VEXlib
Video Extension to X Client Library
Version 1.10

Scott Hennes

Tektronix, Inc.

Section 1

Introduction

This document describes **VEXlib**, the client library for the Video Extension to X. It assumes the reader is familiar with Xlib and with the VEX Protocol. This initial draft does not attempt to duplicate much of the VEX Protocol document and the reader must rely on that document to understand the semantics of many VEXlib routines and data structures.

VEXlib does not have an explicit initialization routine. Each routine will attempt to initialize the library if it is not already initialized. A client can determine whether or not the VEX extension is available in a server by using the core routine, **XQueryExtension.** This routine must also be used to obtain the VEX event and error base numbers. The name of the extension is "VEX".

Library initialization does an implicit call to **XVEXQueryVideo**. Client access to information sent in the server reply is through these routines: **XVEXOverlapsOfScreen**, **XVEXTimestamps**, **XVEXGetVisualInfo**, **XVEXGetVins**, **XVEXGetVouts**, **XVEXGetVdevs**, and **XVEXGetConnectState** All information returned by these routines remains valid for the life of the connection except for **XVEXTimestamps**, **XVEXGetConnectState**, and **XVEXGetVdevs.** Timestamp values change when the library receives VEXConnectivity events and **XVEXQueryVideo** replies. Device connection information should be considered out of date when the client receives a VEXConnectivity event. Valid data may be obtained by calling **XVEXGetConnectState** after the client receives a VEXConnectivity event whose count field is zero. Video Device information should be considered out of date when the client receives a VEXChange event. Valid data may be obtained by calling **XVEXQueryVideo** and then **XVEXGetVdevs.**

Section 2

Data Structures

These data structures are used by routines in VEXlib:

```
typedef XID VideoIn;
typedef XID VideoOut;
typedef XID VideoDevice;
typedef XID VideoIO;
typedef XID VideoID;
typedef XID VideoReference;

typedef struct {
        long            numerator;
        long            denominator;
} XVEXFraction;

typedef struct {
        long            num_base;
        long            num_increment;
        long            num_limit;
        int             num_type;               /* VEXGeometricRange or VEXLinearRange */
        long            denom_base;
        long            denom_increment;
        long            denom_limit;
        int             denom_type;             /* VEXGeometricRange or VEXLinearRange */
} XVEXFractionRange;

typedef struct {
        Atom            name;                   /* Control name */
        int             setting_format;         /* 8, 16, or 32 */
        int             description_format;     /* 8, 16, or 32 */
        unsigned long   setting_length;         /* setting length (bytes) */
        unsigned long   description_length;     /* description length (bytes) */
        union {         /* description data */
          char          *p8list;
          short         *p16list;
          long          *p32list;
        } value
} XVEXControl;
```

```
typedef struct {
        Atom                name;                 /* Control name */
        VideoID             id;
        int                 format;               /* 8, 16, or 32 */
        unsigned long       length;               /* setting length (bytes) */
        union {             /* setting  data */
           char             *p8list;
           short            *p16list;
           long             *p32list;
        } value
} XVEXSetting;

typedef struct {
        XRectangle          base;
        XRectangle          limit;
        int                 x_inc;
        int                 y_inc;
        int                 width_inc;
        int                 height_inc;
        int                 type;                 /* VEXGeometricRange or VEXLinearRange */
} XVEXRectangleRange;

typedef struct {
        XVEXFraction        frame_rate;           /* max nominal rate (frames/sec) */
        XVEXRectangleRange  src;                  /* set of possible source rectangles */
        XVEXRectangleRange  dest;                 /* set of possible dest rectangles */
        XVEXFractionRange   x_scale;              /* set of dst/src width ratios */
        XVEXFractionRange   y_scale;              /* set of dst/src height ratios */
        Bool                                      identity_aspect;/* set of dst/src height ratios */
} XVEXPlacement;

typedef struct {
        XVEXFraction        frame_rate;           /* frames/sec in signal format */
        XVEXFraction        field_rate;           /* fields/sec in signal format */
        int                 width;                /* signal width (in pixels) */
        int                 height;               /* signal height (in pixels) */
        int                 concurrent_use;       /* max simultaneous use */
        int                 nplacement;           /* number of placements */
        XVEXPlacement       *placement;           /* placement list */
} XVEXGeometry;

typedef struct {
        Window              window;               /* identifies client requesting ownership */
        VideoID             vid;                  /* resource to be owned */
} XVEXOwner;
```

```
typedef struct {
        Bool                composite;        /* True if RenderVideo will modify pixels or
                                               * if CaptureGraphics will blend video and
                                               *  graphics */
        Bool                opaque;           /* True if pixels cannot be interpreted
                                               * by a Core type */
        VisualID            visualid;         /* video window visual ID */
} XVEXWindowModel;

typedef struct {
        int                 depth;            /* pixmap depth */
        int                 nvisualid;
        VisualID            *visualid;        /* visual IDs which may be used for
                                               * colormaps in CaptureGraphics */
        unsigned long       red_mask;
        unsigned long       green_mask;
        unsigned long       blue_mask;
} XVEXPixmapModel;

typedef struct {
        int                 nwindow_models;
        XVEXWindowModel     *window_models;
        int                 npixmap_models;
        XVEXPixmapModel     *pixmap_models;
} XVEXModelInfo;

typedef struct {
        VideoReference      vref;             /* VideoIn reference ID */
        int                 ncontrols;
        XVEXControl         *controls;
        XVEXGeometry        geometry;
        XVEXModelInfo       model_info;
        XRectangle          clip_size;
} XVEXVin;

typedef struct {
        VideoReference      vref;             /* VideoOut reference ID */
        int                 ncontrols;
        XVEXControl         *controls;
        XVEXGeometry        geometry;
        XVEXModelInfo       model_info;
} XVEXVout;

typedef struct {
        VideoDevice         id;               /* VideoDevice id */
        int                 ncontrols;
        XVEXControl         *controls;
} XVEXVdev;

typedef struct {
        VideoID             src;
        VideoID             dst;
        long                matrix1;
        long                matrix2;
} XVEXTransition;
```

```
typedef struct {
        int nrow;                       /* number of Rows in connection matrix */
        int ncol;                       /* number of Columns in connection matrix */
        XID *row;                       /* ID's labelling the rows of the matrix */
        XID *col;                       /* ID's labelling the columns of the matrix */
} XVEXConnectLabel;

typedef struct {
        unsigned char row;      /* connectivity for devices or videoIns
                                 * in the row */
        unsigned char col;      /* connectivity for devices or videoOuts
                                 * in the column */
} XVEXConnectCell;

typedef struct {
        XVEXConnectCell  **matrix       /* Matrix representing a possible set
                                         * of connections among video inputs,
                                         * outputs and devices */
} XVEXConnectMatrix;
```

Section 3

VEX Protocol Functions

1. Video Manager Routines

XVEXChangeOwnership(*display*, *owners*, *nowners*)
    Display **display*;
    XVEXOwner *owners*[];
    unsigned long *nowners*;

display Specifies the connection to the X server.

owners Specifies array of ownerships to set.

nowners Specifies number of entries in *owners* array.

For each member of the *owners* list, the client that created the specified window is given ownership of the specified resource. [See the VEX Protocol ChangeOwnership request description]

2. Device Controls

XVEXChangeControls(*display*, *settings*, *nsettings*)
    Display **display*;
    XVEXSetting *settings*[];
    unsigned long *nsettings*;

display Specifies the connection to the X server.

settings Specifies array of settings to set.

nsettings Specifies number of entries in *settings* array.

Changes the setting of each control named in each member of the *settings* array. The *settings* array may be created by **XVEXCreateSettingList**, but this is not required. [See the VEX Protocol ChangeVideoControls request description]

XVEXSetting *XVEXQueryControls(*display*, *vid*, *controls*, *ncontrols*, *nsettings_return*)
    Display **display*;
    VideoID *vid*;
    Atom *controls*[];
    unsigned long *ncontrols*;
    unsigned long **nsettings_return*;

display Specifies the connection to the X server.

vid Specifies the video id whose settings are requested.

controls Specifies the controls whose settings are requested.

ncontrols Specifies number of entries in *controls* array.

nsettings_return Returns number of entries in the returned XVEXSetting array.

Returns the current setting associated with each of the controls in the *controls* array for the video device named by *vid*. To free the data returned by this function, use **XFree**. [See the VEX Protocol QueryVideoControls request description]

3. Video Input and Video Output

VideoIO XVEXCreateVideo(*display*, *reference*)
    Display **display*;
    VideoReference *reference*;

display Specifies the connection to the X server.

reference Specifies reference for which a VIDEOIO alias is requested.

Returns a VideoIO corresponding to the given VideoReference. If the VEX extension cannot be initialized, the returned value is None. [See the VEX Protocol CreateVideo request description]

XVEXRenderVideo(*display*, *src*, *dest*, *src_x*, *src_y*, *src_width*, *src_height*, *dest_x*, *dest_y*, *dest_width*, *dest_height*, *full_motion*, *priority*, *subwindow_mode*)
Display **display*;
VideoIn *src*;
Drawable *dest*;
int *src_x*, *src_y*;
unsigned int *src_width*, *src_height*;
int *dest_x*, *dest_y*;
unsigned int *dest_width*, *dest_height*;
Bool *full_motion*;
int *priority*;
int *subwindow_mode*;

display Specifies the connection to the X server.

src Specifies the video source.

dest Specifies drawable which accepts the video signal.

src_x
src_y Specify the x and y coordinates of the upper-left corner of the video source rectangle.

src_width
src_height Specify the width and height of the video source rectangle.

dest_x
dest_y Specify the x and y coordinates of the upper-left corner of the destination drawable rectangle. The coordinates are relative to the origin of the destination drawable.

dest_width
dest_height Specify the width and height of the destination drawable rectangle.

full_motion Specifies whether the video input signal is continually updated.

priority Specifies the percentage of video bandwidth that the server should try to preserve.

subwindow_mode
Specifies the treatment of the destination window's children. Ignored if destination is a pixmap. You can pass ClipByChildren or IncludeInferiors.

The specified region of the source video signal is decoded and directed to the destination drawable. [See the VEX Protocol RenderVideo request description]

XVEXCaptureGraphics(*display*, *src*, *dest*, *cmap*, *src_x*, *src_y*, *src_width*, *src_height*, *dest_x*, *dest_y*, *dest_width*, *dest_height*, *full_motion*, *priority*, *subwindow_mode*)
Display **display*;
Drawable *src*;
VideoOut *dest*;
Colormap *cmap*;
int *src_x*, *src_y*;
unsigned int *src_width*, *src_height*;
int *dest_x*, *dest_y*;
unsigned int *dest_width*, *dest_height*;
Bool *full_motion*;
int *priority*;
int *subwindow_mode*;

display Specifies the connection to the X server.

src Specifies the drawable source.

dest Specifies the video output which accepts the video signal.

cmap Specifies the colormap to use for a Pixmap drawable. Must be None for window drawable.

src_x
src_y Specify the x and y coordinates of the upper-left corner of the source drawable rectangle. The coordinates are relative to the origin of the source drawable.

src_width
src_height Specify the width and height of the drawable source rectangle.

dest_x
dest_y Specify the x and y coordinates of the upper-left corner of the destination video rectangle.

dest_width
dest_height Specify the width and height of the destination video rectangle.

full_motion Specifies whether the video input signal is continually updated.

priority Specifies the percentage of video bandwidth that the server should try to preserve.

subwindow_mode
Specifies the treatment of the source window’s children. Ignored if source is a pixmap. You can pass ClipByChildren or IncludeInferiors.

The specified region of the drawable source is encoded into a video signal by the destination encoder. [See the VEX Protocol CaptureGraphics request description]

XVEXStopVideo(*display*, *action_mask*, *drawables*, *ndrawables*)
Display **display*;
int *action_mask*;
Drawable *drawables*[];
unsigned long *ndrawables*;

display	Specifies the connection to the X server.
action_mask	Specifies the action to take on the drawables list. The mask values is a bitwise OR of these masks: VEXStopRenderVideo and VEXStopCaptureVideo.
drawables	Specifies array of drawables to stop.
ndrawables	Specifies number of entries in *drawables* array.

The video actions previously requested on any of the drawables given in the *drawables* array are stopped, even if one of the drawables produces an error. [See the VEX Protocol StopVideo request description]

XVEXSelectEvents(*display*, *id*, *event_mask*)
  Display **display*;
  XID *id*;
  unsigned long *event_mask*;

display Specifies the connection to the X server.

id Specifies the VideoID or Drawable on which events are selected.

event_mask Specifies the events to select.

The following table lists the event mask constants you can pass to the event_mask argument and the circumstances in which you would want to specify the event mask:

Event Mask	Circumstances
NoEventMask	No events wanted
VEXRedirectMask	VEXRequest events wanted on a screen (should be used only by Video Manager)
VEXConnectivityMask	VEXConnectivity events wanted on a screen
VEXSyncMask	VEXSync events wanted
VEXControlMask	VEXControl events wanted
VEXViolationMask	VEXViolation events wanted
VEXOverrideMask	VEXOverride events wanted

VEXRequest, VEXConnectivity, VEXViolation, and VEXOverride events can be selected only on a Drawable. VEXSync events can be selected only on a VideoIO and VEXControl events can be selected only on a VideoID. Selections are not cumulative; if an event is not asserted in conjunction with a valid resource for that event (e.g. not asserting VEXOverrideMask when using a Drawable resource), the event will not be delivered on that resource regardless of earlier selections on that resource. VEXConnectivity and VEXRequest events are actually selected on the screen associated with the drawable id argument; to maintain these selections, VEXRedirectMask and VEXConnectivityMask must be asserted on each drawable id associated with that screen. [See the VEX Protocol SelectVideoEvents request description]

Status XVEXQueryVideo(*display*, *screen_number*)
  Display **display*;
  int *screen_number*;

display Specifies the connection to the X server.

screen_number Specifies the screen to be updated.

[See the VEX Protocol QueryVideo request description]

XVEXQueryVideo updates connectivity and device list information in VEXlib. It should be called by the client after receiving a XVEXConnectivityEvent or a XVEXChangeEvent. The client should get a new XVout array by calling **XVEXGetVouts** after calling **XVEXQueryVideo.** The client should get a new XVEXConnectState structure by calling **XVEXGetConnectState** after calling **XVEXQueryVideo.**

XVEXQueryVideo does an implicit selection for VEXChange events. This selection cannot be withdrawn.

Status return is non-zero if VEXlib can be initialized. Otherwise the return is zero and VEXlib data are not updated.

Section 4

Application Utility Functions

1. Extension Information Utilities

int XVEXBaseOpCode(*display*)
  Display **display*;

display Specifies the connection to the X server.

Returns the major opcode for the VEX extension if VEXlib can be initialized; otherwise, False is returned.

int XVEXBaseEvent(*display*)
  Display **display*;

display Specifies the connection to the X server.

Returns the base event type code for the VEX extension if VEXlib can be initialized; otherwise, False is returned.

int XVEXBaseError(*display*)
  Display **display*;

display Specifies the connection to the X server.

Returns the base error code for the VEX extension if VEXlib can be initialized; otherwise, False is returned.

2. Screen Information Utilities

Status XVEXOverlapsOfScreen(*display*, *screen_number*, *input_overlap*, *capture_overlap*, *io_overlap*)
  Display **display*;
  int *screen_number*;
  Bool **input_overlap*, **capture_overlap*, **io_overlap*;

display Specifies the connection to the X server.

screen_number Specifies the screen to be examined.

input_overlap Returns True if VIRs may overlap, False otherwise.

capture_overlap Returns True if VORs may overlap, False otherwise.

io_overlap Returns True if a VIR and a VOR may overlap, False otherwise.

Status return is non-zero if VEXlib can be initialized. Otherwise the return is zero and the return arguments are not updated. [See VEX Protocol QueryVideo request description]

Status XVEXTimestamps(*display*, *screen_number*, *video_change_time*, *video_connectivity_time*)
  Display **display*;
  int *screen_number*;
  unsigned long **video_change_time*;
  unsigned long **video_connectivity_time*;

display Specifies the connection to the X server.

screen_number Specifies the screen to be examined.

video_change_time
  Returns the most recent time returned by XVEXQueryVideo (the most recent time the list of video devices changed).

video_connectivity_time
  Returns the time of the most recent device network change caused by a ChangeConnectivity request (reported by a VideoConnectivityState event).

Status return is non-zero if VEXlib can be initialized. Otherwise the return is zero and the return arguments are not updated.

During the course of a connection to the X server, the list of devices may change and the connectivity of devices may change. This routine will report the most recent times for these changes. These times are set within the VEX library when VideoConnectivity events are received (video_connectivity_time) and when the XVEXQueryVideo reply is received (video_change_time).

XVisualInfo *XVEXGetVisualInfo(*display*, *vinfo_mask*, *vinfo_template*, *nitems_return*, *include_core_visuals*)
    Display **display*;
    long *vinfo_mask*;
    XVisualInfo **vinfo_template*;
    int **nitems_return*;
    Bool *include_core_visuals*;

display Specifies the connection to the X server.

vinfo_mask Specifies the visual mask value.

vinfo_template Specifies the visual attributes that are to be used in matching the visual structures.

nitems_return Returns the number of matching visual structures.

include_core_visuals
    Specifies which visuals lists should be searched. If the value is True, then both the Core and the VEX visuals lists are searched, otherwise, only the VEX list is searched.

XVEXGetVisualInfo function returns a list of visual structures that match the attributes specified by vinfo_template. If *include_core_visuals* is False, it returns only those visuals that are unique to VEX (not provided in the Core connection information), otherwise it returns visuals from either the Core list or the VEX list. Visuals returned by this routine are not guaranteed to have any video capability; association with video is determined by XVEXWindowModel structures returned by **XVEXGetVins** and **XVEXGetVouts**. If no visual structures match the template using the specified vinfo_mask, **XVEXGetVisualInfo** returns a NULL. To free the data returned by this function, use **XFree**. The choices for vinfo_mask and the definition for the XVisualInfo data structure are given in the Core Xlib specification (see **XGetVisualInfo** in Chapter 10, Application Utility Functions). Possible values for the class field of the XVisualInfo structure include VEXVideoGray and VEXVideoColor in addition to the Core class types.

XVEXVin *XVEXGetVins(*display*, *screen_number*, *nitems_return*)
Display **display*;
int *screen_number*;
int **nitems_return*;

display Specifies the connection to the X server.

screen_number Specifies the screen to be examined.

nitems_return Returns number of XVEXVin structures in the returned XVEXVin array.

Returns an array of XVEXVin structures for the named screen. The array remains valid for the life of the connection. Returns NULL if VEXlib cannot be initialized or cannot allocate space or if there are no VideoInputs for this screen. To free the data returned by this function, use **XFree**.

XVEXVout *XVEXGetVouts(*display*, *screen_number*, *nitems_return*)
Display **display*;
int *screen_number*;
int **nitems_return*;

display Specifies the connection to the X server.

screen_number Specifies the screen to be examined.

nitems_return Returns number of XVEXVout structures in the returned XVEXVout array.

Returns an array of XVEXVout structures for the named screen. The array remains valid for the life of the connection. Returns NULL if VEXlib cannot be initialized or cannot allocate space or if there are no VideoOutputs for this screen. To free the data returned by this function, use **XFree**.

XVEXVdev *XVEXGetVdevs(*display*, *screen_number*, *nitems_return*)
  Display **display*;
  int *screen_number*;
  int **nitems_return*;

display Specifies the connection to the X server.

screen_number Specifies the screen to be examined.

nitems_return Returns number of XVEXVdev structures in the returned XVEXVdev array.

Returns an array of XVEXVdev structures for the named screen. The information contained in the array will be out of date once a VideoChange event is received (because the device list maintained by the X server has changed). The client should call **XVEXQueryVideo** to update device lists and/or connectivity, and then call **XVEXGetVdevs** to get a new array of XVEXVdev structures. Returns NULL if VEXlib cannot be initialized or cannot allocate space or if there are no Video devices for this screen. To free the data returned by this function, use **XFree**.

XVEXConnectLabel *XVEXGetConnectLabel(*display*, *screen_number*)
  Display **display*;
  int *screen_number*;

display Specifies the connection to the X server.

screen_number Specifies the screen to be examined.

Returns a XVEXConnectLabel structure which gives the dimensions and row and colum labels of connectivity matrices returned by **XVEXGetConnectMatrices** and **XVEXGetConnectState**. Returns NULL if VEXlib cannot be initialized or cannot allocate space. To free the data returned by this function, use **XFree**. [See VEX Protocol QueryVideo request description]

unsigned char **XVEXGetConnectState(*display*, *screen_number*, *index_return*)
Display **display*;
int *screen_number*;
long **index_return*;

display Specifies the connection to the X server.

screen_number Specifies the screen to be examined.

index_return Returns the index of the initial state matrix for the current state. The index is for the list of **XVEXConnectMatrix** structures returned by **XVEXGetConnectMatrices**.

Returns a 2 dimensional array of **XVEXConnectCell** structures describing the current connectivity of the named screen. The data is valid only if the client has selected VEXConnectivity events on the named screen. This information is valid until a new XVEXConnectivityEvent is received. A client should call **XVEXGetConnectState** again to get new state information when a XVEXConnectivityEvent with count of 0 is received.

Returns NULL if VEX could not be initialized or if memory allocation failed. To free the data returned by this function, use **XFree**.

XVEXConnectMatrix *XVEXGetConnectMatrices(*display*, *screen_number*, *nitems_return*)
Display **display*;
int *screen_number*;
long **nitems_return*;

display Specifies the connection to the X server.

screen_number Specifies the screen to be examined.

nitems_return Returns the number of matrices in the array.

Returns a 3 dimensional array of XVEXConnectCell structures. This array describes the potential connectivity of all video inputs, outputs and devices. Each element of the returned array is a 2 dimensional array whose row and column labels are given by the current XVEXConnectLabel structure returned by **XVEXGetConnectLabel**. The value of each member of the XVEXConnectCell structure is one of: **VEXConnected**, **VEXNotConnected,** or the maximum potential connectivity. [See VEX Protocol QueryVideo request description]

XVEXConnectTransition *XVEXGetConnectTransitions(*display*, *screen_number*, *nitems_return*)
Display **display*;
int *screen_number*;
long **nitems_return*;

display Specifies the connection to the X server.

screen_number Specifies the screen to be examined.

nitems_return Returns the number of transitions in the array.

Returns an array of XVEXConnectTransition structures which describe the transitions between matrices in the array returned by **XVEXGetConnectMatrices.** [See VEX Protocol QueryVideo request description]

3. Setting List Utilities

XVEXSetting *XVEXCreateSettingList(*vid*, *control_list*, *ncontrol*)
VideoID *vid*;
XVEXControl **control_list*;
int *ncontrol*;

vid Specifies the video resource for the settings.

control_list Specifies the array of XVEXControl structures whose settings are desired

ncontrol Specifies the number of elements in the *control_list*

Creates an array of XVEXSettings which correspond one-to-one with the input array of XVEXControls. Constant data are copied from the XVEXControls to the XVEXSettings: name, format, length. The setting id's are set to *vid* . Space is allocated for value data for each XVEXSetting structure and the pointer element points to this data.

If VEXlib cannot allocate space, the return will be NULL. To free the data returned by this function, use **XFree**.

4. Range Searching Utilities

Bool XVEXMatchFraction(*given*, *ranges*, *nranges*, *lower*, *higher*)
  XVEXFraction **given*;
  XVEXFractionRange **ranges*;
  int *nranges*;
  XVEXFraction **lower*;
  XVEXFraction **higher*;

given Specifies the fraction to be checked.

ranges Specifies the fraction range list to be compared against.

nranges Specifies the number of fraction ranges in the list.

lower Returns the nearest lower fraction in the range.

higher Returns the nearest lower higher in the range.

Returns True if *given* is numerically equal to a fraction in one of the fraction ranges in *ranges*, in which case *lower* and *higher* are returned numerically equal to *given*, with the smallest values for numerator and denominator within *range*. For example, if *given* is 22/33, *lower* and *higher* could be 2/3 even if 22/33 is also in *range*.

Returns False otherwise. If *given* is between the highest and lowest fractions in all the fraction ranges in *ranges*, then *lower* is the closest fraction in *ranges* of less value and *higher* is the closest fraction in *ranges* of greater value. If *given* is below the lowest fraction in *ranges*, both *lower* and *higher* are that lowest fraction, while if above both are the highest.

One use of this function is to make sure a src and dest match according to the x_scale and y_scale ranges, so that the **XVEXRenderVideo** or **XVEXCaptureGraphics** functions do not get a VideoViolation event.

Bool XVEXMatchRectangle(*given*, *range*, *enclosed*, *enclosing*)
   XRectangle **given*;
   XVEXRectangleRange **range*;
   XRectangle **enclosed*;
   XRectangle **enclosing*;

given	Specifies a given rectangle.
range	Specifies a rectangle range.
enclosed	Returns the largest rectangle in the range that is enclosed by the given rectangle, if possible.
enclosing	Returns the smallest rectangle in the range that encloses the given rectangle, if possible.

Returns True if *given* exactly matches a rectangle in *range*. Then both *enclosed* and *enclosing* are returned equal to *given*.

Returns false otherwise. If *given* is within the extents of *range*, then *enclosed* is the largest rectangle in *range* that is completely enclosed by *given* and *enclosing* is the smallest rectangle in *range* that completely encloses *given*. But if *given* is partially or completely outside the extents of *range*, then *enclosed* and *enclosing* are as close as possible to being enclosed/enclosing while still being rectangles in *range*.

This function can be used to find either the proper enclosing area around a desired source area or the largest proper destination area within a given boundary, for rendering video or capturing graphics.

Section 5

Events

For each event type, a corresponding structure is declared in <X11/VEX/VEXlib.h**>.** In each event, the type member is set to the VEX event base plus the event type constant name that uniquely identifies it. For example, when the X server reports a **Control** event to a client application, the client receives a **XVEXControlEvent** structure with the type member set to **VEX event base** + **Control**. The event base is obtained by calling **XQueryExtension**. The display, send_event, and serial members are set as defined in Xlib.

The event constant names are: **VEXControl**, **VEXViolation**, **VEXSync**, **VEXOverride**, **VEXChange**, **VEXConnectivity**, **VEXRequest.**

There is one new Error code defined by VEX, **BadVideo**. The error code of the XErrorEvent will have a value of **VEX error base** + **BadVideo**. The error base is obtained by calling **XQueryExtension**.

```
typedef struct {
        int type;                       /* of event */
        unsigned long serial;           /* # of last request processed by server */
        Bool send_event;                /* true if this came from a SendEvent request */
        Display *display;               /* Display the event was read from */
        Window window;                  /* unused */
        VideoID vid;                    /* from Setting given to ChangeVideoControls */
        Atom name;                      /* from Setting given to ChangeVideoControls */
        Time time;                      /* current server time */
        int state;                      /* VEXControlSuccess, VEXControlFail,
                                         * or VEXControlDenied */
} XVEXControlEvent;

typedef struct {
        int type;                       /* of event */
        unsigned long serial;           /* # of last request processed by server */
        Bool send_event;                /* true if this came from a SendEvent request */
        Display *display;               /* Display the event was read from */
        Window window;                  /* window on which video constraints were violated */
        VideoIO vid;                    /* VideoIO whose video constraints were violated */
        Time time;                      /* current server time */
        unsigned long action_mask;      /* VEXScale, VEXPlacement, VEXClip, VEXOverlap */
        int state;                      /* VEXViolationSuccess,
                                         * VEXViolationFail, or
                                         * VEXViolationSubset */
} XVEXViolationEvent;
```

```
typedef struct {
        int type;                       /* of event */
        unsigned long serial;           /* # of last request processed by server */
        Bool send_event;                /* true if this came from a SendEvent request */
        Display *display;               /* Display the event was read from */
        Drawable drawable;              /* drawable on which Render/Capture was requested */
        VideoIO vid;                    /* vid on which Render/Capture was requested */
        Time time;                      /* current server time */
        int state;                      /* VEXSyncAcquired or VEXSyncLost */
} XVEXSyncEvent;

typedef struct {
        int type;                       /* of event */
        unsigned long serial;           /* # of last request processed by server */
        Bool send_event;                /* true if this came from a SendEvent request */
        Display *display;               /* Display the event was read from */
        Drawable drawable;              /* drawable on which Render/Capture was requested */
        VideoIO vid;                    /* vid on which Render/Capture was requested */
        Time time;                      /* current server time */
} XVEXOverrideEvent;

typedef struct {
        int type;                       /* of event */
        unsigned long serial;           /* # of last request processed by server */
        Bool send_event;                /* true if this came from a SendEvent request */
        Display *display;               /* Display the event was read from */
        Window window;                  /* unused */
        int screen;
        VideoDevice vdev;               /* vdev which changed state */
        Time time;                      /* current server time */
        int state;                      /* VEXOnline or VEXOffline */
} XVEXChangeEvent;

typedef struct {
        int type;                       /* of event */
        unsigned long serial;           /* # of last request processed by server */
        Bool send_event;                /* true if this came from a SendEvent request */
        Display *display;               /* Display the event was read from */
        Window window;                  /* unused */
        int screen;
        unsigned long new_state;        /* index into list of adjacency Matrices */
        Time time;                      /* of last successful ChangeVideoConnectivity */
} XVEXConnectivityEvent;
```

```
typedef struct {
        int type;                    /* of event */
        unsigned long serial;        /* # of last request processed by server */
        Bool send_event;             /* true if this came from a SendEvent request */
        Display *display;            /* Display the event was read from */
        Window window;               /* unused */
        XID src;
        XID dst;
        XID cmap;
        int src_x;
        int src_y;
        int dst_x;
        int dst_y;
        int src_width;
        int src_height;
        int dst_width;
        int dst_height;
        Bool redirect;
        Bool full_motion;
        int request;                 /* VEXRender or VEXCapture */
        int priority;
        int subwindow_mode;
        Time                         time;/* current server time */
} XVEXRequestEvent;
```

Table of Contents