>
<td>HORIZONTAL LINE, SCAN 1</td>
<td>Special</td>
</tr>
<tr>
<td>#x09F0</td>
<td>U+23BB</td>
<td>HORIZONTAL LINE, SCAN 3</td>
<td>Special</td>
</tr>
<tr>
<td>#x09F1</td>
<td>U+2500</td>
<td>HORIZONTAL LINE, SCAN 5</td>
<td>Special</td>
</tr>
<tr>
<td>#x09F2</td>
<td>U+23BC</td>
<td>HORIZONTAL LINE, SCAN 7</td>
<td>Special</td>
</tr>
<tr>
<td>#x09F3</td>
<td>U+23BD</td>
<td>HORIZONTAL LINE, SCAN 9</td>
<td>Special</td>
</tr>
<tr>
<td>KEYSYM value</td>
<td>Unicode value</td>
<td>Name</td>
<td>Set</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td>#x09F4</td>
<td>U+251C</td>
<td>LEFT “T”</td>
<td>Special</td>
</tr>
<tr>
<td>#x09F5</td>
<td>U+2524</td>
<td>RIGHT “T”</td>
<td>Special</td>
</tr>
<tr>
<td>#x09F6</td>
<td>U+2534</td>
<td>BOTTOM “T”</td>
<td>Special</td>
</tr>
<tr>
<td>#x09F7</td>
<td>U+252C</td>
<td>TOP “T”</td>
<td>Special</td>
</tr>
<tr>
<td>#x09F8</td>
<td>U+2502</td>
<td>VERTICAL BAR</td>
<td>Special</td>
</tr>
<tr>
<td>#x0AA1</td>
<td>U+2003</td>
<td>EM SPACE</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AA2</td>
<td>U+2002</td>
<td>EN SPACE</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AA3</td>
<td>U+2004</td>
<td>3/EM SPACE</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AA4</td>
<td>U+2005</td>
<td>4/EM SPACE</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AA5</td>
<td>U+2007</td>
<td>DIGIT SPACE</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AA6</td>
<td>U+2008</td>
<td>PUNCTUATION SPACE</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AA7</td>
<td>U+2009</td>
<td>THIN SPACE</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AA8</td>
<td>U+200A</td>
<td>HAIR SPACE</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AA9</td>
<td>U+2014</td>
<td>EM DASH</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AAA</td>
<td>U+2013</td>
<td>EN DASH</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AAC</td>
<td>–</td>
<td>SIGNIFICANT BLANK SYMBOL</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AAE</td>
<td>U+2026</td>
<td>ELLIPSIS</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AAF</td>
<td>U+2025</td>
<td>DOUBLE BASELINE DOT</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AB0</td>
<td>U+2153</td>
<td>VULGAR FRACTION ONE THIRD</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AB1</td>
<td>U+2154</td>
<td>VULGAR FRACTION TWO THIRDS</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AB2</td>
<td>U+2155</td>
<td>VULGAR FRACTION ONE FIFTH</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AB3</td>
<td>U+2156</td>
<td>VULGAR FRACTION TWO FIFTHS</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AB4</td>
<td>U+2157</td>
<td>VULGAR FRACTION THREE FIFTHS</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AB5</td>
<td>U+2158</td>
<td>VULGAR FRACTION FOUR FIFTHS</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AB6</td>
<td>U+2159</td>
<td>VULGAR FRACTION ONE SIXTH</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AB7</td>
<td>U+215A</td>
<td>VULGAR FRACTION FIVE SIXTHS</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AB8</td>
<td>U+2105</td>
<td>CARE OF</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AB9</td>
<td>U+2012</td>
<td>FIGURE DASH</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0ABC</td>
<td>–</td>
<td>LEFT ANGLE BRACKET</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0ABD</td>
<td>–</td>
<td>DECIMAL POINT</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0ABE</td>
<td>–</td>
<td>RIGHT ANGLE BRACKET</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0ABF</td>
<td>–</td>
<td>MARKER</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AC3</td>
<td>U+215B</td>
<td>VULGAR FRACTION ONE EIGHTH</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AC4</td>
<td>U+215C</td>
<td>VULGAR FRACTION THREE EIGHTHS</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AC5</td>
<td>U+215D</td>
<td>VULGAR FRACTION FIVE EIGHTHS</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AC6</td>
<td>U+215E</td>
<td>VULGAR FRACTION SEVEN EIGHTHS</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AC9</td>
<td>U+2122</td>
<td>TRADEMARK SIGN</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0ACA</td>
<td>–</td>
<td>SIGNATURE MARK</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0ACB</td>
<td>–</td>
<td>TRADEMARK SIGN IN CIRCLE</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0ACC</td>
<td>–</td>
<td>LEFT OPEN TRIANGLE</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0ACD</td>
<td>–</td>
<td>RIGHT OPEN TRIANGLE</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0ACE</td>
<td>–</td>
<td>EM OPEN CIRCLE</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0ACF</td>
<td>–</td>
<td>EM OPEN RECTANGLE</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AD0</td>
<td>U+2018</td>
<td>LEFT SINGLE QUOTATION MARK</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AD1</td>
<td>U+2019</td>
<td>RIGHT SINGLE QUOTATION MARK</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AD2</td>
<td>U+201C</td>
<td>LEFT DOUBLE QUOTATION MARK</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AD3</td>
<td>U+201D</td>
<td>RIGHT DOUBLE QUOTATION MARK</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AD4</td>
<td>U+211E</td>
<td>PRESCRIPTION, TAKE, RECIPE</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AD6</td>
<td>U+2032</td>
<td>MINUTES</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AD7</td>
<td>U+2033</td>
<td>SECONDS</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AD9</td>
<td>U+271D</td>
<td>LATIN CROSS</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0ADA</td>
<td>–</td>
<td>HEXAGRAM</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0ADB</td>
<td>–</td>
<td>FILLED RECTANGLE BULLET</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0ADC</td>
<td>–</td>
<td>FILLED LEFT TRIANGLE BULLET</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0ADD</td>
<td>–</td>
<td>FILLED RIGHT TRIANGLE BULLET</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0ADE</td>
<td>–</td>
<td>EM FILLED CIRCLE</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0ADF</td>
<td>–</td>
<td>EM FILLED RECTANGLE</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AE0</td>
<td>–</td>
<td>EN OPEN CIRCLE BULLET</td>
<td>Publish</td>
</tr>
<tr>
<td>KEYSYM value</td>
<td>Unicode value</td>
<td>Name</td>
<td>Set</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------</td>
<td>-----------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>#x0AE1</td>
<td></td>
<td>EN OPEN SQUARE BULLET</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AE2</td>
<td></td>
<td>OPEN RECTANGULAR BULLET</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AE3</td>
<td></td>
<td>OPEN TRIANGULAR BULLET UP</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AE4</td>
<td></td>
<td>OPEN TRIANGULAR BULLET DOWN</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AE5</td>
<td></td>
<td>OPEN STAR</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AE6</td>
<td></td>
<td>EN FILLED CIRCLE BULLET</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AE7</td>
<td></td>
<td>EN FILLED SQUARE BULLET</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AE8</td>
<td></td>
<td>FILLED TRIANGULAR BULLET UP</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AE9</td>
<td></td>
<td>FILLED TRIANGULAR BULLET DOWN</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AEa</td>
<td></td>
<td>LEFT POINTER</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AEB</td>
<td></td>
<td>RIGHT POINTER</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AEC</td>
<td>U+2663</td>
<td>CLUB</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AEd</td>
<td>U+2666</td>
<td>DIAMOND</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0Aee</td>
<td>U+2665</td>
<td>HEART</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0Af0</td>
<td>U+2720</td>
<td>MALTESE CROSS</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0Af1</td>
<td>U+2020</td>
<td>DAGGER</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0Af2</td>
<td>U+2021</td>
<td>DOUBLE DAGGER</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0Af3</td>
<td>U+2713</td>
<td>CHECK MARK, TICK</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0Af4</td>
<td>U+2717</td>
<td>BALLOT CROSS</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0Af5</td>
<td>U+266F</td>
<td>MUSICAL SHARP</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0Af6</td>
<td>U+266D</td>
<td>MUSICAL FLAT</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0Af7</td>
<td>U+2642</td>
<td>MALE SYMBOL</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0Af8</td>
<td>U+2640</td>
<td>FEMALE SYMBOL</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0Af9</td>
<td>U+260E</td>
<td>TELEPHONE SYMBOL</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AfA</td>
<td>U+2315</td>
<td>TELEPHONE RECORDER SYMBOL</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AfB</td>
<td>U+2117</td>
<td>PHONOGRAPH COPYRIGHT SIGN</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AfC</td>
<td>U+2038</td>
<td>CARET</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AfD</td>
<td>U+201A</td>
<td>SINGLE LOW QUOTATION MARK</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AfE</td>
<td>U+201E</td>
<td>DOUBLE LOW QUOTATION MARK</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0AFF</td>
<td></td>
<td>CURSOR</td>
<td>Publish</td>
</tr>
<tr>
<td>#x0Ba3</td>
<td></td>
<td>LEFT CARET</td>
<td>APL</td>
</tr>
<tr>
<td>#x0Ba6</td>
<td></td>
<td>RIGHT CARET</td>
<td>APL</td>
</tr>
<tr>
<td>#x0Ba8</td>
<td></td>
<td>DOWN CARET</td>
<td>APL</td>
</tr>
<tr>
<td>#x0Ba9</td>
<td></td>
<td>UP CARET</td>
<td>APL</td>
</tr>
<tr>
<td>#x0Bc0</td>
<td></td>
<td>OVERBAR</td>
<td>APL</td>
</tr>
<tr>
<td>#x0Bc2</td>
<td>U+22A5</td>
<td>DOWN TACK</td>
<td>APL</td>
</tr>
<tr>
<td>#x0Bc3</td>
<td></td>
<td>UP SHOE (CAP)</td>
<td>APL</td>
</tr>
<tr>
<td>#x0Bc4</td>
<td>U+230A</td>
<td>DOWN STILE</td>
<td>APL</td>
</tr>
<tr>
<td>#x0Bc6</td>
<td></td>
<td>UNDERBAR</td>
<td>APL</td>
</tr>
<tr>
<td>#x0BcA</td>
<td>U+2218</td>
<td>JOT</td>
<td>APL</td>
</tr>
<tr>
<td>#x0Bcc</td>
<td>U+2395</td>
<td>QUAD</td>
<td>APL</td>
</tr>
<tr>
<td>#x0Bce</td>
<td>U+22A4</td>
<td>UP TACK</td>
<td>APL</td>
</tr>
<tr>
<td>#x0Bcf</td>
<td>U+25CB</td>
<td>CIRCLE</td>
<td>APL</td>
</tr>
<tr>
<td>#x0Bd3</td>
<td>U+2308</td>
<td>UP STILE</td>
<td>APL</td>
</tr>
<tr>
<td>#x0Bd6</td>
<td></td>
<td>DOWN SHOE (CUP)</td>
<td>APL</td>
</tr>
<tr>
<td>#x0Bd8</td>
<td></td>
<td>RIGHT SHOE</td>
<td>APL</td>
</tr>
<tr>
<td>#x0Bda</td>
<td></td>
<td>LEFT SHOE</td>
<td>APL</td>
</tr>
<tr>
<td>#x0Bdc</td>
<td>U+22A2</td>
<td>LEFT TACK</td>
<td>APL</td>
</tr>
<tr>
<td>#x0Bfc</td>
<td>U+22A3</td>
<td>RIGHT TACK</td>
<td>APL</td>
</tr>
<tr>
<td>#x0CDF</td>
<td>U+2017</td>
<td>DOUBLE LOW LINE</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0Ce0</td>
<td>U+05D0</td>
<td>HEBREW LETTER ALEF</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0Ce1</td>
<td>U+05D1</td>
<td>HEBREW LETTER BET</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0Ce2</td>
<td>U+05D2</td>
<td>HEBREW LETTER GIMEL</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0Ce3</td>
<td>U+05D3</td>
<td>HEBREW LETTER DALET</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0Ce4</td>
<td>U+05D4</td>
<td>HEBREW LETTER HE</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0Ce5</td>
<td>U+05D5</td>
<td>HEBREW LETTER VAV</td>
<td>Hebrew</td>
</tr>
</tbody>
</table>

105
<table>
<thead>
<tr>
<th>KEYSYM value</th>
<th>Unicode value</th>
<th>Name</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>#x0CE6</td>
<td>U+05D6</td>
<td>HEBREW LETTER ZAYIN</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CE7</td>
<td>U+05D7</td>
<td>HEBREW LETTER HET</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CE8</td>
<td>U+05D8</td>
<td>HEBREW LETTER TET</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CE9</td>
<td>U+05D9</td>
<td>HEBREW LETTER YOD</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CEA</td>
<td>U+05DA</td>
<td>HEBREW LETTER FINAL KAF</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CEB</td>
<td>U+05DB</td>
<td>HEBREW LETTER KAF</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CEC</td>
<td>U+05DC</td>
<td>HEBREW LETTER LAMED</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CED</td>
<td>U+05DD</td>
<td>HEBREW LETTER FINAL MEM</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CEE</td>
<td>U+05DE</td>
<td>HEBREW LETTER MEM</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CEF</td>
<td>U+05DF</td>
<td>HEBREW LETTER FINAL NUN</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CF0</td>
<td>U+05E0</td>
<td>HEBREW LETTER NUN</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CF1</td>
<td>U+05E1</td>
<td>HEBREW LETTER SAMEKH</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CF2</td>
<td>U+05E2</td>
<td>HEBREW LETTER AYIN</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CF3</td>
<td>U+05E3</td>
<td>HEBREW LETTER FINAL PE</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CF4</td>
<td>U+05E4</td>
<td>HEBREW LETTER PE</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CF5</td>
<td>U+05E5</td>
<td>HEBREW LETTER FINAL TSADI</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CF6</td>
<td>U+05E6</td>
<td>HEBREW LETTER TSADI</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CF7</td>
<td>U+05E7</td>
<td>HEBREW LETTER QOF</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CF8</td>
<td>U+05E8</td>
<td>HEBREW LETTER RESH</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CF9</td>
<td>U+05E9</td>
<td>HEBREW LETTER SHIN</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0CFA</td>
<td>U+05EA</td>
<td>HEBREW LETTER TAV</td>
<td>Hebrew</td>
</tr>
<tr>
<td>#x0DA1</td>
<td>U+0E01</td>
<td>THAI CHARACTER KO KAI</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DA2</td>
<td>U+0E02</td>
<td>THAI CHARACTER KO KHAI</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DA3</td>
<td>U+0E03</td>
<td>THAI CHARACTER KO KHUAT</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DA4</td>
<td>U+0E04</td>
<td>THAI CHARACTER KHO KHWAI</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DA5</td>
<td>U+0E05</td>
<td>THAI CHARACTER KHO KHON</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DA6</td>
<td>U+0E06</td>
<td>THAI CHARACTER KHO RAKHANG</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DA7</td>
<td>U+0E07</td>
<td>THAI CHARACTER NGO NGU</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DA8</td>
<td>U+0E08</td>
<td>THAI CHARACTER CHO CHAN</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DA9</td>
<td>U+0E09</td>
<td>THAI CHARACTER CHO CHING</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DAA</td>
<td>U+0E0A</td>
<td>THAI CHARACTER CHO CHANG</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DAB</td>
<td>U+0E0B</td>
<td>THAI CHARACTER SO SO</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DAC</td>
<td>U+0E0C</td>
<td>THAI CHARACTER CHO CHOE</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DAD</td>
<td>U+0E0D</td>
<td>THAI CHARACTER YO YING</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DAE</td>
<td>U+0E0E</td>
<td>THAI CHARACTER DO CHADA</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DAF</td>
<td>U+0E0F</td>
<td>THAI CHARACTER TO PAKA</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DB0</td>
<td>U+0E10</td>
<td>THAI CHARACTER THO THAN</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DB1</td>
<td>U+0E11</td>
<td>THAI CHARACTER THO NANGMONTHO</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DB2</td>
<td>U+0E12</td>
<td>THAI CHARACTER THO PHUTHAO</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DB3</td>
<td>U+0E13</td>
<td>THAI CHARACTER NO NEN</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DB4</td>
<td>U+0E14</td>
<td>THAI CHARACTER DO DEK</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DB5</td>
<td>U+0E15</td>
<td>THAI CHARACTER TO TAO</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DB6</td>
<td>U+0E16</td>
<td>THAI CHARACTER THO THUNG</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DB7</td>
<td>U+0E17</td>
<td>THAI CHARACTER THO THAHAN</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DB8</td>
<td>U+0E18</td>
<td>THAI CHARACTER THO THONG</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DB9</td>
<td>U+0E19</td>
<td>THAI CHARACTER NO NU</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DBA</td>
<td>U+0E1A</td>
<td>THAI CHARACTER BO BAIMAI</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DBB</td>
<td>U+0E1B</td>
<td>THAI CHARACTER PO PLA</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DBC</td>
<td>U+0E1C</td>
<td>THAI CHARACTER PHO PHUNG</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DBD</td>
<td>U+0E1D</td>
<td>THAI CHARACTER FO FA</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DBE</td>
<td>U+0E1E</td>
<td>THAI CHARACTER PHO PHAN</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DBF</td>
<td>U+0E1F</td>
<td>THAI CHARACTER FO FAN</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DC0</td>
<td>U+0E20</td>
<td>THAI CHARACTER PHO SAMPHAO</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DC1</td>
<td>U+0E21</td>
<td>THAI CHARACTER MO MA</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DC2</td>
<td>U+0E22</td>
<td>THAI CHARACTER YO YAK</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DC3</td>
<td>U+0E23</td>
<td>THAI CHARACTER RO RUA</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DC4</td>
<td>U+0E24</td>
<td>THAI CHARACTER RU</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DC5</td>
<td>U+0E25</td>
<td>THAI CHARACTER LO LING</td>
<td>Thai</td>
</tr>
<tr>
<td>KEYSYM value</td>
<td>Unicode value</td>
<td>Name</td>
<td>Set</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>-------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>#x0DC6</td>
<td>U+0E26</td>
<td>THAI CHARACTER LU</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DC7</td>
<td>U+0E27</td>
<td>THAI CHARACTER WO WAEN</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DC8</td>
<td>U+0E28</td>
<td>THAI CHARACTER SO SALA</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DC9</td>
<td>U+0E29</td>
<td>THAI CHARACTER SO RUSI</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DCA</td>
<td>U+0E2A</td>
<td>THAI CHARACTER SO SUA</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DCB</td>
<td>U+0E2B</td>
<td>THAI CHARACTER HO HIP</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DCC</td>
<td>U+0E2C</td>
<td>THAI CHARACTER LO CHULA</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0CDC</td>
<td>U+0E2D</td>
<td>THAI CHARACTER O ANG</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DCE</td>
<td>U+0E2E</td>
<td>THAI CHARACTER HO NOKHUK</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DCF</td>
<td>U+0E2F</td>
<td>THAI CHARACTER PAIYANNOI</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DD0</td>
<td>U+0E30</td>
<td>THAI CHARACTER SARA A</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DD1</td>
<td>U+0E31</td>
<td>THAI CHARACTER MAI HAN-AKAT</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DD2</td>
<td>U+0E32</td>
<td>THAI CHARACTER SARA AA</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DD3</td>
<td>U+0E33</td>
<td>THAI CHARACTER SARA AM</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DD4</td>
<td>U+0E34</td>
<td>THAI CHARACTER SARA I</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DD5</td>
<td>U+0E35</td>
<td>THAI CHARACTER SARA II</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DD6</td>
<td>U+0E36</td>
<td>THAI CHARACTER SARA UE</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DD7</td>
<td>U+0E37</td>
<td>THAI CHARACTER SARA UEE</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DD8</td>
<td>U+0E38</td>
<td>THAI CHARACTER SARA U</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DD9</td>
<td>U+0E39</td>
<td>THAI CHARACTER SARA UU</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DDA</td>
<td>U+0E3A</td>
<td>THAI CHARACTER PHINTHU</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DDF</td>
<td>U+0E3F</td>
<td>THAI CURRENCY SYMBOL BAHT</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DE0</td>
<td>U+0E40</td>
<td>THAI CHARACTER SARA E</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DE1</td>
<td>U+0E41</td>
<td>THAI CHARACTER SARA AE</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DE2</td>
<td>U+0E42</td>
<td>THAI CHARACTER SARA O</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DE3</td>
<td>U+0E43</td>
<td>THAI CHARACTER SARA AI MAIMUAN</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DE4</td>
<td>U+0E44</td>
<td>THAI CHARACTER SARA AI MAIMALAI</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DE5</td>
<td>U+0E45</td>
<td>THAI CHARACTER LAKHANGYAO</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DE6</td>
<td>U+0E46</td>
<td>THAI CHARACTER MAIYAMOK</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DE7</td>
<td>U+0E47</td>
<td>THAI CHARACTER MAITAIKHU</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DE8</td>
<td>U+0E48</td>
<td>THAI CHARACTER MAI EK</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DE9</td>
<td>U+0E49</td>
<td>THAI CHARACTER MAI THO</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DEA</td>
<td>U+0E4A</td>
<td>THAI CHARACTER MAI TRI</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DEB</td>
<td>U+0E4B</td>
<td>THAI CHARACTER MAI CHATTAWA</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DEC</td>
<td>U+0E4C</td>
<td>THAI CHARACTER THANTHAKHAT</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DED</td>
<td>U+0E4D</td>
<td>THAI CHARACTER NIKHAHIT</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DF0</td>
<td>U+0E50</td>
<td>THAI DIGIT ZERO</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DF1</td>
<td>U+0E51</td>
<td>THAI DIGIT ONE</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DF2</td>
<td>U+0E52</td>
<td>THAI DIGIT TWO</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DF3</td>
<td>U+0E53</td>
<td>THAI DIGIT THREE</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DF4</td>
<td>U+0E54</td>
<td>THAI DIGIT FOUR</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DF5</td>
<td>U+0E55</td>
<td>THAI DIGIT FIVE</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DF6</td>
<td>U+0E56</td>
<td>THAI DIGIT SIX</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DF7</td>
<td>U+0E57</td>
<td>THAI DIGIT SEVEN</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DF8</td>
<td>U+0E58</td>
<td>THAI DIGIT EIGHT</td>
<td>Thai</td>
</tr>
<tr>
<td>#x0DF9</td>
<td>U+0E59</td>
<td>THAI DIGIT NINE</td>
<td>Thai</td>
</tr>
</tbody>
</table>

- #x0EA1: HANGUL KIYEONG
- #x0EA2: HANGUL SSANG KIYEONG
- #x0EA3: HANGUL SSANG KIYEONG SIOS
- #x0EA4: HANGUL NIEUN
- #x0EA5: HANGUL NIEUN JIEUJ
- #x0EA6: HANGUL NIEUN HIEUH
- #x0EA7: HANGUL DIKEUD
- #x0EA8: HANGUL SSANG DIKEUD
- #x0EA9: HANGUL RIEUL
- #x0EAA: HANGUL RIEUL KIYEONG
- #x0EAB: HANGUL RIEUL MIEUM
- #x0EAC: HANGUL RIEUL PIEUB

Korean
<table>
<thead>
<tr>
<th>KEYSYM value</th>
<th>Unicode value</th>
<th>Name</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>\x0EAD</td>
<td></td>
<td>HANGUL RIEUL SIOS</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EAE</td>
<td></td>
<td>HANGUL RIEUL TIEUT</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EAF</td>
<td></td>
<td>HANGUL RIEUL PHIEUF</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EB0</td>
<td></td>
<td>HANGUL RIEUL HIEUH</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EB1</td>
<td></td>
<td>HANGUL MIEUM</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EB2</td>
<td></td>
<td>HANGUL PIEUB</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EB3</td>
<td></td>
<td>HANGUL SSANG PIEUB</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EB4</td>
<td></td>
<td>HANGUL PIEUB SIOS</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EB5</td>
<td></td>
<td>HANGUL SIOS</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EB6</td>
<td></td>
<td>HANGUL SSANG SIOS</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EB7</td>
<td></td>
<td>HANGUL IEUNG</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EB8</td>
<td></td>
<td>HANGUL JIEUJ</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EB9</td>
<td></td>
<td>HANGUL SSANG JIEUJ</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EBA</td>
<td></td>
<td>HANGUL CIEUC</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EBB</td>
<td></td>
<td>HANGUL KIEUQ</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EBC</td>
<td></td>
<td>HANGUL TIEUT</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EBD</td>
<td></td>
<td>HANGUL PHIEUF</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EBE</td>
<td></td>
<td>HANGUL HIEUH</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EBF</td>
<td></td>
<td>HANGUL A</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EC0</td>
<td></td>
<td>HANGUL AE</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EC1</td>
<td></td>
<td>HANGUL YA</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EC2</td>
<td></td>
<td>HANGUL YAE</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EC3</td>
<td></td>
<td>HANGUL EO</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EC4</td>
<td></td>
<td>HANGUL E</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EC5</td>
<td></td>
<td>HANGUL YEO</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EC6</td>
<td></td>
<td>HANGUL YE</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EC7</td>
<td></td>
<td>HANGUL O</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EC8</td>
<td></td>
<td>HANGUL WA</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EC9</td>
<td></td>
<td>HANGUL WAE</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0ECA</td>
<td></td>
<td>HANGUL OE</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0ECB</td>
<td></td>
<td>HANGUL YO</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0ECC</td>
<td></td>
<td>HANGUL U</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0ECD</td>
<td></td>
<td>HANGUL WEO</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0ECE</td>
<td></td>
<td>HANGUL WE</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0ECF</td>
<td></td>
<td>HANGUL WI</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0ED0</td>
<td></td>
<td>HANGUL YU</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0ED1</td>
<td></td>
<td>HANGUL EU</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0ED2</td>
<td></td>
<td>HANGUL YI</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0ED3</td>
<td></td>
<td>HANGUL I</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0ED4</td>
<td></td>
<td>HANGUL JONG SEONG KIYEOG</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0ED5</td>
<td></td>
<td>HANGUL JONG SEONG SSANG KIYEOG</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0ED6</td>
<td></td>
<td>HANGUL JONG SEONG KIYEOG SIOS</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0ED7</td>
<td></td>
<td>HANGUL JONG SEONG NIEUN</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0ED8</td>
<td></td>
<td>HANGUL JONG SEONG NIEUN JIEUJ</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0ED9</td>
<td></td>
<td>HANGUL JONG SEONG NIEUN HIEUH</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EDA</td>
<td></td>
<td>HANGUL JONG SEONG DIKEUD</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EDB</td>
<td></td>
<td>HANGUL JONG SEONG RIEUL</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EDC</td>
<td></td>
<td>HANGUL JONG SEONG RIEUL KIYEOG</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EDD</td>
<td></td>
<td>HANGUL JONG SEONG RIEUL MIEUM</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EDE</td>
<td></td>
<td>HANGUL JONG SEONG RIEUL PIEUB</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EDF</td>
<td></td>
<td>HANGUL JONG SEONG RIEUL SIOS</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EE0</td>
<td></td>
<td>HANGUL JONG SEONG RIEUL TIEUT</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EE1</td>
<td></td>
<td>HANGUL JONG SEONG RIEUL PHIEUF</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EE2</td>
<td></td>
<td>HANGUL JONG SEONG RIEUL HIEUH</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EE3</td>
<td></td>
<td>HANGUL JONG SEONG MIEUM</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EE4</td>
<td></td>
<td>HANGUL JONG SEONG PIEUB</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EE5</td>
<td></td>
<td>HANGUL JONG SEONG PIEUB SIOS</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EE6</td>
<td></td>
<td>HANGUL JONG SEONG SIOS</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EE7</td>
<td></td>
<td>HANGUL JONG SEONG SSANG SIOS</td>
<td>Korean</td>
</tr>
<tr>
<td>\x0EE8</td>
<td></td>
<td>HANGUL JONG SEONG IEUNG</td>
<td>Korean</td>
</tr>
<tr>
<td>KEYSYM value</td>
<td>Unicode value</td>
<td>Name</td>
<td>Set</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>---------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>#x0EE9</td>
<td></td>
<td>HANGUL JONG SEONG JIEUJ</td>
<td>Korean</td>
</tr>
<tr>
<td>#x0EEA</td>
<td></td>
<td>HANGUL JONG SEONG CIEUC</td>
<td>Korean</td>
</tr>
<tr>
<td>#x0EEB</td>
<td></td>
<td>HANGUL JONG SEONG KHEIUQ</td>
<td>Korean</td>
</tr>
<tr>
<td>#x0EEC</td>
<td></td>
<td>HANGUL JONG SEONG TIEUT</td>
<td>Korean</td>
</tr>
<tr>
<td>#x0EED</td>
<td></td>
<td>HANGUL JONG SEONG PHIEUF</td>
<td>Korean</td>
</tr>
<tr>
<td>#x0EEE</td>
<td></td>
<td>HANGUL JONG SEONG HIEUH</td>
<td>Korean</td>
</tr>
<tr>
<td>#x0EEF</td>
<td></td>
<td>HANGUL RIEUL YEORIN HIEUH</td>
<td>Korean</td>
</tr>
<tr>
<td>#x0EF0</td>
<td></td>
<td>HANGUL SUNKYEONGEUM MIEUM</td>
<td>Korean</td>
</tr>
<tr>
<td>#x0EF1</td>
<td></td>
<td>HANGUL SUNKYEONGEUM PIEUB</td>
<td>Korean</td>
</tr>
<tr>
<td>#x0EF2</td>
<td></td>
<td>HANGUL PAN SIOS</td>
<td>Korean</td>
</tr>
<tr>
<td>#x0EF3</td>
<td></td>
<td>HANGUL KKOJI DALRIN IEUNG</td>
<td>Korean</td>
</tr>
<tr>
<td>#x0EF4</td>
<td></td>
<td>HANGUL SUNKYEONGEUM PHIEUF</td>
<td>Korean</td>
</tr>
<tr>
<td>#x0EF5</td>
<td></td>
<td>HANGUL YEORIN HIEUH</td>
<td>Korean</td>
</tr>
<tr>
<td>#x0EF6</td>
<td></td>
<td>HANGUL ARAE A</td>
<td>Korean</td>
</tr>
<tr>
<td>#x0EF7</td>
<td></td>
<td>HANGUL ARAE AE</td>
<td>Korean</td>
</tr>
<tr>
<td>#x0EF8</td>
<td></td>
<td>HANGUL JONG SEONG PAN SIOS</td>
<td>Korean</td>
</tr>
<tr>
<td>#x0EF9</td>
<td></td>
<td>HANGUL JONG SEONG KKOJI DALRIN IEUNG</td>
<td>Korean</td>
</tr>
<tr>
<td>#x0EFA</td>
<td></td>
<td>HANGUL JONG SEONG YEORIN HIEUH</td>
<td>Korean</td>
</tr>
<tr>
<td>#x0EFF</td>
<td></td>
<td>KOREAN WON</td>
<td>Korean</td>
</tr>
</tbody>
</table>

| #x13BC       | U+0152        | LATIN CAPITAL LIGATURE OE                         | Latin-9   |
| #x13BD       | U+0153        | LATIN SMALL LIGATURE OE                           | Latin-9   |
| #x13BE       | U+0178        | LATIN CAPITAL LETTER Y WITH DIAERESIS             | Latin-9   |

| #x20A0       |               | CURRENCY ECU SIGN                                 | Currency  |
| #x20A1       |               | CURRENCY COLON SIGN                               | Currency  |
| #x20A2       |               | CURRENCY CRUZEIRO SIGN                            | Currency  |
| #x20A3       |               | CURRENCY FRENCH FRANC SIGN                        | Currency  |
| #x20A4       |               | CURRENCY LIRA SIGN                                | Currency  |
| #x20A5       |               | CURRENCY MILL SIGN                                | Currency  |
| #x20A6       |               | CURRENCY NAIRA SIGN                               | Currency  |
| #x20A7       |               | CURRENCY PESETA SIGN                              | Currency  |
| #x20A8       |               | CURRENCY RUPEE SIGN                               | Currency  |
| #x20A9       |               | CURRENCY WON SIGN                                 | Currency  |
| #x20AA       |               | CURRENCY NEW SHEQEL SIGN                          | Currency  |
| #x20AB       |               | CURRENCY DONG SIGN                                | Currency  |
| #x20AC       | U+20AC        | CURRENCY EURO SIGN                                | Currency  |
Appendix B
Protocol Encoding

Syntactic Conventions
All numbers are in decimal, unless prefixed with #x, in which case they are in hexadecimal (base 16).
The general syntax used to describe requests, replies, errors, events, and compound types is:

```
NameOfThing
   encode-form
   ...
   encode-form
```

Each encode-form describes a single component.
For components described in the protocol as:

```
name: TYPE
```
the encode-form is:

```
N   TYPE   name
```

N is the number of bytes occupied in the data stream, and TYPE is the interpretation of those bytes. For example,

```
depth: CARD8
```
becomes:

```
1   CARD8   depth
```
For components with a static numeric value the encode-form is:

```
N   value   name
```
The value is always interpreted as an N-byte unsigned integer. For example, the first two bytes of a Window error are always zero (indicating an error in general) and three (indicating the Window error in particular):

```
1   0   Error
1   3   code
```
For components described in the protocol as:

```
name: { Name1,..., NameI }
```
the encode-form is:

```
N   name
   value1 Name1
   ...
   valueI NameI
```
The value is always interpreted as an N-byte unsigned integer. Note that the size of N is sometimes larger than that strictly required to encode the values. For example:

```
class: { InputOutput, InputOnly, CopyFromParent }
```
becomes:
For components described in the protocol as:

NAME: TYPE or Alternative1...or AlternativeI

the encode-form is:

<table>
<thead>
<tr>
<th>N</th>
<th>TYPE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>value1</td>
<td>Alternative1</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>valueI</td>
<td>AlternativeI</td>
</tr>
</tbody>
</table>

The alternative values are guaranteed not to conflict with the encoding of TYPE. For example:

destination: WINDOW or PointerWindow or InputFocus

becomes:

4 WINDOW  destination
0 PointerWindow
1 InputFocus

For components described in the protocol as:

value-mask: BITMASK

the encode-form is:

<table>
<thead>
<tr>
<th>N</th>
<th>BITMASK</th>
<th>value-mask</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mask1</td>
<td>mask-name1</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>maskI</td>
<td>mask-nameI</td>
</tr>
</tbody>
</table>

The individual bits in the mask are specified and named, and N is 2 or 4. The most-significant bit in a BITMASK is reserved for use in defining chained (multiword) bitmasks, as extensions augment existing core requests. The precise interpretation of this bit is not yet defined here, although a probable mechanism is that a 1-bit indicates that another N bytes of bitmask follows, with bits within the overall mask still interpreted from least-significant to most-significant with an N-byte unit, with N-byte units interpreted in stream order, and with the overall mask being byte-swapped in individual N-byte units.

For LIsTofVALUE encodings, the request is followed by a section of the form:

VALUEs
  encode-form
  ...
  encode-form

listing an encode-form for each VALUE. The NAME in each encode-form keys to the corresponding BITMASK bit. The encoding of a VALUE always occupies four bytes, but the number of bytes specified in the encoding-form indicates how many of the least-significant bytes are actually used; the remaining bytes are unused and their values do not matter.

In various cases, the number of bytes occupied by a component will be specified by a lowercase single-letter variable name instead of a specific numeric value, and often some other component will have its value specified as a simple numeric expression involving these variables. Components specified with such expressions are always interpreted as unsigned integers. The scope of such variables is always just the enclosing request, reply, error, event, or compound type structure. For example:

2 3+n request length
4n LISTofPOINT points
For unused bytes (the values of the bytes are undefined and do no matter), the encode-form is:

\[ N \text{ unused} \]

If the number of unused bytes is variable, the encode-form typically is:

\[ p \text{ unused}, p=\text{pad}(E) \]

where \( E \) is some expression, and \( \text{pad}(E) \) is the number of bytes needed to round \( E \) up to a multiple of four.

\[ \text{pad}(E) = (4 - (E \mod 4)) \mod 4 \]

**Common Types**

LIST\text{of}FOO

In this document the LIST\text{of} notation strictly means some number of repetitions of the FOO encoding; the actual length of the list is encoded elsewhere.

SEToFOO

A set is always represented by a bitmask, with a 1-bit indicating presence in the set.

BITMASK: CARD32
WINDOW: CARD32
PIXMAP: CARD32
CURSOR: CARD32
FONT: CARD32
GCONTEXT: CARD32
COLORMAP: CARD32
DRAWABLE: CARD32
FONTABLE: CARD32
ATOM: CARD32
VISUALID: CARD32
BYTE: 8-bit value
INT8: 8-bit signed integer
INT16: 16-bit signed integer
INT32: 32-bit signed integer
CARD8: 8-bit unsigned integer
CARD16: 16-bit unsigned integer
CARD32: 32-bit unsigned integer
TIMESTAMP: CARD32

**BITGRAVITY**

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Forget</td>
</tr>
<tr>
<td>1</td>
<td>NorthWest</td>
</tr>
<tr>
<td>2</td>
<td>North</td>
</tr>
<tr>
<td>3</td>
<td>NorthEast</td>
</tr>
<tr>
<td>4</td>
<td>West</td>
</tr>
<tr>
<td>5</td>
<td>Center</td>
</tr>
<tr>
<td>6</td>
<td>East</td>
</tr>
<tr>
<td>7</td>
<td>SouthWest</td>
</tr>
<tr>
<td>8</td>
<td>South</td>
</tr>
<tr>
<td>9</td>
<td>SouthEast</td>
</tr>
<tr>
<td>10</td>
<td>Static</td>
</tr>
</tbody>
</table>

**WINGRAVITY**

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unmap</td>
</tr>
<tr>
<td>1</td>
<td>NorthWest</td>
</tr>
</tbody>
</table>
2 North
3 NorthEast
4 West
5 Center
6 East
7 SouthWest
8 South
9 SouthEast
10 Static

BOOL
0 False
1 True

SETofEVENT
#x00000001 KeyPress
#x00000002 KeyRelease
#x00000004 ButtonPress
#x00000008 ButtonRelease
#x00000010 EnterWindow
#x00000020 LeaveWindow
#x00000040 PointerMotion
#x00000080 PointerMotionHint
#x00000100 Button1Motion
#x00000200 Button2Motion
#x00000400 Button3Motion
#x00000800 Button4Motion
#x00001000 Button5Motion
#x00002000 ButtonMotion
#x00004000 KeymapState
#x00008000 Exposure
#x00100000 VisibilityChange
#x00200000 StructureNotify
#x00400000 ResizeRedirect
#x00800000 SubstructureNotify
#x01000000 SubstructureRedirect
#x02000000 FocusChange
#x04000000 PropertyChange
#x08000000 ColormapChange
#x10000000 OwnerGrabButton
#xFE000000 unused but must be zero

SETofPOINTEREVENT
encodings are the same as for SETofEVENT, except with
#xFFFF8003 unused but must be zero

SETofDEVICEEVENT
encodings are the same as for SETofEVENT, except with
#xFFFFC0B0 unused but must be zero

KEYSYM: CARD32
KEYCODE: CARD8
BUTTON: CARD8

SETofKEYBUTMASK
#x0001 Shift
#x0002 Lock
#x0004 Control
#x0008 Mod1
#x0010 Mod2
#x0020 Mod3
#x0040 Mod4
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#x0080</td>
<td>Mod5</td>
</tr>
<tr>
<td>#x0100</td>
<td>Button1</td>
</tr>
<tr>
<td>#x0200</td>
<td>Button2</td>
</tr>
<tr>
<td>#x0400</td>
<td>Button3</td>
</tr>
<tr>
<td>#x0800</td>
<td>Button4</td>
</tr>
<tr>
<td>#x1000</td>
<td>Button5</td>
</tr>
<tr>
<td>#xE000</td>
<td>unused but must be zero</td>
</tr>
</tbody>
</table>

SETofKEYMASK
encodings are the same as for SETofKEYBUTMASK, except with
#xFF00 unused but must be zero

STRING8: LISTofCARD8
STRING16: LISTofCHAR2B

<table>
<thead>
<tr>
<th>CARD8</th>
<th>byte1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>byte2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INT16</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>y</td>
</tr>
</tbody>
</table>

RECTANGLE
<table>
<thead>
<tr>
<th>INT16</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>y</td>
</tr>
<tr>
<td>CARD16</td>
<td>width</td>
</tr>
<tr>
<td>CARD16</td>
<td>height</td>
</tr>
</tbody>
</table>

ARC
<table>
<thead>
<tr>
<th>INT16</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>y</td>
</tr>
<tr>
<td>CARD16</td>
<td>width</td>
</tr>
<tr>
<td>CARD16</td>
<td>height</td>
</tr>
<tr>
<td>INT16</td>
<td>angle1</td>
</tr>
<tr>
<td>INT16</td>
<td>angle2</td>
</tr>
</tbody>
</table>

HOST
<table>
<thead>
<tr>
<th>family</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet</td>
</tr>
<tr>
<td>DECnet</td>
</tr>
<tr>
<td>Chaos</td>
</tr>
<tr>
<td>ServerInterpreted</td>
</tr>
<tr>
<td>InternetV6</td>
</tr>
<tr>
<td>unused</td>
</tr>
<tr>
<td>length of address</td>
</tr>
<tr>
<td>address</td>
</tr>
<tr>
<td>unused, p=pad(n)</td>
</tr>
</tbody>
</table>

STR
<table>
<thead>
<tr>
<th>length of name in bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>
Value
1 0 Error
1 2 code
2 CARD16 sequence number
4 <32-bits> bad value
2 CARD16 minor opcode
1 CARD8 major opcode
21 unused

Window
1 0 Error
1 3 code
2 CARD16 sequence number
4 CARD32 bad resource id
2 CARD16 minor opcode
1 CARD8 major opcode
21 unused

Pixmap
1 0 Error
1 4 code
2 CARD16 sequence number
4 CARD32 bad resource id
2 CARD16 minor opcode
1 CARD8 major opcode
21 unused

Atom
1 0 Error
1 5 code
2 CARD16 sequence number
4 CARD32 bad atom id
2 CARD16 minor opcode
1 CARD8 major opcode
21 unused

Cursor
1 0 Error
1 6 code
2 CARD16 sequence number
4 CARD32 bad resource id
2 CARD16 minor opcode
1 CARD8 major opcode
21 unused

Font
1 0 Error
1 7 code
2 CARD16 sequence number
4 CARD32 bad resource id
2 CARD16 minor opcode
1 CARD8 major opcode
21 unused

Match
1 0 Error
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>code</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>sequence number</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>unused</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>minor opcode</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CARD8</td>
<td>major opcode</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>unused</td>
<td></td>
</tr>
</tbody>
</table>

**Drawable**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>Error</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>code</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>sequence number</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>CARD32</td>
<td>bad resource id</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>minor opcode</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CARD8</td>
<td>major opcode</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>unused</td>
<td></td>
</tr>
</tbody>
</table>

**Access**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>Error</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>code</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>sequence number</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>unused</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>minor opcode</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CARD8</td>
<td>major opcode</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>unused</td>
<td></td>
</tr>
</tbody>
</table>

**Alloc**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>Error</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>code</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>sequence number</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>unused</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>minor opcode</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CARD8</td>
<td>major opcode</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>unused</td>
<td></td>
</tr>
</tbody>
</table>

**Colormap**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>Error</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>code</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>sequence number</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>CARD32</td>
<td>bad resource id</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>minor opcode</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CARD8</td>
<td>major opcode</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>unused</td>
<td></td>
</tr>
</tbody>
</table>

**GContext**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>Error</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>13</td>
<td>code</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>sequence number</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>CARD32</td>
<td>bad resource id</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>minor opcode</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CARD8</td>
<td>major opcode</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>unused</td>
<td></td>
</tr>
</tbody>
</table>

**IDChoice**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>Error</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>code</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>sequence number</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>CARD32</td>
<td>bad resource id</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>minor opcode</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CARD8</td>
<td>major opcode</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>unused</td>
<td></td>
</tr>
</tbody>
</table>
### Name

<table>
<thead>
<tr>
<th>Name</th>
<th>Length</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Error</td>
<td>Error</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>code</td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>sequence number</td>
</tr>
<tr>
<td>4</td>
<td>unused</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>minor opcode</td>
</tr>
<tr>
<td>1</td>
<td>CARD8</td>
<td>major opcode</td>
</tr>
<tr>
<td>21</td>
<td>unused</td>
<td></td>
</tr>
</tbody>
</table>

### Length

<table>
<thead>
<tr>
<th>Name</th>
<th>Length</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Error</td>
<td>Error</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>code</td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>sequence number</td>
</tr>
<tr>
<td>4</td>
<td>unused</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>minor opcode</td>
</tr>
<tr>
<td>1</td>
<td>CARD8</td>
<td>major opcode</td>
</tr>
<tr>
<td>21</td>
<td>unused</td>
<td></td>
</tr>
</tbody>
</table>

### Implementation

<table>
<thead>
<tr>
<th>Name</th>
<th>Length</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Error</td>
<td>Error</td>
</tr>
<tr>
<td>1</td>
<td>17</td>
<td>code</td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>sequence number</td>
</tr>
<tr>
<td>4</td>
<td>unused</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>minor opcode</td>
</tr>
<tr>
<td>1</td>
<td>CARD8</td>
<td>major opcode</td>
</tr>
<tr>
<td>21</td>
<td>unused</td>
<td></td>
</tr>
</tbody>
</table>

### Keyboards

KEYCODE values are always greater than 7 (and less than 256).

KEYSYM values with the bit #x10000000 set are reserved as vendor-specific.

The names and encodings of the standard KEYSYM values are contained in Appendix A, Keysym Encoding.

### Pointers

BUTTON values are numbered starting with one.

### Predefined Atoms

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Value Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIMARY</td>
<td>1</td>
<td>WM_NORMAL_HINTS</td>
</tr>
<tr>
<td>SECONDARY</td>
<td>2</td>
<td>WM_SIZE_HINTS</td>
</tr>
<tr>
<td>ARC</td>
<td>3</td>
<td>WM_ZOOM_HINTS</td>
</tr>
<tr>
<td>ATOM</td>
<td>4</td>
<td>MIN_SPACE</td>
</tr>
<tr>
<td>BITMAP</td>
<td>5</td>
<td>NORM_SPACE</td>
</tr>
<tr>
<td>CARDINAL</td>
<td>6</td>
<td>MAX_SPACE</td>
</tr>
<tr>
<td>COLORMAP</td>
<td>7</td>
<td>END_SPACE</td>
</tr>
<tr>
<td>CURSOR</td>
<td>8</td>
<td>SUPERSCRIPT_X</td>
</tr>
<tr>
<td>CUT_BUFFER0</td>
<td>9</td>
<td>SUPERSCRIPT_Y</td>
</tr>
<tr>
<td>CUT_BUFFER1</td>
<td>10</td>
<td>SUBSCRIPT_X</td>
</tr>
<tr>
<td>CUT_BUFFER2</td>
<td>11</td>
<td>SUBSCRIPT_Y</td>
</tr>
<tr>
<td>CUT_BUFFER3</td>
<td>12</td>
<td>UNDERLINE_POSITION</td>
</tr>
<tr>
<td>CUT_BUFFER4</td>
<td>13</td>
<td>UNDERLINE_THICKNESS</td>
</tr>
<tr>
<td>CUT_BUFFER5</td>
<td>14</td>
<td>STRIKEOUT_ASCENT</td>
</tr>
<tr>
<td>CUT_BUFFER6</td>
<td>15</td>
<td>STRIKEOUT_DESCENT</td>
</tr>
<tr>
<td>CUT_BUFFER7</td>
<td>16</td>
<td>ITALIC_ANGLE</td>
</tr>
<tr>
<td>DRAWABLE</td>
<td>17</td>
<td>X_HEIGHT</td>
</tr>
<tr>
<td>FONT</td>
<td>18</td>
<td>QUAD_WIDTH</td>
</tr>
<tr>
<td>INTEGER</td>
<td>19</td>
<td>WEIGHT</td>
</tr>
<tr>
<td>PIXMAP</td>
<td>20</td>
<td>POINT_SIZE</td>
</tr>
<tr>
<td>POINT</td>
<td>21</td>
<td>RESOLUTION</td>
</tr>
<tr>
<td>RECTANGLE</td>
<td>22</td>
<td>COPYRIGHT</td>
</tr>
<tr>
<td>RESOURCE_MANAGER</td>
<td>23</td>
<td>NOTICE</td>
</tr>
<tr>
<td>RGB_COLOR_MAP</td>
<td>24</td>
<td>FONT_NAME</td>
</tr>
</tbody>
</table>

117
RGB_BEST_MAP 25  FAMILY_NAME 64
RGB_BLUE_MAP 26  FULL_NAME 65
RGB_DEFAULT_MAP 27  CAP_HEIGHT 66
RGB_GRAY_MAP 28  WM_CLASS 67
RGB_GREEN_MAP 29  WM_TRANSIENT_FOR 68
RGB_RED_MAP 30
STRING 31
VISUALID 32
WINDOW 33
WM_COMMAND 34
WM_HINTS 35
WM_CLIENT_MACHINE 36
WM_ICON_NAME 37
WM_ICON_SIZE 38
WM_NAME 39

Connection Setup
For TCP connections, displays on a given host are numbered starting from 0, and the server for display N listens and accepts connections on port 6000 + N. For DECnet connections, displays on a given host are numbered starting from 0, and the server for display N listens and accepts connections on the object name obtained by concatenating “X$X” with the decimal representation of N, for example, X$X0 and X$X1.

Information sent by the client at connection setup:

1
#x42 MSB first
#x6C LSB first
1
CARD16 protocol-major-version
CARD16 protocol-minor-version
n
length of authorization-protocol-name
len
length of authorization-protocol-data
unused
STRING8 authorization-protocol-name
unused, p=pad(n)
d
STRING8 authorization-protocol-data
unused, q=pad(d)

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

Information received by the client if the connection is refused:

1
0 Failed
n
length of reason in bytes
CARD16 protocol-major-version
CARD16 protocol-minor-version
(n+p)/4
length in 4-byte units of “additional data”
STRING8 reason
unused, p=pad(n)

Information received by the client if further authentication is required:

1
2 Authenticate
5
unused
(n+p)/4
length in 4-byte units of “additional data”
STRING8 reason
unused, p=pad(n)

Information received by the client if the connection is accepted:

1
1 Success
1
unused
2 CARD16 protocol-major-version
2 CARD16 protocol-minor-version
2 S+2n+(v+p+m)/4 length in 4-byte units of “additional data”
4 CARD32 release-number
4 CARD32 resource-id-base
4 CARD32 resource-id-mask
4 CARD32 motion-buffer-size
2 v length of vendor
2 CARD16 maximum-request-length
1 CARD8 number of SCREENs in roots
1 n number for FORMATs in pixmap-formats
1 image-byte-order

0 LSBFirst
1 MSBFirst

1 0 LeastSignificant
1 1 MostSignificant

1 CARD8 bitmap-format-scanline-unit
1 CARD8 bitmap-format-scanline-pad
1 KEYCODE min-keycode
1 KEYCODE max-keycode
4 unused
v STRING8 vendor
p unused, p=pad(v)
8n LIST of FORMAT pixmap-formats
m LIST of SCREEN roots (m is always a multiple of 4)

FORMAT
1 CARD8 depth
1 CARD8 bits-per-pixel
1 CARD8 scanline-pad
5 unused

SCREEN
4 WINDOW root
4 COLORMAP default-colormap
4 CARD32 white-pixel
4 CARD32 black-pixel
4 SETOfEVENT current-input-masks
2 CARD16 width-in-pixels
2 CARD16 height-in-pixels
2 CARD16 width-in-millimeters
2 CARD16 height-in-millimeters
2 CARD16 min-installed-maps
2 CARD16 max-installed-maps
4 VISUALID root-visual
1 backing-stores

0 Never
1 WhenMapped
2 Always
1 BOOL save-unders
1 CARD8 root-depth
1 CARD8 number of DEPTHS in allowed-depths
n LIST Of DEPTH allowed-depths (n is always a multiple of 4)

DEPTH
1 CARD8 depth
1 unused
2 n number of VISUALTYPES in visuals
4 unused
24n LIST of VISUALTYPE visuals
VISUALTYPE
  4 VISUALID visual-id
  1 class
    0 StaticGray
    1 GrayScale
    2 StaticColor
    3 PseudoColor
    4 TrueColor
    5 DirectColor
  1 CARD8 bits-per-rgb-value
  2 CARD16 colormap-entries
  4 CARD32 red-mask
  4 CARD32 green-mask
  4 CARD32 blue-mask
  4 unused

Requests

CreateWindow
  1 1 opcode
  1 CARD8 depth
  2 8+n request length
  4 WINDOW wid
  4 WINDOW parent
  2 INT16 x
  2 INT16 y
  2 CARD16 width
  2 CARD16 height
  2 CARD16 border-width
  2 class
    0 CopyFromParent
    1 InputOutput
    2 InputOnly
  4 VISUALID visual
    0 CopyFromParent
  4 BITMASK value-mask (has n bits set to 1)
    #x00000001 background-pixmap
    #x00000002 background-pixel
    #x00000004 border-pixmap
    #x00000008 border-pixel
    #x00000010 bit-gravity
    #x00000020 win-gravity
    #x00000040 backing-store
    #x00000080 backing-planes
    #x00000200 override-redirect
    #x00000400 save-under
    #x00000800 event-mask
    #x00010000 do-not-propagate-mask
    #x00020000 colormap
    #x00040000 cursor
  4n LIST of VALUE value-list

VALUEs
  4 PIXMAP background-pixmap
    0 None
    1 ParentRelative
  4 CARD32 background-pixel
  4 PIXMAP border-pixmap
    0 CopyFromParent
  4 CARD32 border-pixel
    1 BITGRAVITY bit-gravity
    1 WINGRAVITY win-gravity
    1 backing-store
<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
<th>Type</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NotUseful</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>WhenMapped</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Always</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>CARD32</td>
<td>backing-planes</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>CARD32</td>
<td>backing-pixel</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>BOOL</td>
<td>override-redirect</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>BOOL</td>
<td>save-under</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SETofEVENT</td>
<td>event-mask</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SETofDEVICEEVENT</td>
<td>do-not-propagate-mask</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>COLORMAP</td>
<td>colormap</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>CURSOR</td>
<td>CopyFromParent</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>None</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**ChangeWindowAttributes**

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
<th>Type</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>opcode</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>unused</td>
<td>unused</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3+n</td>
<td>request length</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>WINDOW</td>
<td>window</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>BITMASK</td>
<td>value-mask (has n bits set to 1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>encodings are the same as for CreateWindow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4n</td>
<td>LISTofVALUE</td>
<td>value-list</td>
<td></td>
</tr>
<tr>
<td></td>
<td>encodings are the same as for CreateWindow</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**GetWindowAttributes**

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
<th>Type</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>opcode</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>unused</td>
<td>unused</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>request length</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>WINDOW</td>
<td>window</td>
<td></td>
</tr>
</tbody>
</table>

→

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
<th>Type</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Reply</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>backing-store</td>
<td>backing-store</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>NotUseful</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>WhenMapped</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Always</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
<td>sequence number</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>reply length</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>VISUALID</td>
<td>visual</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>InputOutput</td>
<td>InputOutput</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>InputOnly</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>BITGRAVITY</td>
<td>bit-gravity</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>WINGRAVITY</td>
<td>win-gravity</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>CARD32</td>
<td>backing-planes</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>CARD32</td>
<td>backing-pixel</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>BOOL</td>
<td>save-under</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>BOOL</td>
<td>map-is-installed</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>map-state</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Unmapped</td>
<td>Unmapped</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Unviewable</td>
<td>Unviewable</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Viewable</td>
<td>Viewable</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>BOOL</td>
<td>override-redirect</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>COLORMAP</td>
<td>colormap</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>None</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SETofEVENT</td>
<td>all-event-masks</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SETofEVENT</td>
<td>your-event-mask</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>SETofDEVICEEVENT</td>
<td>do-not-propagate-mask</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>unused</td>
<td>unused</td>
<td></td>
</tr>
</tbody>
</table>

**DestroyWindow**

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
<th>Type</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>opcode</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>unused</td>
<td>unused</td>
<td></td>
</tr>
</tbody>
</table>
DestroySubwindows
1 5 opcode
1 unused
2 2 request length
4 WINDOW window

ChangeSaveSet
1 6 opcode
1
0 Insert
1 Delete
2 2 request length
4 WINDOW window

ReparentWindow
1 7 opcode
1 unused
2 4 request length
4 WINDOW window
4 WINDOW parent
2 INT16 x
2 INT16 y

MapWindow
1 8 opcode
1 unused
2 2 request length
4 WINDOW window

MapSubwindows
1 9 opcode
1 unused
2 2 request length
4 WINDOW window

UnmapWindow
1 10 opcode
1 unused
2 2 request length
4 WINDOW window

UnmapSubwindows
1 11 opcode
1 unused
2 2 request length
4 WINDOW window

ConfigureWindow
1 12 opcode
1 unused
2 3+n request length
4 WINDOW window
2 BITMASK value-mask (has n bits set to 1)
  #x0001 x
  #x0002 y
  #x0004 width
  #x0008 height
#x0010 border-width
#x0020 sibling
#x0040 stack-mode

2 unused
4n LISTofVALUE value-list

VALUEs
2 INT16 x
2 INT16 y
2 CARD16 width
2 CARD16 height
2 CARD16 border-width
4 WINDOW sibling
1 stack-mode

0 Above
1 Below
2 TopIf
3 BottomIf
4 Opposite

CirculateWindow
1 13 opcode
1 direction
0 RaiseLowest
1 LowerHighest
2 2 request length
4 WINDOW window

GetGeometry
1 14 opcode
1 unused
2 2 request length
4 DRAWABLE drawable

→
1 1 Reply
1 CARD8 depth
2 CARD16 sequence number
4 0 reply length
4 WINDOW root
2 INT16 x
2 INT16 y
2 CARD16 width
2 CARD16 height
2 CARD16 border-width
10 unused

QueryTree
1 15 opcode
1 unused
2 2 request length
4 WINDOW window

→
1 1 Reply
1 unused
2 CARD16 sequence number
4 n reply length
4 WINDOW root
4 WINDOW parent
0 None number of WINDOWs in children
14
4n LISTofWINDOW

InternAtom
1 16     opcode
1 BOOL    only-if-exists
2 2+(n+p)/4  request length
2 n       length of name
2 unused
n STRING8  name
p unused, p=pad(n)

→
1 1     Reply
1 unused
2 CARD16 sequence number
4 0      reply length
4 ATOM atom
0 None unused

GetAtomName
1 17     opcode
1 unused
2 2      request length
4 ATOM atom

→
1 1     Reply
1 unused
2 CARD16 sequence number
4 (n+p)/4 reply length
2 n      length of name
22 unused
n STRING8 name
p unused, p=pad(n)

ChangeProperty
1 18     opcode
1 mode
0 Replace
1 Prepend
2 Append
2 6+(n+p)/4 request length
4 WINDOW window
4 ATOM property
4 ATOM type
1 CARD8 format
3 unused
4 CARD32 length of data in format units
   (= n for format = 8)
   (= n/2 for format = 16)
   (= n/4 for format = 32)
n LISTofBYTE data
   (n is a multiple of 2 for format = 16)
   (n is a multiple of 4 for format = 32)
p unused, p=pad(n)

DeleteProperty
1 19     opcode
1 unused
2 3      request length
GetProperty
1  20  opcode
1  BOOL  delete
2  6  request length
4  WINDOW  window
4  ATOM  property
4  ATOM  type
0  AnyPropertyType
4  CARD32  long-offset
4  CARD32  long-length

→
1  1  Reply
1  CARD8  format
2  CARD16  sequence number
4  (n+p)/4  reply length
4  ATOM  type
0  None
4  CARD32  bytes-after
4  CARD32  length of value in format units
  (= 0 for format = 0)
  (= n for format = 8)
  (= n/2 for format = 16)
  (= n/4 for format = 32)
12  unused
n LISTofBYTE value
  (n is zero for format = 0)
  (n is a multiple of 2 for format = 16)
  (n is a multiple of 4 for format = 32)
p

ListProperties
1  21  opcode
1  unused
2  2  request length
4  WINDOW  window

→
1  1  Reply
1  unused
2  CARD16  sequence number
4  n  reply length
2  n  number of ATOMs in atoms
22  unused
4n LISTofATOM atoms

SetSelectionOwner
1  22  opcode
1  unused
2  4  request length
4  WINDOW  owner
0  None
4  ATOM  selection
4  TIMESTAMP  time
0  CurrentTime

GetSelectionOwner
1  23  opcode
1  unused
2 2 request length
4 ATOM selection

→
1 1 Reply
1 unused
2 CARD16 sequence number
4 0 reply length
4 WINDOW owner
0 None unused

ConvertSelection
1 24 opcode
1 unused
2 6 request length
4 WINDOW requestor
4 ATOM selection
4 ATOM target
4 ATOM property
0 None time
0 CurrentTime

SendEvent
1 25 opcode
1 BOOL propagate
2 11 request length
4 WINDOW destination
0 PointerWindow
1 InputFocus
4 SETofEVENT event-mask
32 event
standard event format (see the Events section)

GrabPointer
1 26 opcode
1 BOOL owner-events
2 6 request length
4 WINDOW grab-window
2 SETofPOINTEREVENT event-mask
1 pointer-mode
0 Synchronous
1 Asynchronous
10 Synchronous
11 Asynchronous
4 WINDOW confine-to
0 None
4 CURSOR cursor
0 None
4 TIMEcAMP time
0 CurrentTime

→
1 1 Reply
1 status
0 Success
1 AlreadyGrabbed
2 InvalidTime
3 NotViewable
4 Frozen
2 CARD16 sequence number
UngrabPointer
1 27   opcode
1   unused
2  2   request length
4  TIME STAMP  time
   0  CurrentTime

GrabButton
1  28   opcode
1  BOOL  owner-events
2  6   request length
4  WINDOW  grab-window
2  SETofPOINTEREVENT  event-mask
1   pointer-mode
0  Synchronous
1  Asynchronous
1 0  Synchronous
1  Asynchronous
4  WINDOW  confine-to
0   None
4  CURSOR  cursor
0   None
1  BUTTON  button
0   AnyButton
1   unused
2  SETofKEYMASK  modifiers
#x8000  AnyModifier

UngrabButton
1  29   opcode
1  BUTTON  button
0   AnyButton
2  3   request length
4  WINDOW  grab-window
2  SETofKEYMASK  modifiers
#x8000  AnyModifier
2   unused

ChangeActivePointerGrab
1  30   opcode
1   unused
2  4   request length
4  CURSOR  cursor
0   None
4  TIME STAMP  time
0  CurrentTime
2  SETofPOINTEREVENT  event-mask
2   unused

GrabKeyboard
1  31   opcode
1  BOOL  owner-events
2  4   request length
4  WINDOW  grab-window
4  TIME STAMP  time
0  CurrentTime
1   pointer-mode
0  Synchronous
Asynchronous keyboard-mode
Synchronous
Asynchronous

Reply
status
Success
AlreadyGrabbed
InvalidTime
NotViewable
Frozen
sequence number
reply length
unused

UngrabKeyboard
opcode
unused
request length
time

GrabKey
opcode
owner-events
request length
grab-window
modifiers
key
pointer-mode
keyboard-mode
unused

UngrabKey
opcode
key
request length
grab-window
modifiers
unused

AllowEvents
opcode
mode
AsyncPointer
SyncPointer
ReplayPointer
AsyncKeyboard
SyncKeyboard
ReplayKeyboard
AsyncBoth
7 SyncBoth                  request length
2 2                        
4  TIMESTAMP              time
0  CurrentTime

GrabServer
1  36                        opcode
1                        unused
2  1                        request length

UngrabServer
1  37                        opcode
1                        unused
2  1                        request length

QueryPointer
1  38                        opcode
1                        unused
2  2                        request length
4  WINDOW                   window

→
1  1                        Reply
1  BOOL                     same-screen
2  CARD16                   sequence number
4  0                        reply length
4  WINDOW                   root
4  WINDOW                   child
0  None
2  INT16                    root-x
2  INT16                    root-y
2  INT16                    win-x
2  INT16                    win-y
2  SETofKEYBUTMASK          mask
6                        unused

GetMotionEvents
1  39                        opcode
1                        unused
2  4                        request length
4  WINDOW                   window
4  TIMESTAMP               start
0  CurrentTime
4  TIMESTAMP               stop
0  CurrentTime

→
1  1                        Reply
1                        unused
2  CARD16                   sequence number
4  2n                       reply length
4  n                        number of TIMECOORDs in events
20                        unused
8n  LISTofTIMECOORD         events

TIMECOORD
4  TIMESTAMP               time
2  INT16                    x
2  INT16                    y
TranslateCoordinates

| 1 | 40 | opcode   |
| 1 |     | unused   |
| 2 | 4   | request length |
| 4 | WINDOW | src-window |
| 4 | WINDOW | dst-window |
| 2 | INT16 | src-x     |
| 2 | INT16 | src-y     |

→

| 1 | 1   | Reply |
| 1 | BOOL | same-screen |
| 2 | CARD16 | sequence number |
| 4 | 0   | reply length |
| 4 | WINDOW | child |
| 0 |     | None   |
| 2 | INT16 | dst-x |
| 2 | INT16 | dst-y |
|16| | unused |

WarpPointer

| 1 | 41 | opcode   |
| 1 |     | unused   |
| 2 | 6   | request length |
| 4 | WINDOW | src-window |
| 0 |     | None |
| 4 | WINDOW | dst-window |
| 0 |     | None |
| 2 | INT16 | src-x |
| 2 | INT16 | src-y |
| 2 | CARD16 | src-width |
| 2 | CARD16 | src-height |
| 2 | INT16 | dst-x |
| 2 | INT16 | dst-y |

SetInputFocus

| 1 | 42 | opcode |
| 1 |     | revert-to |
| 0 |     | None |
| 1 | PointerRoot |
| 2 | Parent    |
| 2 | 3 | request length |
| 4 | WINDOW | focus |
| 0 |     | None |
| 1 | PointerRoot |
| 4 | TIMESTAMP | time |

GetInputFocus

| 1 | 43 | opcode   |
| 1 |     | unused   |
| 2 | 1   | request length |

→

| 1 | 1   | Reply |
| 1 |     | revert-to |
| 0 |     | None   |
| 1 | PointerRoot |
| 2 | Parent |
| 2 | CARD16 | sequence number |
| 4 | 0   | reply length |
| 4 | WINDOW | focus |
| 0 |     | None   |
QueryKeymap
1 44  opcode
1  unused
2 1  request length

→
1 1  Reply
1  unused
2  CARD16  sequence number
4 2  reply length
32 LISTofCARD8  keys

OpenFont
1 45  opcode
1  unused
2 3+(n+p)/4  request length
4  FONT  fid
2  n  length of name
2  unused
n  STRING8  name
p  unused, p=pad(n)

CloseFont
1 46  opcode
1  unused
2 2  request length
4  FONT  font

QueryFont
1 47  opcode
1  unused
2 2  request length
4  FONTABLE  font

→
1 1  Reply
1  unused
2  CARD16  sequence number
4 7+2n+3m  reply length
12 CHARINFO  min-bounds
4  unused
12 CHARINFO  max-bounds
4  unused
2  CARD16  min-char-or-byte2
2  CARD16  max-char-or-byte2
2  CARD16  default-char
2  n  number of FONTPROPs in properties
1  draw-direction
0  LeftToRight
1  RightToLeft
1 CARD8  min-byte1
1 CARD8  max-byte1
1 BOOL  all-chars-exist
2 INT16  font-ascent
2 INT16  font-descent
4 m  number of CHARINFOs in char-infos
8n LISTofFONTPROP  properties
12m LISTofCHARINFO  char-infos
FONTPROP
4  ATOM name
4  <32-bits> value

CHARINFO
2  INT16 left-side-bearing
2  INT16 right-side-bearing
2  INT16 character-width
2  INT16 ascent
2  INT16 descent
2  CARD16 attributes

QueryTextExtents
1  48 opcode
1  BOOL odd length, True if p = 2
2  2+(2n+p)/4 request length
4  FONTABLE font
2n STRING16 string
p unused, p=pad(2n)
→
1  1 Reply
0 LeftToLeft
1 RightToRight
2 CARD16 sequence number
4 0 reply length
2 INT16 font-ascent
2 INT16 font-descent
2 INT16 overall-ascent
2 INT16 overall-descent
4 INT16 overall-width
4 INT16 overall-left
4 INT16 overall-right
4 unused

ListFonts
1  49 opcode
1 unused
2  2+(n+p)/4 request length
2 CARD16 max-names
2 n length of pattern
n STRING8 pattern
p unused, p=pad(n)
→
1  1 Reply
1 unused
2 CARD16 sequence number
4 (n+p)/4 reply length
2 CARD16 number of STRs in names
22 unused
n LISTofSTR names
p unused, p=pad(n)

ListFontsWithInfo
1  50 opcode
1 unused
2  2+(n+p)/4 request length
2 CARD16 max-names
2 n length of pattern
n STRING8 pattern
p

→ (except for last in series)
1 1
2 CARD16
4 7+2m+(n+p)/4
12 CHARINFO
4 unused
12 CHARINFO
4 unused
2 CARD16
2 CARD16
2 CARD16
2 m
1
0
1
1
1
2
2
2
4
4
8m
n
p

CARD8
CARD8
BOOL
INT16
INT16
CARD32
LISTofFONTPROP
STRING8

min-byte
max-byte
all-chars-exist
font-ascent
font-descent
replies-hint
properties
name

FONTPROP
encodings are the same as for QueryFont

CHARINFO
encodings are the same as for QueryFont

→ (last in series)
1 1
2 0
4 7
52
1 51
2 2+(n+p)/4
2 CARD16
2 unused
n LISTofSTR
p

GetFontPath
1 52
2 1

SetFontPath
1 51
2 2+(n+p)/4
2 CARD16
2 unused
n LISTofSTR
p

→
1 1
2 CARD16
4 (n+p)/4
2 CARD16
22 unused
n LISTofSTR

Reply
length of name in bytes
sequence number
reply length
min-bounds
unused
max-bounds
unused
min-char-or-byte2
max-char-or-byte2
default-char
number of FONTPROPs in properties
draw-direction
LeftToRight
RightToLeft
Reply
last-reply indicator
sequence number
reply length
unused
opcode
unused
request list
Reply
unused
sequence number
reply length
number of STRs in path
unused
path

133
CreatePixmap
1 15 53
2 CARD8
4 4
4 PIXMAP
4 DRAWABLE
2 CARD16
2 CARD16

FreePixmap
1 15 54
2 2
4 PIXMAP

CreateGC
1 15 55
2 4+n
4 GCONTEXT
4 DRAWABLE
4 BITMASK

VALUEs
1 function
0 Clear
1 And
2 AndReverse
3 Copy
4 AndInverted
5 NoOp
6 Xor
7 Or
8 Nor
9 Equiv
10 Invert
11 OrReverse
12 CopyInverted
13 OrInverted
14 Nand
15 Set
4 CARD32 plane-mask
4 CARD32 foreground
4 CARD32 background
2 CARD16 line-width
1 line-style
0 Solid
1 OnOffDash
2 DoubleDash
1 cap-style
0 NotLast
1 Butt
2 Round
3 Projecting
1 join-style
0 Miter
1 Round
2 Bevel
1 fill-style
0 Solid
1 Tiled
2 Stippled
3 OpaqueStippled
1 fill-rule
0 EvenOdd
1 Winding
4 PIXMAP tile
4 PIXMAP stipple
2 INT16 tile-stipple-x-origin
2 INT16 tile-stipple-y-origin
4 FONT font
1 subwindow-mode
0 ClipByChildren
1 IncludeInferiors
1 BOOL graphics-exposures
2 INT16 clip-x-origin
2 INT16 clip-y-origin
4 PIXMAP clip-mask
0 None
2 CARD16 dash-offset
1 CARD8 dashes
1 arc-mode
0 Chord
1 PieSlice

ChangeGC
1 56 opcode
1 unused
2 3+n request length
4 GCONTEXT gc
4 BTMAsk value-mask (has n bits set to 1)
encodings are the same as for CreateGC
4n LISTofVALUE value-list
encodings are the same as for CreateGC

CopyGC
1 57 opcode
1 unused
2 4 request length
4 GCONTEXT src-gc
4 GCONTEXT dst-gc
SetDashes
1  58  opcode
1  unused
2  \(3+(n+p)/4\)  request length
4  GCONTEXT  gc
2  CARD16  dash-offset
2  n  length of dashes
n  LISTtCARD8  dashes
p  unused, \(p=\text{pad}(n)\)

SetClipRectangles
1  59  opcode
1  ordering
0  UnSorted
1  YSorted
2  YXSorted
3  YXBanded
2  \(3+2n\)  request length
4  GCONTEXT  gc
2  INT16  clip-x-origin
2  INT16  clip-y-origin
8n  LISTtRECTANGLE  rectangles

FreeGC
1  60  opcode
1  unused
2  2  request length
4  GCONTEXT  gc

ClearArea
1  61  opcode
1  BOOL  exposures
2  4  request length
4  WINDOW  window
2  INT16  x
2  INT16  y
2  INT16  width
2  CARD16  height

CopyArea
1  62  opcode
1  unused
2  7  request length
4  DRAWABLE  src-drawable
4  DRAWABLE  dst-drawable
4  GCONTEXT  gc
2  INT16  src-x
2  INT16  src-y
2  INT16  dst-x
2  INT16  dst-y
2  CARD16  width
2  CARD16  height

CopyPlane
1  63  opcode
1  unused
2  8  request length
4  DRAWABLE  src-drawable
PolyPoint
1 64  opcode
1 0  Origin
1 1  Previous
2 3+n  request length
4 DRAWABLE  drawable
4 GCONTEXT  gc
4n LISTofPOINT  points

PolyLine
1 65  opcode
1 0  Origin
1 1  Previous
2 3+n  request length
4 DRAWABLE  drawable
4 GCONTEXT  gc
4n LISTofPOINT  points

PolySegment
1 66  opcode
1 2  unused
2 3+2n  request length
4 DRAWABLE  drawable
4 GCONTEXT  gc
8n LISTofSEGMENT  segments

SEGMENT
2 INT16  x1
2 INT16  y1
2 INT16  x2
2 INT16  y2

PolyRectangle
1 67  opcode
1 2  unused
2 3+2n  request length
4 DRAWABLE  drawable
4 GCONTEXT  gc
8n LISTofRECTANGLE  rectangles

PolyArc
1 68  opcode
1 2  unused
2 3+3n  request length
4 DRAWABLE  drawable
4 GCONTEXT  gc
12n LISTofARC  arcs

FillPoly
PolyFillRectangle

```
1 70  opcode
1 unused
2 3+2n request length
4 DRAWABLE drawable
4 GCONTEXT gc
8n LISTofRECTANGLE rectangles
```

PolyFillArc

```
1 71  opcode
1 unused
2 3+3n request length
4 DRAWABLE drawable
4 GCONTEXT gc
12n LISTofARC arcs
```

PutImage

```
1 72  opcode
1 format
0 Bitmap
1 XYPixmap
2 ZPixmap
2 6+(n+p)/4 request length
4 DRAWABLE drawable
4 GCONTEXT gc
2 CARD16 width
2 CARD16 height
2 INT16 dst-x
2 INT16 dst-y
1 CARD8 left-pad
1 CARD8 depth
2 unused
n LISTofBYTE data
p unused, p=pad(n)
```

GetImage

```
1 73  opcode
1 format
1 XYPixmap
2 ZPixmap
2 5 request length
4 DRAWABLE drawable
2 INT16 x
2 INT16 y
2 CARD16 width
2 CARD16 height
4 CARD32 plane-mask
```
PolyText8

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>opcode</td>
</tr>
<tr>
<td>1</td>
<td>unused</td>
</tr>
<tr>
<td>2</td>
<td>request length</td>
</tr>
<tr>
<td>4</td>
<td>drawable</td>
</tr>
<tr>
<td>4</td>
<td>gc</td>
</tr>
<tr>
<td>2</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>y</td>
</tr>
<tr>
<td>n</td>
<td>items</td>
</tr>
<tr>
<td>p</td>
<td>unused, p=pad(n) (p is always 0 or 1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TEXTITEM8</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>length of string (cannot be 255)</td>
</tr>
<tr>
<td>1</td>
<td>delta</td>
</tr>
<tr>
<td>m</td>
<td>string</td>
</tr>
<tr>
<td>or</td>
<td>font-shift indicator</td>
</tr>
<tr>
<td>1</td>
<td>font byte 3 (most-significant)</td>
</tr>
<tr>
<td>1</td>
<td>font byte 2</td>
</tr>
<tr>
<td>1</td>
<td>font byte 1</td>
</tr>
<tr>
<td>1</td>
<td>font byte 0 (least-significant)</td>
</tr>
</tbody>
</table>

PolyText16

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>opcode</td>
</tr>
<tr>
<td>1</td>
<td>unused</td>
</tr>
<tr>
<td>2</td>
<td>request length</td>
</tr>
<tr>
<td>4</td>
<td>drawable</td>
</tr>
<tr>
<td>4</td>
<td>gc</td>
</tr>
<tr>
<td>2</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>y</td>
</tr>
<tr>
<td>n</td>
<td>items</td>
</tr>
<tr>
<td>p</td>
<td>unused, p=pad(n) (p must be 0 or 1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TEXTITEM16</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>number of CHAR2Bs in string (cannot be 255)</td>
</tr>
<tr>
<td>1</td>
<td>delta</td>
</tr>
<tr>
<td>2m</td>
<td>string</td>
</tr>
<tr>
<td>or</td>
<td>font-shift indicator</td>
</tr>
<tr>
<td>1</td>
<td>font byte 3 (most-significant)</td>
</tr>
<tr>
<td>1</td>
<td>font byte 2</td>
</tr>
<tr>
<td>1</td>
<td>font byte 1</td>
</tr>
<tr>
<td>1</td>
<td>font byte 0 (least-significant)</td>
</tr>
</tbody>
</table>

ImageText8

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>opcode</td>
</tr>
<tr>
<td>1</td>
<td>length of string</td>
</tr>
<tr>
<td>2</td>
<td>request length</td>
</tr>
<tr>
<td>4</td>
<td>drawable</td>
</tr>
<tr>
<td>4</td>
<td>gc</td>
</tr>
<tr>
<td>2</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>y</td>
</tr>
</tbody>
</table>
n STRING8 string
p unused, p=pad(n)

ImageText16
1 77 opcode
1 n number of CHAR2Bs in string
2 4+(2n+p)/4 request length
4 DRAWABLE drawable
4 GCONTEXT gc
2 INT16 x
2 INT16 y
2n STRING16 string
p unused, p=pad(2n)

CreateColormap
1 78 opcode
1 0 None
1 1 All
2 4 request length
4 COLORMAP mid
4 WINDOW window
4 VISUALID visual

FreeColormap
1 79 opcode
1 unused
2 2 request length
4 COLORMAP cmap

CopyColormapAndFree
1 80 opcode
1 unused
2 3 request length
4 COLORMAP mid
4 COLORMAP src-cmap

InstallColormap
1 81 opcode
1 unused
2 2 request length
4 COLORMAP cmap

UninstallColormap
1 82 opcode
1 unused
2 2 request length
4 COLORMAP cmap

ListInstalledColormaps
1 83 opcode
1 unused
2 2 request length
4 WINDOW window

→
1 1 Reply
1 unused
2 CARD16 sequence number
4 n reply length
number of COLORMAPs in cmaps

LISTofCOLORMAP

cmaps

AllocColor

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>84</td>
<td>AllocColor</td>
</tr>
<tr>
<td>1</td>
<td>unused</td>
</tr>
<tr>
<td>4</td>
<td>request length</td>
</tr>
<tr>
<td>4</td>
<td>COLORMAP</td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
</tr>
<tr>
<td>4</td>
<td>CARD32</td>
</tr>
</tbody>
</table>

Allocated

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Reply</td>
</tr>
<tr>
<td>1</td>
<td>unused</td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
</tr>
<tr>
<td>4</td>
<td>CARD32</td>
</tr>
</tbody>
</table>

AllocNamedColor

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>85</td>
<td>AllocNamedColor</td>
</tr>
<tr>
<td>2</td>
<td>request length</td>
</tr>
<tr>
<td>4</td>
<td>COLORMAP</td>
</tr>
<tr>
<td>2</td>
<td>n</td>
</tr>
<tr>
<td>n</td>
<td>STRING8</td>
</tr>
<tr>
<td>p</td>
<td>unused, p=pad(n)</td>
</tr>
</tbody>
</table>

Allocated

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Reply</td>
</tr>
<tr>
<td>1</td>
<td>unused</td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
</tr>
</tbody>
</table>

AllocColorCells

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>86</td>
<td>AllocColorCells</td>
</tr>
<tr>
<td>1</td>
<td>BOOL</td>
</tr>
<tr>
<td>2</td>
<td>request length</td>
</tr>
<tr>
<td>4</td>
<td>COLORMAP</td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
</tr>
</tbody>
</table>

Allocated

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Reply</td>
</tr>
<tr>
<td>1</td>
<td>unused</td>
</tr>
<tr>
<td>2</td>
<td>CARD16</td>
</tr>
</tbody>
</table>

141
4 n+mreply length
2 nnumber of CARD32s in pixels
2 mnumber of CARD32s in masks
20 unused
4n LISTofCARD32 pixels
4m LISTofCARD32 masks

AllocColorPlanes
1 87 opcode
1 BOOL contiguous
2 4 request length
4 COLORMAP cmap
2 CARD16 colors
2 CARD16 reds
2 CARD16 greens
2 CARD16 blues

→
1 1 Reply
1 unused
2 CARD16 sequence number
4 n reply length
2 n number of CARD32s in pixels
2 unused
4 CARD32 red-mask
4 CARD32 green-mask
4 CARD32 blue-mask
8 unused
4n LISTofCARD32 pixels

FreeColors
1 88 opcode
1 unused
2 3+n request length
4 COLORMAP cmap
4 CARD32 plane-mask
4n LISTofCARD32 pixels

StoreColors
1 89 opcode
1 unused
2 2+3n request length
4 COLORMAP cmap
12n LISTofCOLORITEM items

COLORITEM
4 CARD32 pixel
2 CARD16 red
2 CARD16 green
2 CARD16 blue
1 do-red, do-green, do-blue
   #x01 do-red (1 is True, 0 is False)
   #x02 do-green (1 is True, 0 is False)
   #x04 do-blue (1 is True, 0 is False)
   #xF8 unused
1 unused

StoreNamedColor
1 90 opcode
1 do-red, do-green, do-blue
   #x01 do-red (1 is True, 0 is False)
   #x02 do-green (1 is True, 0 is False)
XP rotocol X11, Release 6.9/7.0

#x04 do-blue (1 is True, 0 is False)

#xF8 unused

2  4+(n+p)/4 request length

4  4 CARD32 pixel

2  n length of name

2 unused

n  STRING8 name

p unused, p=pad(n)

QueryColors

1  91 opcode

1 unused

2  2+n request length

4  4 COLORMAP cmap

4n LISTofCARD32 pixels

→

1  1 Reply

1 unused

2  CARD16 sequence number

4  2n reply length

2  n number of RGBs in colors

22 unused

8n LISTofRGB colors

RGB

2  CARD16 red

2  CARD16 green

2  CARD16 blue

2 unused

LookupColor

1  92 opcode

1 unused

2  3+(n+p)/4 request length

4  4 COLORMAP cmap

2  n length of name

2 unused

n  STRING8 name

p unused, p=pad(n)

→

1  1 Reply

1 unused

2  CARD16 sequence number

4  0 reply length

2  CARD16 exact-red

2  CARD16 exact-green

2  CARD16 exact-blue

2  CARD16 visual-red

2  CARD16 visual-green

2  CARD16 visual-blue

12 unused

CreateCursor

1  93 opcode

1 unused

2  8 request length

4  CURSOR cid

4  PIXMAP source

4  PIXMAP mask
CreateGlyphCursor
1  94          opcode
1          unused
2  8          request length
4  CURSOR      cid
4  FONT        source-font
4  FONT        mask-font
0          None
2  CARD16      source-char
2  CARD16      mask-char
2  CARD16      fore-red
2  CARD16      fore-green
2  CARD16      fore-blue
2  CARD16      back-red
2  CARD16      back-green
2  CARD16      back-blue

FreeCursor
1  95          opcode
1          unused
2  2          request length
4  CURSOR      cursor

RecolorCursor
1  96          opcode
1          unused
2  5          request length
4  CURSOR      cursor
2  CARD16      fore-red
2  CARD16      fore-green
2  CARD16      fore-blue
2  CARD16      back-red
2  CARD16      back-green
2  CARD16      back-blue

QueryBestSize
1  97          opcode
1          class
0          Cursor
1          Tile
2          Stipple
2  3          request length
4  DRAWABLE    drawable
2  CARD16      width
2  CARD16      height

→
1  1          Reply
1          unused
2  CARD16      sequence number
4  0          reply length
2  CARD16      width
2 CARD16 height
20 unused

QueryExtension
1 98 opcode
1 unused
2 2+(n+p)/4 request length
2 n length of name
2 unused
n STRING8 name
p unused, p=pad(n)

→
1 1 Reply
1 unused
2 CARD16 sequence number
4 0 reply length
1 BOOL present
1 CARD8 major-opcode
1 CARD8 first-event
1 CARD8 first-error
20 unused

ListExtensions
1 99 opcode
1 unused
2 1 request length

→
1 1 Reply
1 CARD8 number of STRs in names
2 CARD16 sequence number
4 (n+p)/4 reply length
24 unused
n LISTofSTR names
p unused, p=pad(n)

ChangeKeyboardMapping
1 100 opcode
1 n keycode-count
2 2+nm request length
1 KEYCODE first-keycode
1 m keysyms-per-keycode
2 unused
4nm LISTofKEYSYM keysyms

GetKeyboardMapping
1 101 opcode
1 unused
2 2 request length
1 KEYCODE first-keycode
1 m count
2 unused

→
1 1 Reply
1 n keysyms-per-keycode
2 CARD16 sequence number
4 nm reply length (m = count field from the request)
24 unused
4nm LISTofKEYSYM keysyms
ChangeKeyboardControl
1 102  opcode
1 unused
2 2+n  request length
4 BITMASK  value-mask (has n bits set to 1)
  #x0001  key-click-percent
  #x0002  bell-percent
  #x0004  bell-pitch
  #x0008  bell-duration
  #x0010  led
  #x0020  led-mode
  #x0040  key
  #x0080  auto-repeat-mode
4n LISTtoVALUE  value-list

VALUES
1 INT8  key-click-percent
1 INT8  bell-percent
2 INT16  bell-pitch
2 INT16  bell-duration
1 CARD8  led
1 led-mode
0  Off
1  On
1 KEYCODE  key
1 auto-repeat-mode
0  Off
1  On
2 Default

GetKeyboardControl
1 103  opcode
1 unused
2 1  request length

→
1 1  Reply
1 global-auto-repeat
0  Off
1  On
2 CARD16  sequence number
4 5  reply length
4 CARD32  led-mask
1 CARD8  key-click-percent
1 CARD8  bell-percent
2 CARD16  bell-pitch
2 CARD16  bell-duration
2 unused
32 LISTtoCARD8  auto-repeats

Bell
1 104  opcode
1 INT8  percent
2 1  request length

ChangePointerControl
1 105  opcode
1 unused
2 3  request length
2 INT16  acceleration-numerator
2 INT16  acceleration-denominator
2 INT16  threshold
1 BOOL  do-acceleration
GetPointerControl
1 106  opcode
1 unused
2 1 request length
→
1 1 Reply
1 unused
2 CARD16 sequence number
4 0 reply length
2 CARD16 acceleration-numerator
2 CARD16 acceleration-denominator
2 CARD16 threshold
18 unused

SetScreenSaver
1 107  opcode
1 unused
2 3 request length
2 INT16 timeout
2 INT16 interval
1 prefer-blanking
0 No
1 Yes
2 Default
1 allow-exposures
0 No
1 Yes
2 Default
2 unused

GetScreenSaver
1 108  opcode
1 unused
2 1 request length
→
1 1 Reply
1 unused
2 CARD16 sequence number
4 0 reply length
2 CARD16 timeout
2 CARD16 interval
1 prefer-blanking
0 No
1 Yes
1 allow-exposures
0 No
1 Yes
18 unused

ChangeHosts
1 109  opcode
1 mode
0 Insert
1 Delete
2 2+(n+p)/4 request length
1 family
0 Internet
1 DECnet
XP rotocol X11, Release 6.9/7.0

2C haos
1u nused
2n l ength of address
nL ISTofCARD8 address
p unused, p=pad(n)

ListHosts
1 110 opcode
1 unused
2 1 request length

→
1 1 Reply
1 mode
0 0 Disabled
 1 Enabled
2 CARD16 sequence number
4 n/4 reply length
2 CARD16 number of HOSTs in hosts
22 unused
n LISTofHOST hosts (n always a multiple of 4)

SetAccessControl
1 111 opcode
1 mode
0 0 Disable
 1 Enable
2 1 request length

SetCloseDownMode
1 112 opcode
1 mode
0 0 Destroy
 1 RetainPermanent
 2 RetainTemporary
2 1 request length

KillClient
1 113 opcode
1 unused
2 2 request length
4 CARD32 resource
0 0 AllTemporary

RotateProperties
1 114 opcode
1 unused
2 3+n request length
4 WINDOW window
2 n number of properties
2 INT16 delta
4n LISTofATOM properties

ForceScreenSaver
1 115 opcode
1 mode
0 0 Reset
 1 Activate
2 1 request length
SetPointerMapping
1 116  opcode
1 n  length of map
2 1+(n+p)/4  request length
n LISTofCARD8  map
p  unused, p=pad(n)

→
1 1  Reply
1 0  Success
1 1  Busy
2 CARD16  sequence number
4 0  reply length
24  unused

GetPointerMapping
1 117  opcode
1  unused
2 1  request length

→
1 1  Reply
1 n  length of map
2 CARD16  sequence number
4 (n+p)/4  reply length
24  unused
n LISTofCARD8  map
p  unused, p=pad(n)

SetModifierMapping
1 118  opcode
1 n  keycodes-per-modifier
2 1+2n  request length
8n LISTofKEYCODE  keycodes

→
1 1  Reply
1 0  Success
1 1  Busy
2 2  Failed
2 CARD16  sequence number
4 0  reply length
24  unused

GetModifierMapping
1 119  opcode
1  unused
2 1  request length

→
1 1  Reply
1 n  keycodes-per-modifier
2 CARD16  sequence number
4 2n  reply length
24  unused
8n LISTofKEYCODE  keycodes

NoOperation
1 127  opcode
1
2 1+n
3 unused
4n unused

Events

KeyPress
1 2 code
1 KEYCODE detail
2 CARD16 sequence number
4 TIMESTAMP time
4 WINDOW root
4 WINDOW event
4 WINDOW child
0 None
2 INT16 root-x
2 INT16 root-y
2 INT16 event-x
2 INT16 event-y
2 SETofKEYBUTMASK state
1 BOOL same-screen
1 unused

KeyRelease
1 3 code
1 KEYCODE detail
2 CARD16 sequence number
4 TIMESTAMP time
4 WINDOW root
4 WINDOW event
4 WINDOW child
0 None
2 INT16 root-x
2 INT16 root-y
2 INT16 event-x
2 INT16 event-y
2 SETofKEYBUTMASK state
1 BOOL same-screen
1 unused

ButtonPress
1 4 code
1 BUTTON detail
2 CARD16 sequence number
4 TIMESTAMP time
4 WINDOW root
4 WINDOW event
4 WINDOW child
0 None
2 INT16 root-x
2 INT16 root-y
2 INT16 event-x
2 INT16 event-y
2 SETofKEYBUTMASK state
1 BOOL same-screen
1 unused

ButtonRelease
1 5 code
1 BUTTON detail
2 CARD16 sequence number
4 TIMESTAMP time
4 WINDOW root
MotionNotify
1 6
2 0 Normal
2 1 Hint
2 2 CARD16 sequence number
2 4 TIMESTAMP time
2 4 WINDOW root
2 4 WINDOW event
2 4 WINDOW child
2 0 None
2 2 INT16 root-x
2 2 INT16 root-y
2 2 INT16 event-x
2 2 INT16 event-y
2 2 SETofKEYBUTMASK state
2 1 BOOL same-screen
2 1 unused

EnterNotify
1 7
1 0 Ancestor
1 1 Virtual
1 2 Inferior
1 3 Nonlinear
1 4 NonlinearVirtual
1 2 CARD16 sequence number
1 4 TIMESTAMP time
1 4 WINDOW root
1 4 WINDOW event
1 4 WINDOW child
1 0 None
1 2 INT16 root-x
1 2 INT16 root-y
1 2 INT16 event-x
1 2 INT16 event-y
1 2 SETofKEYBUTMASK state
1 0 mode
1 0 Normal
1 1 Grab
1 2 Ungrab
1 #x01 focus (1 is True, 0 is False)
1 #x02 same-screen (1 is True, 0 is False)
1 #xF0 unused

LeaveNotify
1 8
2 0 Ancestor
2 1 Virtual

151
2 Inferior
3 Nonlinear
4 NonlinearVirtual
2 CARD16 sequence number
4 TIMESTAMP time
4 WINDOW root
4 WINDOW event
4 WINDOW child
0 None
2 INT16 root-x
2 INT16 root-y
2 INT16 event-x
2 INT16 event-y
2 SETofKEYBUTMASK state
1 mode
0 Normal
1 Grab
2 Ungrab
1 same-screen, focus
#x01 focus (1 is True, 0 is False)
#x02 same-screen (1 is True, 0 is False)
#xFF unused

FocusIn
1 9 code
1 detail
0 Ancestor
1 Virtual
2 Inferior
3 Nonlinear
4 NonlinearVirtual
5 Pointer
6 PointerRoot
7 None
2 CARD16 sequence number
4 WINDOW event
1 mode
0 Normal
1 Grab
2 Ungrab
3 WhileGrabbed
23 unused

FocusOut
1 10 code
1 detail
0 Ancestor
1 Virtual
2 Inferior
3 Nonlinear
4 NonlinearVirtual
5 Pointer
6 PointerRoot
7 None
2 CARD16 sequence number
4 WINDOW event
1 mode
0 Normal
1 Grab
2 Ungrab
3 WhileGrabbed
23 unused
KeymapNotify
1 11 code
31 LISTtoCARD8 keys (byte for keycodes 0–7 is omitted)

Expose
1 12 code
1 unused
2 CARD16 sequence number
4 WINDOW window
2 CARD16 x
2 CARD16 y
2 CARD16 width
2 CARD16 height
2 CARD16 count
14 unused

GraphicsExposure
1 13 code
1 unused
2 CARD16 sequence number
4 DRAWABLE drawable
2 CARD16 x
2 CARD16 y
2 CARD16 width
2 CARD16 height
2 CARD16 minor-opcode
2 CARD16 count
1 CARD8 major-opcode
11 unused

NoExposure
1 14 code
1 unused
2 CARD16 sequence number
4 DRAWABLE drawable
2 CARD16 minor-opcode
1 CARD8 major-opcode
21 unused

VisibilityNotify
1 15 code
1 unused
2 CARD16 sequence number
4 WINDOW window
1 state
0 Unobscured
1 PartiallyObscured
2 FullyObscured
23 unused

CreateNotify
1 16 code
1 unused
2 CARD16 sequence number
4 WINDOW parent
4 WINDOW window
2 INT16 x
2 INT16 y
2 CARD16 width
2 CARD16 height
2 CARD16 border-width
1 BOOL override-redirect
DestroyNotify
1 17    code
1    unused
2 CARD16    sequence number
4 WINDOW    event
4 WINDOW    window
20    unused

UnmapNotify
1 18    code
1    unused
2 CARD16    sequence number
4 WINDOW    event
4 WINDOW    window
1 BOOL    from-configure
19    unused

MapNotify
1 19    code
1    unused
2 CARD16    sequence number
4 WINDOW    event
4 WINDOW    window
1 BOOL    override-redirect
19    unused

MapRequest
1 20    code
1    unused
2 CARD16    sequence number
4 WINDOW    parent
4 WINDOW    window
20    unused

ReparentNotify
1 21    code
1    unused
2 CARD16    sequence number
4 WINDOW    event
4 WINDOW    window
4 WINDOW    parent
2 INT16    x
2 INT16    y
1 BOOL    override-redirect
11    unused

ConfigureNotify
1 22    code
1    unused
2 CARD16    sequence number
4 WINDOW    event
4 WINDOW    window
4 WINDOW    above-sibling
0    None
2 INT16    x
2 INT16    y
2 CARD16    width
2 CARD16    height
2 CARD16    border-width
ConfigureRequest
1 23
1 code
1 stack-mode
0 Above
1 Below
2 TopIf
3 BottomIf
4 Opposite
2 CARD16 sequence number
4 WINDOW parent
4 WINDOW window
4 WINDOW sibling
0 None
2 INT16 x
2 INT16 y
2 CARD16 width
2 CARD16 height
2 CARD16 border-width
2 BITMASK value-mask
 #x0001 x
 #x0002 y
 #x0004 width
 #x0008 height
 #x0010 border-width
 #x0020 sibling
 #x0040 stack-mode
4 unused

GravityNotify
1 24
1 unused
2 CARD16 sequence number
4 WINDOW event
4 WINDOW window
2 INT16 x
2 INT16 y
16 unused

ResizeRequest
1 25
1 unused
2 CARD16 sequence number
4 WINDOW window
2 CARD16 width
2 CARD16 height
20 unused

CirculateNotify
1 26
1 unused
2 CARD16 sequence number
4 WINDOW event
4 WINDOW window
4 WINDOW unused
1 place
0 Top
1 Bottom
15 unused
**CirculateRequest**
```plaintext
1  27  code
1  unused
2  CARD16  sequence number
4  WINDOW  parent
4  WINDOW  window
4  unused
1  place
   0  Top
   1  Bottom
15  unused
```

**PropertyNotify**
```plaintext
1  28  code
1  unused
2  CARD16  sequence number
4  WINDOW  window
4  ATOM  atom
4  TIMESTAMP  time
1  state
   0  NewValue
   1  Deleted
15  unused
```

**SelectionClear**
```plaintext
1  29  code
1  unused
2  CARD16  sequence number
4  TIMESTAMP  time
4  WINDOW  owner
4  ATOM  selection
16  unused
```

**SelectionRequest**
```plaintext
1  30  code
1  unused
2  CARD16  sequence number
4  TIMESTAMP  time
   0  CurrentTime
4  WINDOW  owner
4  WINDOW  requestor
4  ATOM  selection
4  ATOM  target
4  ATOM  property
   0  None
16  unused
```

**SelectionNotify**
```plaintext
1  31  code
1  unused
2  CARD16  sequence number
4  TIMESTAMP  time
   0  CurrentTime
4  WINDOW  requestor
4  ATOM  selection
4  ATOM  target
4  ATOM  property
   0  None
8  unused
```

**ColormapNotify**
```plaintext
1  32  code
```
1 unused
2 CARD16 sequence number
4 WINDOW window
4 COLORMAP colormap
0 None
1 BOOL new
1 state
0 Uninstalled
1 Installed
18 unused

ClientMessage
1 33 code
1 CARD8 format
2 CARD16 sequence number
4 WINDOW window
4 ATOM type
20 data

MappingNotify
1 34 code
1 unused
2 CARD16 sequence number
1 request
0 Modifier
1 Keyboard
2 Pointer
1 KEYCODE first-keycode
1 CARD8 count
25 unused
Glossary

Access control list
X maintains a list of hosts from which client programs can be run. By default, only programs on the local host and hosts specified in an initial list read by the server can use the display. Clients on the local host can change this access control list. Some server implementations can also implement other authorization mechanisms in addition to or in place of this mechanism. The action of this mechanism can be conditional based on the authorization protocol name and data received by the server at connection setup.

Active grab
A grab is active when the pointer or keyboard is actually owned by the single grabbing client.

Ancestors
If W is an inferior of A, then A is an ancestor of W.

Atom
An atom is a unique ID corresponding to a string name. Atoms are used to identify properties, types, and selections.

Background
An InputOutput window can have a background, which is defined as a pixmap. When regions of the window have their contents lost or invalidated, the server will automatically tile those regions with the background.

Backing store
When a server maintains the contents of a window, the pixels saved off screen are known as a backing store.

Bit gravity
When a window is resized, the contents of the window are not necessarily discarded. It is possible to request that the server relocate the previous contents to some region of the window (though no guarantees are made). This attraction of window contents for some location of a window is known as bit gravity.

Bit plane
When a pixmap or window is thought of as a stack of bitmaps, each bitmap is called a bit plane or plane.

Bitmap
A bitmap is a pixmap of depth one.

Border
An InputOutput window can have a border of equal thickness on all four sides of the window. A pixmap defines the contents of the border, and the server automatically maintains the contents of the border. Exposure events are never generated for border regions.

Button grabbing
Buttons on the pointer may be passively grabbed by a client. When the button is pressed, the pointer is then actively grabbed by the client.

Byte order
For image ( pixmap/ bitmap) data, the server defines the byte order, and clients with different native byte ordering must swap bytes as necessary. For all other parts of the protocol, the client defines the byte order, and the server swaps bytes as necessary.

Children
The children of a window are its first-level subwindows.
Client
An application program connects to the window system server by some interprocess communication path, such as a TCP connection or a shared memory buffer. This program is referred to as a client of the window system server. More precisely, the client is the communication path itself; a program with multiple paths open to the server is viewed as multiple clients by the protocol. Resource lifetimes are controlled by connection lifetimes, not by program lifetimes.

Clipping region
In a graphics context, a bitmap or list of rectangles can be specified to restrict output to a particular region of the window. The image defined by the bitmap or rectangles is called a clipping region.

Colormap
A colormap consists of a set of entries defining color values. The colormap associated with a window is used to display the contents of the window; each pixel value indexes the colormap to produce RGB values that drive the guns of a monitor. Depending on hardware limitations, one or more colormaps may be installed at one time, so that windows associated with those maps display with correct colors.

Connection
The interprocess communication path between the server and client program is known as a connection. A client program typically (but not necessarily) has one connection to the server over which requests and events are sent.

Containment
A window “contains” the pointer if the window is viewable and the hotspot of the cursor is within a visible region of the window or a visible region of one of its inferiors. The border of the window is included as part of the window for containment. The pointer is “in” a window if the window contains the pointer but no inferior contains the pointer.

Coordinate system
The coordinate system has the X axis horizontal and the Y axis vertical, with the origin [0, 0] at the upper left. Coordinates are integral, in terms of pixels, and coincide with pixel centers. Each window and pixmap has its own coordinate system. For a window, the origin is inside the border at the inside upper left.

Cursor
A cursor is the visible shape of the pointer on a screen. It consists of a hot spot, a source bitmap, a shape bitmap, and a pair of colors. The cursor defined for a window controls the visible appearance when the pointer is in that window.

Depth
The depth of a window or pixmap is the number of bits per pixel that it has. The depth of a graphics context is the depth of the drawables it can be used in conjunction with for graphics output.

Device
Keyboards, mice, tablets, track-balls, button boxes, and so on are all collectively known as input devices. The core protocol only deals with two devices, “the keyboard” and “the pointer.”

DirectColor
DirectColor is a class of colormap in which a pixel value is decomposed into three separate subfields for indexing. The first subfield indexes an array to produce red intensity values. The second subfield indexes a second array to produce blue intensity values. The third subfield indexes a third array to produce green intensity values. The RGB values can be changed dynamically.

Display
A server, together with its screens and input devices, is called a display.

Drawable
Both windows and pixmaps can be used as sources and destinations in graphics operations. These windows and pixmaps are collectively known as drawables. However, an InputOnly window cannot be used as a source or destination in a graphics operation.
Event
Clients are informed of information asynchronously by means of events. These events can be generated either asynchronously from devices or as side effects of client requests. Events are grouped into types. The server never sends events to a client unless the client has specifically asked to be informed of that type of event. However, other clients can force events to be sent to other clients. Events are typically reported relative to a window.

Event mask
Events are requested relative to a window. The set of event types that a client requests relative to a window is described by using an event mask.

Event synchronization
There are certain race conditions possible when demultiplexing device events to clients (in particular deciding where pointer and keyboard events should be sent when in the middle of window management operations). The event synchronization mechanism allows synchronous processing of device events.

Event propagation
Device-related events propagate from the source window to ancestor windows until some client has expressed interest in handling that type of event or until the event is discarded explicitly.

Event source
The window the pointer is in is the source of a device-related event.

Exposure event
Servers do not guarantee to preserve the contents of windows when windows are obscured or reconfigured. Exposure events are sent to clients to inform them when contents of regions of windows have been lost.

Extension
Named extensions to the core protocol can be defined to extend the system. Extension to output requests, resources, and event types are all possible and are expected.

Focus window
The focus window is another term for the input focus.

Font
A font is a matrix of glyphs (typically characters). The protocol does no translation or interpretation of character sets. The client simply indicates values used to index the glyph array. A font contains additional metric information to determine interglyph and interline spacing.

GC, GContext
GC and gcontext are abbreviations for graphics context.

Glyph
A glyph is an image, typically of a character, in a font.

Grab
Keyboard keys, the keyboard, pointer buttons, the pointer, and the server can be grabbed for exclusive use by a client. In general, these facilities are not intended to be used by normal applications but are intended for various input and window managers to implement various styles of user interfaces.

Graphics context
Various information for graphics output is stored in a graphics context such as foreground pixel, background pixel, line width, clipping region, and so on. A graphics context can only be used with drawables that have the same root and the same depth as the graphics context.

Gravity
See bit gravity and window gravity.
GrayScale

GrayScale can be viewed as a degenerate case of PseudoColor, in which the red, green, and blue values in any given colormap entry are equal, thus producing shades of gray. The gray values can be changed dynamically.

Hotspot

A cursor has an associated hotspot that defines the point in the cursor corresponding to the coordinates reported for the pointer.

Identifier

An identifier is a unique value associated with a resource that clients use to name that resource. The identifier can be used over any connection.

Inferiors

The inferiors of a window are all of the subwindows nested below it: the children, the children’s children, and so on.

Input focus

The input focus is normally a window defining the scope for processing of keyboard input. If a generated keyboard event would normally be reported to this window or one of its inferiors, the event is reported normally. Otherwise, the event is reported with respect to the focus window. The input focus also can be set such that all keyboard events are discarded and such that the focus window is dynamically taken to be the root window of whatever screen the pointer is on at each keyboard event.

Input manager

Control over keyboard input is typically provided by an input manager client.

InputOnly window

An InputOnly window is a window that cannot be used for graphics requests. InputOnly windows are invisible and can be used to control such things as cursors, input event generation, and grabbing. InputOnly windows cannot have InputOutput windows as inferiors.

InputOutput window

An InputOutput window is the normal kind of opaque window, used for both input and output. InputOutput windows can have both InputOutput and InputOnly windows as inferiors.

Key grabbing

Keys on the keyboard can be passively grabbed by a client. When the key is pressed, the keyboard is then actively grabbed by the client.

Keyboard grabbing

A client can actively grab control of the keyboard, and key events will be sent to that client rather than the client the events would normally have been sent to.

Keysym

An encoding of a symbol on a keycap on a keyboard.

Mapped

A window is said to be mapped if a map call has been performed on it. Unmapped windows and their inferiors are never viewable or visible.

Modifier keys

Shift, Control, Meta, Super, Hyper, Alt, Compose, Apple, CapsLock, ShiftLock, and similar keys are called modifier keys.

Monochrome

Monochrome is a special case of StaticGray in which there are only two colormap entries.

Obscure

A window is obscured if some other window obscures it. Window A obscures window B if both are viewable InputOutput windows, A is higher in the global stacking order, and the rectangle defined by the outside edges of A intersects the rectangle defined by the outside edges of B. Note the distinction between obscure and occludes. Also note that window borders are included in the calculation and that a window can be obscured and yet still have visible regions.
Oclude
A window is occluded if some other window occludes it. Window A occludes window B if both are mapped, A is higher in the global stacking order, and the rectangle defined by the outside edges of A intersects the rectangle defined by the outside edges of B. Note the distinction between occludes and obscures. Also note that window borders are included in the calculation.

Padding
Some padding bytes are inserted in the data stream to maintain alignment of the protocol requests on natural boundaries. This increases ease of portability to some machine architectures.

Parent window
If C is a child of P, then P is the parent of C.

Passive grab
Grabbing a key or button is a passive grab. The grab activates when the key or button is actually pressed.

Pixel value
A pixel is an N-bit value, where N is the number of bit planes used in a particular window or pixmap (that is, N is the depth of the window or pixmap). For a window, a pixel value indexes a colormap to derive an actual color to be displayed.

Pixmap
A pixmap is a three-dimensional array of bits. A pixmap is normally thought of as a two-dimensional array of pixels, where each pixel can be a value from 0 to (2^N)-1 and where N is the depth (z axis) of the pixmap. A pixmap can also be thought of as a stack of N bitmaps.

Plane
When a pixmap or window is thought of as a stack of bitmaps, each bitmap is called a plane or bit plane.

Plane mask
Graphics operations can be restricted to only affect a subset of bit planes of a destination. A plane mask is a bit mask describing which planes are to be modified. The plane mask is stored in a graphics context.

Pointer
The pointer is the pointing device attached to the cursor and tracked on the screens.

Pointer grabbing
A client can actively grab control of the pointer. Then button and motion events will be sent to that client rather than the client the events would normally have been sent to.

Pointing device
A pointing device is typically a mouse, tablet, or some other device with effective dimensional motion. There is only one visible cursor defined by the core protocol, and it tracks whatever pointing device is attached as the pointer.

Property
Windows may have associated properties, which consist of a name, a type, a data format, and some data. The protocol places no interpretation on properties. They are intended as a general-purpose naming mechanism for clients. For example, clients might use properties to share information such as resize hints, program names, and icon formats with a window manager.

Property list
The property list of a window is the list of properties that have been defined for the window.

PseudoColor
PseudoColor is a class of colormap in which a pixel value indexes the colormap to produce independent red, green, and blue values; that is, the colormap is viewed as an array of triples (RGB values). The RGB values can be changed dynamically.
Redirecting control

Window managers (or client programs) may want to enforce window layout policy in various ways. When a client attempts to change the size or position of a window, the operation may be redirected to a specified client rather than the operation actually being performed.

Reply

Information requested by a client program is sent back to the client with a reply. Both events and replies are multiplexed on the same connection. Most requests do not generate replies, although some requests generate multiple replies.

Request

A command to the server is called a request. It is a single block of data sent over a connection.

Resource

Windows, pixmaps, cursors, fonts, graphics contexts, and colormaps are known as resources. They all have unique identifiers associated with them for naming purposes. The lifetime of a resource usually is bounded by the lifetime of the connection over which the resource was created.

RGB values

Red, green, and blue (RGB) intensity values are used to define color. These values are always represented as 16-bit unsigned numbers, with 0 being the minimum intensity and 65535 being the maximum intensity. The server scales the values to match the display hardware.

Root

The root of a pixmap, colormap, or graphics context is the same as the root of whatever drawable was used when the pixmap, colormap, or graphics context was created. The root of a window is the root window under which the window was created.

Root window

Each screen has a root window covering it. It cannot be reconfigured or unmapped, but it otherwise acts as a full-fledged window. A root window has no parent.

Save set

The save set of a client is a list of other clients' windows that, if they are inferiors of one of the client's windows at connection close, should not be destroyed and that should be remapped if currently unmapped. Save sets are typically used by window managers to avoid lost windows if the manager terminates abnormally.

Scanline

A scanline is a list of pixel or bit values viewed as a horizontal row (all values having the same y coordinate) of an image, with the values ordered by increasing x coordinate.

Scanline order

An image represented in scanline order contains scanlines ordered by increasing y coordinate.

Screen

A server can provide several independent screens, which typically have physically independent monitors. This would be the expected configuration when there is only a single keyboard and pointer shared among the screens.
Selection
A selection can be thought of as an indirect property with dynamic type; that is, rather than having the property stored in the server, it is maintained by some client (the “owner”). A selection is global in nature and is thought of as belonging to the user (although maintained by clients), rather than as being private to a particular window subhierarchy or a particular set of clients. When a client asks for the contents of a selection, it specifies a selection “target type”. This target type can be used to control the transmitted representation of the contents. For example, if the selection is “the last thing the user clicked on” and that is currently an image, then the target type might specify whether the contents of the image should be sent in XY format or Z format. The target type can also be used to control the class of contents transmitted; for example, asking for the “looks” (fonts, line spacing, indentation, and so on) of a paragraph selection rather than the text of the paragraph. The target type can also be used for other purposes. The protocol does not constrain the semantics.

Server
The server provides the basic windowing mechanism. It handles connections from clients, multiplexes graphics requests onto the screens, and demultiplexes input back to the appropriate clients.

Server grabbing
The server can be grabbed by a single client for exclusive use. This prevents processing of any requests from other client connections until the grab is completed. This is typically only a transient state for such things as rubber-banding, pop-up menus, or to execute requests indivisibly.

Sibling
Children of the same parent window are known as sibling windows.

Stacking order
Sibling windows may stack on top of each other. Windows above other windows both obscure and occlude those lower windows. This is similar to paper on a desk. The relationship between sibling windows is known as the stacking order.

StaticColor
StaticColor can be viewed as a degenerate case of PseudoColor in which the RGB values are predefined and read-only.

StaticGray
StaticGray can be viewed as a degenerate case of GrayScale in which the gray values are predefined and read-only. The values are typically linear or near-linear increasing ramps.

Stipple
A stipple pattern is a bitmap that is used to tile a region that will serve as an additional clip mask for a fill operation with the foreground color.

String Equivalence
Two ISO Latin-1 STRING8 values are considered equal if they are the same length and if corresponding bytes are either equal or are equivalent as follows: decimal values 65 to 90 inclusive (characters “A” to “Z”) are pairwise equivalent to decimal values 97 to 122 inclusive (characters “a” to “z”), decimal values 192 to 214 inclusive (characters “A grave” to “O diaeresis”) are pairwise equivalent to decimal values 224 to 246 inclusive (characters “a grave” to “o diaeresis”), and decimal values 216 to 222 inclusive (characters “O oblique” to “THORN”) are pairwise equivalent to decimal values 246 to 254 inclusive (characters “o oblique” to “thorn”).

Tile
A pixmap can be replicated in two dimensions to tile a region. The pixmap itself is also known as a tile.
Timestamp

A timestamp is a time value, expressed in milliseconds. It typically is the time since the last server reset. Timestamp values wrap around (after about 49.7 days). The server, given its current time is represented by timestamp T, always interprets timestamps from clients by treating half of the timestamp space as being earlier in time than T and half of the timestamp space as being later in time than T. One timestamp value (named CurrentTime) is never generated by the server. This value is reserved for use in requests to represent the current server time.

TrueColor

TrueColor can be viewed as a degenerate case of DirectColor in which the subfields in the pixel value directly encode the corresponding RGB values; that is, the colormap has predefined read-only RGB values. The values are typically linear or near-linear increasing ramps.

Type

A type is an arbitrary atom used to identify the interpretation of property data. Types are completely uninterpreted by the server and are solely for the benefit of clients.

Viewable

A window is viewable if it and all of its ancestors are mapped. This does not imply that any portion of the window is actually visible. Graphics requests can be performed on a window when it is not viewable, but output will not be retained unless the server is maintaining backing store.

Visible

A region of a window is visible if someone looking at the screen can actually see it; that is, the window is viewable and the region is not occluded by any other window.

Window gravity

When windows are resized, subwindows may be repositioned automatically relative to some position in the window. This attraction of a subwindow to some part of its parent is known as window gravity.

Window manager

Manipulation of windows on the screen and much of the user interface (policy) is typically provided by a window manager client.

XYFormat

The data for a pixmap is said to be in XY format if it is organized as a set of bitmaps representing individual bit planes, with the planes appearing from most-significant to least-significant in bit order.

ZFormat

The data for a pixmap is said to be in Z format if it is organized as a set of pixel values in scanline order.
# Table of Contents

Acknowledgments ........................................ iii
1. Protocol Formats .................................... 1
2. Syntactic Conventions ............................... 1
3. Common Types ........................................ 2
4. Errors .................................................. 4
5. Keyboards .............................................. 6
6. Pointers .................................................. 7
7. Predefined Atoms ...................................... 7
8. Connection Setup ...................................... 8
9. Requests ............................................... 13
10. Connection Close ...................................... 76
11. Events .................................................. 76
12. Flow Control and Concurrency ...................... 89
Appendix A – KEYSYM Encoding ....................... 90
Appendix B – Protocol Encoding ....................... 110
Glossary ................................................... 158
Index ....................................................... 166