2
XView Attributes

This section lists all XView attributes in alphabetical order. Each attribute's description is in the format shown by the entry on this page. Only the fields appropriate to the particular attribute are shown.

INTRODUCTION
This field provides a brief description of the attribute.

Return Type: The return type describes the type that the attribute *should* return on a call to `xv_get()`. Note that this return type is normally coerced by the programmer. This field is only shown when the attribute is valid with `xv_get()`.

Argument: This shows the first programmer supplied value associated with the attribute. If an attribute has multiple values, then the type of each value is shown in multiple Argument fields.

Valid Values: Shows a list {in brackets} of values that are valid for the attribute.

Default: The default field shows the default value for the attribute.

Procs: Shows the attribute's valid procedures. In this field, `xv_create`, `xv_find`, `xv_get`, and `xv_set` exclude the `xv_` prefix.

Objects: The objects field shows the valid object or objects for the attribute. Attributes are divided into three classes: generic, common, and specific. (For more information, refer to the *XView Programming Manual*, Section 2.2.3, *Types of Attributes*.)

Callback:

```
returntype
function_name_proc(arg1, arg2, arg3)
        type1    arg1;
        type2   *arg2;
        type3   *arg3;
```

The callback field shows the application programmer defined callback procedure associated with the attribute. If there are multiple callbacks associated with the attribute, each callback has its own Callback field. The function's return value is provided, along with a description of its arguments, if any.

Usage: If an example is provided it will be shown in this field.

See Also: Where there are numbers, they refer to chapters and sections in the *XView Programming Manual*, where the attribute is described (letters refer to appendices). A list of related attributes and/or other related information may also be shown.

XView Attributes

ATTR_LIST

Specifies a NULL-terminated attribute-value list. It has no value type or default, but when used, it must be the first attribute in an attribute-value list. ATTR_STANDARD_SIZE, defined in *attr.h*, defines the maximum size of the attribute-value list.

Argument: Attr_avlist
Procs: create,set
Objects: All
Usage:

```
xv_create(object, pkg,
          ATTR_LIST, avlist,
          other_attrs, ..., 0);
```

See Also: 25.2.2.1, attr_create_list() in Section 3, *Procedures and Macros*

CANVAS_AUTO_CLEAR

Same as the OPENWIN_AUTO_CLEAR attribute.

CANVAS_AUTO_EXPAND

If TRUE, canvas width and height are never allowed to be less than the edges of the canvas window.

Argument: Bool
Default: TRUE
Procs: create, get, set
Objects: Canvas
See Also: 5.4.1

CANVAS_AUTO_SHRINK

If TRUE, canvas width and height are never allowed to be greater than the edges of the canvas window.

Argument: Bool
Default: TRUE
Procs: create, get, set
Objects: Canvas
See Also: 5.4.1

CANVAS_CMS_REPAINT

Specifies whether the canvas repaint procedure is called whenever a new colormap segment is set on the canvas, and/or the foreground and background colors of the canvas are changed.

Argument: Bool
Default: FALSE
Procs: create, get, set
Objects: Canvas
See Also: 5.3

CANVAS_FIXED_IMAGE

Sets the BitGravity for the canvas paint windows to be NorthWestGravity when TRUE or ForgetGravity when FALSE.

Argument: Bool
Default: TRUE
Procs: create, get, set
Objects: Canvas
See Also: 5.2 , WIN_BIT_GRAVITY, CANVAS_RESIZE_PROC

CANVAS_HEIGHT

Specifies the height of the canvas paint window in pixels.

Argument: `int`
Default: `0`
Procs: `create`, `get`, `set`
Objects: `Canvas`
See Also: 5.4.2, `CANVAS_MIN_PAINT_HEIGHT, CANVAS_WIDTH, XV_RECT`

CANVAS_MIN_PAINT_HEIGHT

Specifies the minimum height of the canvas' paint window(s) in pixels. Any attempt to change the `CANVAS_HEIGHT` to be smaller than this value will have the effect of setting it to the value of `CANVAS_MIN_PAINT_HEIGHT`.

Argument: `int`
Default: `0`
Procs: `create`, `get`, `set`
Objects: `Canvas`
See Also: 5.4.1, `CANVAS_AUTO_EXPAND, CANVAS_AUTO_SHRINK, CANVAS_HEIGHT, CANVAS_WIDTH`

CANVAS_MIN_PAINT_WIDTH

Specifies the minimum width of the canvas' paint window(s) in pixels. Any attempt to change the `CANVAS_WIDTH` to be smaller than this value will have the effect of setting it to the value of `CANVAS_MIN_PAINT_WIDTH`.

Argument: `int`
Default: `0`
Procs: `create`, `get`, `set`
Objects: `Canvas`
See Also: 5.4.1, `CANVAS_AU TO_EXPAND, CANVAS_AUTO_SHRINK, CANVAS_HEIGHT, CANVAS_WIDTH`

CANVAS_NO_CLIPPING

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

CANVAS_NTH_PAINT_WINDOW

Returns the paint window associated with the *n*th view, and takes an argument of the value of *n*. `0` is the index of the first view. `NULL` is returned if the view does not exist.

Return Type: `Xv_Window`
Argument: `int`
Procs: `get`
Objects: `Canvas`
Usage:

```
Canvas  canvas;
Xv_Window paint_window;
paint_window = (Xv_Window)xv_get(canvas,
                    CANVAS_NTH_PAINT_WINDOW, 1);
```

The example above will return the second paint window of the canvas.
See Also: 5.6.3.2

XView Attributes

CANVAS_PAINT_CANVAS_WINDOW

Gets the canvas from a canvas paint window.

Return Type: `Canvas`
Procs: `get`
Objects: `Canvas_paint_window`
See Also: `CANVAS_VIEW_CANVAS_WINDOW`

CANVAS_PAINT_VIEW_WINDOW

Gets the canvas view from a given canvas paint window.

Return Type: `Canvas`
Procs: `get`
Objects: `Canvas_paint_window`
See Also: `CANVAS_VIEW_PAINT_WINDOW`

CANVAS_PAINTWINDOW_ATTRS

Distributes specified attribute values across all paint windows in a given canvas. It takes an in-line attribute-value list of window attributes.

Argument: A-V list
Procs: `create`, `set`
Objects: `Canvas`
Usage:

```
xv_set(canvas, CANVAS_PAINTWINDOW_ATTR,
               WIN_EVENT_PROC, canvas_event,
               WIN_FOREGROUND_COLOR, 0
               NULL,
        NULL);
```

See Also: 20.3

CANVAS_REPAINT_PROC

Names a procedure called when canvas paint window has been damaged and must be repaired (repainted).

Argument `void (*`*canvas_repaint_proc*`)()`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Canvas`
Callback: (Used when `CANVAS_X_PAINT_WINDOW` set to `FALSE`)

```
void
canvas_repaint_proc(canvas, paint_window, repaint_area)
        Canvas     canvas;
        Xv_Window  paint_window;
        Rectlist  *repaint_area;
```

`canvas` is the canvas that was damaged.
`paint_window` is the window to repaint.
`repaint_area` is a pointer to the list of `Rect`s that is to be repainted.

Callback 2: (Used when CANVAS_X_PAINT_WINDOW set to TRUE)

When CANVAS_X_PAINT_WINDOW is TRUE, the CANVAS_REPAINT_PROC callback is called with the following parameters:

```
void
canvas_repaint_proc(canvas, paint_window, display, xid,
                    xrectlist)
        Canvas          canvas;
        Xv_Window       paint_window;
        Display        *display;
        Window          xid;
        Xv_xrectlist   *xrects;
```

canvas is the canvas that was damaged.
paint_window is the window to repaint.
display is the display handle to the X11 server connection.
xid is the X11 window identifier for the canvas paint window.
The Xv_xrectlist structure, defined in *<xview/xv_xrect.h>*, contains an array of XRectangles and a count that specifies the repaint area for the canvas paint window.

See Also: 5.3, CANVAS_RESIZE_PROC, CANVAS_X_PAINT_WINDOW, WIN_EVENT_PROC

CANVAS_RESIZE_PROC

Names a procedure called when the canvas paint window's width or height changes. Note that if CANVAS_FIXED_IMAGE is set to TRUE, which it is by default, the resize procedure will not be called when the canvas is resized smaller.

Argument: void (**canvas_resize_proc*)()
Default: NULL
Procs: create, get, set
Objects: Canvas
Callback:

```
void
canvas_resize_proc(canvas, width, height)
        Canvas    canvas;
        int       width;
        int       height;
```

canvas is the canvas being resized.
width and height are the new dimensions of the canvas.

See Also: 5.4.3, CANVAS_FIXED_IMAGE, CANVAS_REPAINT_PROC, WIN_BIT_GRAVITY

CANVAS_RETAINED

Specifies whether the X server should attempt to retain backing store for the canvas paint windows. Note that this does not guarantee that the window is retained as the server may not be able to comply.

Argument: Bool
Default: TRUE
Procs: create, get, set
Objects: Canvas
See Also: 5.2

CANVAS_VIEW_CANVAS_WINDOW

Gets the canvas from a canvas view window.

Argument: `Canvas`
Procs: `get`
Objects: `Canvas_view`
See Also: `CANVAS_PAINT_CANVAS_WINDOW, CANVAS_VIEW_PAINT_WINDOW`

CANVAS_VIEW_PAINT_WINDOW

This attribute is used to get the canvas paint window associated with a given canvas view.

Return Type: `Canvas_paint_window`
Procs: `get`
Objects: `Canvas_view`
Usage:

```
Canvas_view view;
Xv_Window paint_window;

paint_window = (Xv_Window)xv_get(view,
                              CANVAS_VIEW_PAINT_WINDOW);
```

See Also: 5.6.3.1, `CANVAS_PAINT_CANVAS_WINDOW, CANVAS_PAINT_VIEW_WINDOW`

CANVAS_VIEWABLE_RECT

Gets the visible part of the specified paint window in the paint window's coordinates. This attribute operates on a Canvas object and requires as an argument, a handle to one of the canvas' paint windows. The `Rect*` returned should not be freed, as it points to static storage.

Return Type: `Rect *`
Argument: `Canvas_paint_window`
Procs: `get`
Objects: `Canvas`
Usage:

```
Rect *rect;
Xv_Window canvas_pw = canvas_paint_window(canvas);
rect = (Rect *) xv_get(canvas,
                    CANVAS_VIEWABLE_RECT,canvas_pw);
```

See Also: 5.3

CANVAS_WIDTH

Specifies the width of the canvas paint window in pixels.

Argument: `int`
Default: In general, the canvas inherits its parent's width.
Procs: `create`, `get`, `set`
Objects: `Canvas`
See Also: 5.4.2, `CANVAS_MIN_PAINT_WIDTH`

CANVAS_X_PAINT_WINDOW

This attribute controls the parameters used in the `CANVAS_REPAINT_PROC`. Setting this attribute to `TRUE` will cause the `CANVAS_REPAINT_PROC` to be called back with the parameters `canvas, paint_window, display, xid, and xrectlist`; setting `CANVAS_X_PAINT_WINDOW` to `FALSE` will cause it to be called back with `canvas, paint_window, and repaint_area`.

Argument: Bool
Default: FALSE
Procs: create, get, set
Objects: Canvas
See Also: 5.3, CANVAS_REPAINT_PROC

CMS_BACKGROUND_PIXEL

Returns the background pixel (index 0) in a colormap segment.

Return Type: unsigned long
Default: None
Procs: get
Objects: Cms
See Also: 21.3.1

CMS_COLOR_COUNT

Used to specify the number of colors being set with CMS_COLORS or CMS_X_COLORS. This can be used in conjunction with CMS_INDEX to set a range of the colors in the cms.

Argument: unsigned int
Default: CMS_SIZE for a non-control cms, or CMS_SIZE - CMS_CONTROL_COLORS for a control cms.
Procs: create,set
Objects: Cms
Usage: To set the colors in the array colors to the range from 10 to 14 in a cms, you would do the following:

```
XColor colors[5];

xv_set(cms,
        CMS_X_COLORS, colors,
        CMS_INDEX, 10,
        CMS_COLOR_COUNT, 5,
        NULL);
```

See Also: 21.2.0.1

CMS_COLORS

Specifies the colors to be loaded into the colormap segment. Colors are specified as an array of Xv_singlecolor.

Argument: Xv_singlecolor
Default: None
Procs: create,get,set
Objects: Cms
See Also: 21.2.1.2

CMS_CONTROL_CMS

Indicates whether this colormap segment is a control colormap segment for use by control objects (like panel items and scrollbars).

Argument: int
Default: FALSE
Procs: create,get
Objects: Cms
See Also: 21.5

CMS_FOREGROUND_PIXEL

Returns the foreground pixel (index `CMS_SIZE`-1) in a colormap segment.

Return Type: `unsigned long`
Default: None
Procs: `get`
Objects: `Cms`
See Also: 21.3.1

CMS_INDEX

Specifies the starting index to `CMS_COLOR_COUNT` entries into the colormap.

Argument: `unsigned long`
Default: `0` (If control cms, `CMS_CONTROL_COLORS`)
Procs: `create`, `set`
Objects: `Cms`
See Also: 21.2.0.1, `CMS_COLOR_COUNT`

CMS_INDEX_TABLE

Used to translate the logical indices of the window's colormap segment into actual pixel values.

Return Type: `unsigned long *`
Default: N/A
Procs: `get`
Objects: `Cms`
See Also: 21.3.0.1

CMS_NAME

Specifies the name of the colormap segment.

Argument: `char *`
Default: Unique name generated internally
Procs: `create`, `get`, `set`
Objects: `Cms`
See Also: 21.6

CMS_NAMED_COLORS

Specifies the names of the colors to be loaded into the colormap segment. The `NULL`-terminated list of color names is parsed by `XParseColor()`.

Argument: List of `char *`
Default: None
Procs: `create`, `set`
Objects: `Cms`
See Also: 21.2.1.1

CMS_PIXEL

Translates a logical index into the actual colormap pixel value.
Argument: `unsigned long`
Default: None
Procs: `get`
Objects: `Cms`
Usage:

```
unsigned long pixel;
pixel=(unsigned long)xv_get(cms, CMS_PIXEL, 2);
```

See Also: 21.3.0.1

CMS_SCREEN

Returns the screen with which the colormap segment is associated.

Return Type: `Xv_Screen`
Default: Default screen
Procs: `get`
Objects: `Cms`

CMS_SIZE

Specifies the size of the colormap segment.

Argument: `int`
Default: `XV_DEFAULT_CMS_SIZE` or `CMS_CONTROL_COLORS`, for a control cms
Procs: `create`, `get`
Objects: `Cms`
See Also: 21.2.0.1

CMS_TYPE

Specifies the type of the colormap segment. If set to `XV_STATIC_CMS` , then only read-only colors can be allocated from the cms. If set to `XV_DYNAMIC_CMS` , then only read-write colors can be allocated from the cms. `XV_DYNAMIC_CMS` can only be used with a dynamic visual.

Argument: `Cms_type`
Default: `XV_STATIC_CMS`
Procs: `create`
Objects: `Cms`
See Also: 21.1.1

CMS_X_COLORS

Specifies the colors to be loaded into the colormap segment. Colors are specified as an array of `XColor`.

Argument: `XColor *`
Default: None
Procs: `create`, `set`, `get`
Objects: `Cms`
See Also: 21.2.1.2

CURSOR_BACKGROUND_COLOR

Specifies the background color of a cursor as an RGB triplet.

Argument: `Xv_singlecolor *`
Default: `white (255, 255, 255)`
Procs: `create`, `get`, `set`
Objects: `Xv_Cursor`
See Also: 13.4

CURSOR_DRAG_STATE

Indicates whether the cursor is over a neutral zone (`CURSOR_NEUTRAL`), a valid drop zone (`CURSOR_ACCEPT`), or an invalid drop zone (`CURSOR_REJECT`). The shape of the cursor varies depending on the state. Note that the current drag and drop protocol does not support a "reject" cursor.

Argument: `Cursor_drag_state`
Default: `CURSOR_NEUTRAL`
Proc: `create,get`
Objects: `Cursor`
See Also: 13.5, `CURSOR_DRAG_TYPE, CURSOR_STRING`

XView Attributes

CURSOR_DRAG_TYPE

Indicates whether the cursor is "move" (CURSOR_MOVE) or "copy" (CURSOR_DUPLICATE). The duplicate version has a shadow. When combined with CURSOR_STRING, you get either a text move or text duplicate cursor.

Argument: Cursor_drag_type
Default: CURSOR_MOVE
Procs: create, get
Objects: Cursor
See Also: 13.5, CURSOR_DRAG_STATE, CURSOR_STRING

CURSOR_FOREGROUND_COLOR

Specifies the foreground color of a cursor as an RGB triplet.

Argument: Xv_singlecolor *
Default: black (0, 0, 0)
Procs: create, get, set
Objects: Xv_Cursor
See Also: 13.4

CURSOR_IMAGE

Specifies the cursor's image.

Argument: Server_image
Default: None
Procs: create, get, set
Objects: Xv_Cursor
See Also: 13.1

CURSOR_MASK_CHAR

Specifies the index into the mask shape font. Predefined shapes are defined in *<xview/cursor.h>* .

Argument: unsigned int
Default: None
Procs: create, get, set
Objects: Xv_Cursor
Usage:

```
xv_set(cursor,
       CURSOR_MASK_CHAR, OLC_BASIC_MASK_PTR
       NULL);
```

See Also: 13.2, CURSOR_SRC_CHAR

CURSOR_OP

The value for this attribute is the rasterop (defined in *<pixrect.h>*) which will be used to paint the cursor.

Argument: int
Default: { PIX_SRC | PIX_DST }
Procs: create, get, set
Objects: Xv_Cursor

CURSOR_SRC_CHAR

Specifies the index into the shape font. Predefined shapes are in <*xview/cursor.h*>.

Argument: `unsigned int`
Default: -1
Procs: `create`, `get`, `set`
Objects: `Xv_Cursor`
Usage:

```
xv_set(cursor,
        CURSOR_SRC_CHAR, OLC_BASIC_PTR,
        CURSOR_MASK_CHAR, OLC_BASIC_MASK_PTR,
        NULL);
```

See Also: 13.2, `CURSOR_MASK_CHAR`

CURSOR_STRING

Creates a text drag and drop cursor. The value of the attribute is the string which is to be displayed inside the "flying punch card." If the string exceeds 3 characters, then only the first 3 characters are displayed, and a "More arrow" is shown within the cursor. `CURSOR_STRING` is mutually exclusive of `CURSOR_IMAGE`, `CURSOR_SRC_CHAR`, and `CURSOR_MASK_CHAR`. The string is not copied. Once the drag and drop operation is complete, the text cursor objects used in the operation must be destroyed.

Argument: `char *`
Default: None
Procs: `create,get`
Objects: `Cursor`
See Also: 13.5, `CURSOR_DRAG_STATE, CURSOR_DRAG_TYPE`

CURSOR_XHOT

Specifies the x coordinate of the hotspot. Its value cannot be negative.

Argument: `int`
Default: `0`
Procs: `create`, `get`, `set`
Objects: `Cursor`
See Also: 13.3

CURSOR_YHOT

Specifies the y coordinate of the hotspot. Its value cannot be negative.

Argument: `int`
Default: `0`
Procs: `create`, `get`, `set`
Objects: `Cursor`
See Also: 13.3

DND_ACCEPT_CURSOR

During a drag and drop operation, this defines the mouse cursor that is used when the cursor is over an acceptable drop-site.

Argument: `Xv_cursor`
Default: Predefined OPEN LOOK drag and drop cursor
Procs: `create,get,set`
Objects: `Dnd`
See Also: 19.3.3

DND_ACCEPT_X_CURSOR

An alternative to DND_ACCEPT_CURSOR, this attribute accepts an XID of a cursor instead of an Xv_cursor.

Argument: XID
Default: Predefined OPEN LOOK drag and drop cursor
Procs: create,get,set
Objects: Dnd
See Also: 19.3.3

DND_CURSOR

Defines the mouse cursor that will be used during the drag portion of the drag and drop operation.

Argument: Xv_cursor
Default: Predefined OPEN LOOK drag and drop cursor
Procs: create,get,set
Objects: Dnd
See Also: 19.3.3, DND_ACCEPT_CURSOR

DND_TIMEOUT_VALUE

Defines the amount of time to wait for an acknowledgment from the drop destination after the kicker message has been sent (ACTION_DRAG_COPY or ACTION_DRAG_MOVE). The kicker message is sent when the user releases the mouse button, forcing the drop.

Argument: struct timeval *
Default: Default selection package timeout
Procs: create,get,set
Objects: Dnd
See Also: 19.3.4

DND_TYPE

Defines whether this drag and drop operation will be a copy or a move. This is just a hint to the destination. If the type is a move operation and if the destination honors the hint, the destination will ask the source to convert the DELETE target.

Valid Values: DND_COPY or DND_MOVE
Default: DND_MOVE
Procs: create,get,set
See Also: 19.3

DND_X_CURSOR

An alternative to DND_CURSOR, but accepts an XID of a cursor instead of an Xv_cursor.

Argument: XID
Default: Predefined OPEN LOOK drag and drop cursor
Procs: create,get,set
Objects: Dnd
See Also: 19.3.3, DND_ACCEPT_X_CURSOR

DROP_SITE_DEFAULT

Specifies that this drop-site wants forwarded drops from the window manager. Such drops include drops on icons and window manager decorations. Only one drop-site default should be specified per base frame (specifying more than one will have unpredictable results). This attribute is only a hint to the window manager.

Argument: `Bool`
Default: `False`
Procs: `create,get,set`
Objects: `Drop_site_item`
See Also: 19.2

DROP_SITE_DELETE_REGION

Used to remove a region from the drop item. If the owner is destroyed with `xv_destroy()`, any drop-site regions attached to it will also be destroyed. When a `NULL` is passed as an argument, all regions in the drop-site will be removed.

Argument: `Rect *`
Default: N/A
Procs: `create,set`
Objects: `Drop_site_item`
See Also: 19.2.1

DROP_SITE_DELETE_REGION_PTR

Used to remove a list of regions from the drop item. Passing a `NULL` as an argument removes all regions in the drop-site.

Argument: `Rect *` (a `NULL`-terminated array of `Rect` structs)
Default: N/A
Procs: `create,set`
Objects: `Drop_site_item`
See Also: 19.2.1

DROP_SITE_EVENT_MASK

A mask used to specify if the regions within the site will receive synthetic previewing events. The previewing events will come with `event_action()` set to `ACTION_DRAG_PREVIEW` and `event_id()` set to one of `LOC_WINENTER`, `LOC_WINEXIT` or `LOC_DRAG`. These events will be delivered to the event procedure of the owner of the drop-site item. This mask is only a hint to the source. There is no guarantee the source will send these previewing events.

Argument: `int`
Valid Values: {`DND_ENTERLEAVE, DND_MOTION`} Defined in <*xview/dragdrop.h*>
Default: `NULL`
Procs: `create,get,set`
Objects: `Drop_site_item`
See Also: 19.2.2.1

DROP_SITE_ID

An uninterpreted ID used to distinguish one drop-site from the next. Ideal when more than one site has been set on an object.

Argument: `long`
Default: A value generated from `xv_unique_key()`
Procs: `create,get,set`
Objects: `Drop_site_item`
See Also: 19.2.2.1

XView Attributes

DROP_SITE_REGION

This attribute is used to associate a region to a drop-site item. The region is a Rect *. DROP_SITE_REGION will add to any existing regions within the drop-site item. The rect pointed to by Rect * will be copied. The coordinates in the rect should be relative to the drop-site item's owner's window. An xv_get() of a region of type Rect * will return an allocated Rect structure. This should be freed using xv_free() once the application has finished using it.

Argument: Rect *
Default: NULL
Procs: create,get,set
Objects: Drop_site_item
See Also: 19.2.1, DROP_SITE_DELETE_REGION

DROP_SITE_REGION_PTR

This attribute is similar to DROP_SITE_REGION except that it accepts a NULL-terminated array of regions. It will add to any existing regions that exist within the drop item. A NULL rect is defined to be one with width or height equal to 0. An xv_get() of a region list will return a NULL-terminated list of Rects. This data should be freed once the application has finished using it.

Argument: Rect * (A NULL-terminated array of Rect structs)
Default: NULL
Procs: create,get,set
Objects: Drop_site_item
See Also: 19.2.1, DROP_SITE_DELETE_REGION

FILE_CHOOSER_ABBREV_VIEW

Specifies if the FILE_CHOOSER should show invalid file names grayed out or not display them at all. An invalid file name is specified by the FILE_CHOOSER_FILTER_FUNC returning a value of FILE_CHOOSER_IGNORE or with an op set to FILE_CHOOSER_NOT_MATCHED for files, but not for directories. This only works for FILE_CHOOSER_FILTER_STRING. Note that the Save and Save As dialogs require that all documents are grayed out; setting this attribute on them will result in a display with nothing but folders and the "Go up one" entry.

Argument: Bool
Default: FALSE
Procs: create, get, set
Objects: File_chooser
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_CHOOSER_APP_DIR

Adds an application-specific path name to the Go To History Menu's fixed space. The first argument is the string that will show on the Menu_item, and the second is the path name represented by this label. Both strings may be the same. The Application File Choosing Specification imposes a limit of five application-specific entries in the Go To menu. Any more than five unique entries will be *silently truncated* by the FILE_CHOOSER package.

Note that the Goto List is, by default, shared amongst the various FILE_CHOOSER instances, so setting this on one FILE_CHOOSER will affect the rest of the instances. Also, the Default History List used by FILE_CHOOSER ignores duplicate entries in the Recent Space.

Argument 1: char *
Argument 2: char *
Procs: create, set
Objects: File_chooser
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_CHOOSER_AUTO_UPDATE

Tells the FILE_CHOOSER not to re-read its current directory until either FILE_CHOOSER_UPDATE is called explicitly, or until the user performs some action to read the directory.

Argument: Bool
Default: TRUE
Procs: create, get, set
Objects: File_chooser
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_CHOOSER_CD_FUNC

Specifies a callback to be invoked with an op of FILE_CHOOSER_BEFORE_CD before the FILE_CHOOSER attempts to change to another directory.

Argument: int (**cd_func*)()
Procs: create, get, set
Objects: File_chooser
Callback:

```
int
cd_func( chooser, path, stats, op )
   File_chooser chooser;
   char *  path;
   struct stat * stats;
   File_chooser_op op;
```

The expected return values are XV_OK or XV_ERROR. If XV_ERROR is returned, the directory change is effectively vetoed. It is expected that the application provides the user with the appropriate feedback.

The callback is invoked again with an op of FILE_CHOOSER_AFTER_CD after the contents of the scrolling list have been loaded, but before it has been displayed to the user. The return value is ignored from this invocation.

FILE_CHOOSER_CHILD

Get a handle to the opaque handles to the UI objects.

Warning: relying on values that use xv_set or xv_get from these handles is not guaranteed to be compatible in future releases.

Argument: File_chooser_child
Valid Values:

```
FILE_CHOOSER_GOTO_MESSAGE_CHILD
FILE_CHOOSER_GOTO_BUTTON_CHILD
FILE_CHOOSER_GOTO_PATH_CHILD
FILE_CHOOSER_HISTORY_MENU_CHILD
FILE_CHOOSER_CURRENT_FOLDER_CHILD
FILE_CHOOSER_SELECT_MESSAGE_CHILD
FILE_CHOOSER_FILE_LIST_CHILD
FILE_CHOOSER_DOCUMENT_NAME_CHILD
FILE_CHOOSER_OPEN_BUTTON_CHILD
FILE_CHOOSER_SAVE_BUTTON_CHILD
FILE_CHOOSER_CANCEL_BUTTON_CHILD
FILE_CHOOSER_CUSTOM_BUTTON_CHILD
```

Procs: get
Objects: File_chooser
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

XView Attributes

FILE_CHOOSER_COMPARE_FUNC

Function to compare two entires in the list of files. Setting the `FILE_CHOOSER_FILTER_FUNC` to `NULL` does not guarantee any ordering to row entries.

Argument: `int (*`*compare_func*`)()`
Default: Case-insensitive ascending sorting order. Also provided are case-insensitive descending and case sensitive ascending and descending.
Procs: `create`, `get`, `set`
Objects: `File_chooser`
Callback:

```
int
compare_func( row1, row2 )
   File_chooser_row *row1;
   File_chooser_row *row2;
```

where `File_chooser_row` is defined as:

```
typedef struct {
    char *  file;
    struct stat * stats;
    File_chooser_op matched;
    char *  xfrm;
} File_chooser_row;
```

`file` is a string representing the name of the file being sorted.
`stats` is a pointer to a stat structure as returned by the `stat()` system call.
`matched` is one of the values `FILE_CHOOSER_MATCHED` or `FILE_CHOOSER_NOT_MATCHED`, indicating if the file name has matched the regular expression installed using `FILE_CHOOSER_FILTER_STRING`.
`xfrm` field represents the file name, as returned by the `strxfmt(3)` function.

FILE_CHOOSER_CUSTOMIZE_OPEN

Allows the client to re-use the Open dialog within other contexts. Active only on a dialog of type `FILE_CHOOSER_OPEN`. The recommended uses are Insert, Include or Import.

The first argument is a string that will be used as the label for a special-purpose button at the bottom of the dialog. The Open button becomes Open Folder, as with a Save or Save As dialog. The second argument becomes the help message that is displayed immediately above the Scrolling List. The third is of type `File_chooser_op` indicating if this new type should be able to select files, or both files and directories (values are `FILE_CHOOSER_SELECT_FILES` or `FILE_CHOOSER_SELECT_ALL`).

The button added by this call will activate the `FILE_CHOOSER_NOTIFY_FUNC`, with the same arguments as the open callback and is retrievable using `FILE_CHOOSER_CHILD` with a value of `FILE_CHOOSER_CUSTOM_BUTTON`.

Argument 1: `char *`
Argument 2: `char *`
Argument 3: `File_chooser_op`
Procs: `create`
Objects: `File_chooser`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_CHOOSER_DIRECTORY

Specifies the current working directory being displayed in the FILE_CHOOSER. This can be NULL.

Argument: char *
Default: The current working directory.
Procs: create, get, set
Objects: File_chooser
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_CHOOSER_DOC_NAME

Specifies the name of the application's current document for use by the Save As dialog (this is only effective if the FILE_CHOOSER_TYPE attribute is FILE_CHOOSER_SAVEAS). It is suggested that this value be set to *current.doc.name*.1 whenever a new document is created or opened.

Specifies the name of the default document name in a Save dialog (that is, this is only effective if the FILE_CHOOSER_TYPE attribute is FILE_CHOOSER_SAVE).

Argument: char *
Default: Untitled1
Procs: create, get, set
Objects: File_chooser
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_CHOOSER_EXTEN_FUNC

Function that gets called during resize handling that allows the client to adjust its extension controls for the new size of the dialog. The return value is the new height desired for the exten_rect, or -1 if there is no change. Note that new *y* values should be relative to the exten_rect and should expect to increase it upwards, because adding height to the extension area takes away from the number of rows in the Scrolling List. Newly defined height values should never exceed the max_height parameter. Clients may adjust the FRAME_MIN_SIZE attribute to prevent this from being a problem. Returning a new height from this function is *not* the same as xv_set for the FILE_CHOOSER_EXTEN_HEIGHT attribute.

Argument: int (**exten_func*)()
Procs: create, get, set
Objects: File_chooser
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.
Callback:

```
int
exten_func(fc, frame_rect, exten_rect, left_edge,
           right_edge, max_height)
   File_chooser fc;
   Rect *   frame_rect;
   Rect *   exten_rect;
   int      left_edge;
   int      right_edge;
   int      max_height;
```

fc is the File Chooser instance.
frame_rect is the new size of the Command Frame.
exten_rect is the current size of the area delegated for extension items.
left_edge is the position being used as the leftmost x position used by the regular objects on the File Chooser.

XView Attributes

`right_edge` is the position being used as the rightmost point for regular item layout.
`max_height` is the maximum amount of space that the extension rect can occupy given the current size of the Frame.

It is recommended that the client set the `PANEL_PAINT` attribute to the value of `PANEL_NONE` for each `xv_set` call made on a Panel Item within this callback.

FILE_CHOOSER_EXTEN_HEIGHT

Tells the File Chooser to reserve this much vertical space for the added controls. Positioning controls outside this area is not supported by the XView File Chooser.

Argument: `int`
Procs: `get`, `set`
Objects: `File_chooser`

FILE_CHOOSER_FILTER_FUNC

Function called to validate the file names before they will be displayed to the user. The files for which this callback is invoked are chosen using `FILE_CHOOSER_FILTER_MASK`.

Argument: `int (*`*filter_func*`)()`
Procs: `create`, `get`, `set`
Objects: `File_chooser`
Callback:

```
File_chooser_op
filter_func( fc, path, stats, matched, glyph,
                 client_data, mask_glyph )
   File_chooser fc;
   char *  path;
   struct stat * stats;
   File_chooser_op matched;
   Server_image * glyph;
   Xv_opaque * client_data;
   Server_image * mask_glyph;
```

`fc` is the File Chooser instance for which the callback was invoked.
`path` is the path name of the file being decided on.
`statbuf` is the address of a `stat` structure (see the `stat(2)` man page for details).
`matched` is one of the values `FILE_CHOOSER_MATCHED` or `FILE_CHOOSER_NOT_MATCHED`, depending on if the file name was found to match the regular expression given using the `FILE_CHOOSER_FILTER_STRING` attribute.
`glyph` and `client_data` are returnable values. If the filter function returns a `Server_image` in the glyph field, the glyph returned will be displayed in the list along with the file name. The `Server_image` returned is expected to be 16×16 pixels, regardless of the scale or font size of the File Chooser.
`mask_glyph` is the clip mask to be used with the glyph field. The `Server_image` returned must be of depth 1. In most cases, if the `glyph` is of depth 1, the `mask_glyph` is the same `Server_image` as the `glyph`.

If a value is returned in the `client_data` field, this same value will be returned, uninterpreted, to the callback installed from the `FILE_CHOOSER_NOTIFY_FUNC` attribute.

The expected return value from the filter-func is one of the ops `FILE_CHOOSER_ACCEPT` or `FILE_CHOOSER_IGNORE`. If the return value is `FILE_CHOOSER_IGNORE`, this file will be displayed grayed out, and unselectable. This is the default behavior specified by the Application File Choosing Specification. If the FILE_CHOOSER_ABBREV_VIEW attribute is set to TRUE, these files will not be displayed at all.

Also, in a Save or Save As dialog, all documents will be set inactive automatically.

FILE_CHOOSER_FILTER_MASK

Mask bits for the FILE_CHOOSER_FILTER_FUNC.

Argument: int (File_chooser_filter_mask)
Valid Values: Defined by a set or an or of flags from the set of:

```
FC_NONE_MASK
FC_MATCHED_FILES_MASK
FC_NOT_MATCHED_FILES_MASK
FC_MATCHED_DIRS_MASK
FC_NOT_MATCHED_DIRS_MASK
FC_DOTDOT_MASK
FC_ALL_MASK
```

Default: FC_MATCHED_FILES
Procs: create, get, set
Objects: File_chooser
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_CHOOSER_FILTER_STRING

Sets or gets an ex-like regular expression string which files are filtered through before being actively displayed in the list. If no filter string is specified, all entries are assumed to match. ".." is always assumed to match. A default glyph may be specified for entries that match the regular expression using FILE_CHOOSER_MATCH_GLYPH.

Note that the Save and Save As dialogs must have all of the documents grayed out, so setting a filter string on them will not be useful unless you are using a filter function to assign glyphs as well, even though they are grayed out, or issuing FILE_CHOOSER_ABBREV_VIEW.

Argument: char *
Procs: create, get, set
Objects: File_chooser
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_CHOOSER_HISTORY_LIST

Specifies an opaque handle to the HISTORY_LIST object associated with the dialog instance. By default, the FILE_CHOOSER package will look for a list identified by the string "XView GoTo History" and create it if it cannot be found.

Argument: History_list
Procs: create, get, set
Objects: File_chooser
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_CHOOSER_MATCH_GLYPH

Sets or gets a Server_image to be used as the default glyph for all file names matching the filter string (see FILE_CHOOSER_FILTER_STRING attribute).

Argument: Server_image
Procs: create, get, set
Objects: File_chooser
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_CHOOSER_MATCH_GLYPH_MASK

Sets or gets Server_image to be used as a clip mask for the corresponding FILE_CHOOSER_MATCH_GLYPH. The Server_image must be of depth 1.

Argument: Server_image
Default: XV_NULL
Procs: create, get, set
Objects: File_chooser
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_CHOOSER_NO_CONFIRM

Tells the FILE_CHOOSER not to do confirmation before issuing the call to open or save callbacks.

Argument: Bool
Default: FALSE
Procs: create, get, set
Objects: File_chooser
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_CHOOSER_NOTIFY_FUNC

Callbacks invoked when the user selects a file from the dialog for opening or saving. The callback takes different form depending on whether the FILE_CHOOSER_TYPE is FILE_CHOOSER_OPEN or FILE_CHOOSER_SAVE/AS. Expected return values are XV_OK or XV_ERROR.

Argument: int (**notify_callback*)()
Procs: create, get, set
Objects: File_chooser
Callback 1:

```
int
open_callback( fc, path, file, client_data )
   File_chooser fc;
   char *  path;
   char *  file;
   Xv_opaque client_data;
```

client_data is the client_data field set for this row in the list from the FILE_CHOOSER_FILTER_FUNC.
path is the entire path to the file.
The file string is strictly the file-name portion of the path. The directory portion of the path may be obtained with the FILE_CHOOSER_DIRECTORY attribute.

Callback 2:

```
int
save_callback( fc, path, stats )
    File_chooser  fc;
    char *        path;
    struct stat * stats;
```

`path` is the full path of the file to be saved.

`stats` is a pointer to the file's stat structure if the file exists, or `NULL` if the file does not exist.

If the file exists, it is up to the application to check the permissions from the stat structure and act accordingly. The File Chooser package will confirm with the user before overwriting an existing file prior to issuing the callback; this may be prevented with `FILE_CHOOSER_NO_CONFIRM`. Thus, if this callback is issued, the user expects the file to be written or overwritten. It is the responsibility of the client to alert the user of any failures with the appropriate feedback.

FILE_CHOOSER_SAVE_TO_DIR

If set `TRUE`, the Save/As: typein becomes blank and inactive, while the Save button remains active. In the default state (`FALSE`), the behavior is that the Save button is inactive whenever the Save/As: typein becomes empty.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `File_chooser`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_CHOOSER_SHOW_DOT_FILES

Specifies whether or not the `FILE_CHOOSER` package should consider files beginning with the "." prefix.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `File_chooser`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_CHOOSER_TYPE

Specifies the type of dialog to create.

Argument: One of: `FILE_CHOOSER_OPEN`, `FILE_CHOOSER_SAVE` `FILE_CHOOSER_SAVEAS`.
Procs: `create`, `get`
Objects: `File_chooser`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_CHOOSER_UPDATE

Tells the `FILE_CHOOSER` to re-read its current directory and update its display.

Argument: `None`
Procs: `set`
Objects: `File_chooser`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_LIST_ABBREV_VIEW

Tells the `FILE_LIST` package to "compress" out the files in the list for which the client returned `FILE_LIST_IGNORE` from the filter function (see `FILE_LIST_FILTER_FUNC` attribute). This behavior of graying out filenames, rather than leaving them out entirely, is specified in the Open Look File Choosing Spec as the default for a File Chooser object.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `File_list`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_LIST_AUTO_UPDATE

Specifies whether the `FILE_LIST` package should automatically update the current list when the client changes an attribute that would modify the display (e.x. `FILE_LIST_SHOW_DOT_FILES`).

Argument: `Bool`
Default: `TRUE`.
Procs: `create`, `get`, `set`
Objects: `File_list`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_LIST_CHANGE_DIR_FUNC

Sets or gets a function pointer that the client installs for the purpose of being notified for, and possibly to veto a change of directory. This can happen if the application sets the `FILE_LIST_DIRECTORY` attribute, or if the user double-clicks on a valid directory entry in the list (including the "go up one folder" entry.)

Argument: `int (*`*cd_func*`)()`
Procs: `create`, `get`, `set`
Objects: `File_list`
Callback:

```
int
cd_func ( item, path, statbuf, op )
  File_list   item;
  char *      path;
  struct stat *  statbuf;
  File_list_op  op;
```

The valid `File_list_op` values are `FILE_LIST_BEFORE_CD` and `FILE_LIST_AFTER_CD`.

The function is a notification of a pending change of directory to the new directory *path*. The callback is called both before and after the list is updated with the new directory. If the callback returns `XV_ERROR`, and the `op` is `FILE_LIST_BEFORE_CD`, than the pending change will be canceled, and the application will be assumed to provide the appropriate feedback to the user.

The return value is expected to be `XV_OK` or `XV_ERROR`.

The second time this function is called, the op value is equal to `FILE_LIST_AFTER_CD`. This informs the function that the call is after the list has been created, but before the list is displayed for the user. This gives you the opportunity to modify the list before displaying it.

FILE_LIST_COMPARE_FUNC

Used with the qsort library routine. Default function is case-insensitive ascending. Also provided are case-insensitive descending and case-sensitive ascending and descending. See the functions and macros section for a list of the provided comparison functions.

The comparison function is specified as follows:

Argument: int (*compar_func)()
Procs: create, get, set
Objects: File_list
Callback:

```
int
compare_func( row1, row2 )
  File_list_row *row1;
  File_list_row *row2;
```

The File_list_row structure includes the Panel_list_row_values struct for this row, as well as the stat buffer and the matched flag. This will allow you to sort by fields in any of these structures, if desired.

FILE_LIST_DIRECTORY

Sets or gets the directory currently being displayed in the list. If FILE_LIST_DIRECTORY is set to NULL, the FILE_LIST will essentially become empty.

Argument: char *
Default: the current working directory
Procs: create, get, set
Objects: File_list
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_LIST_DOTDOT_STRING

Allows the client to modify the string used by the FILE_LIST package do denote the ".." entry.

Argument: char *
Default: "... Go up one folder ..."
Procs: create, get, set
Objects: File_list
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_LIST_FILTER_STRING

Sets or gets an ex-like regular expression string which files are filtered through before being actively displayed in the list. If no filter string is specified, all files are assumed to match. ".." is always assumed to match.

If a file does not match, it is grayed out by default. This is not true of directories. Directories may only be inactivated using FILE_LIST_FILTER_FUNC.

Argument: char *
Default: A default glyph may be specified for files that match the regular expression using FILE_LIST_MATCH_GLYPH.
Procs: create, get, set
Objects: File_list
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

XView Attributes

FILE_LIST_FILTER_FUNC

Sets or gets a function pointer that the client installs for the purpose of okaying a file to be actively displayed in the list.

Argument: `int (*client_filter)()`
Procs: `create`, `get`, `set`
Objects: `File_list`
Callback:

```
File_list_op
client_filter ( path, row )
   char *          path;
   File_list_row *  row;
```

`path` is a string that contains the absolute path of the file in question (except "..").
`row` is a `File_list_row` structure. `File_list_row` is defined as:

```
typedef struct {
  File_list             file_list;
  Panel_list_row_values  vals;
  struct stat           statbuf;
  File_list_op          matched;
  char *                xfrm;
} File_list_row;
```

`file_list` is the object handle of the `File_list` instance.
`vals`, the `Panel_list_row_values`, is as specified in *<xview/panel.h>*. These are the available parameters to a specific row in the `PANEL_LIST`. It can be conditionally modified from this callback; the exception is that the `FILE_LIST` package may modify the inactive field according to the return value from this function and the `FILE_LIST_ABBREV_VIEW` attribute. In addition, the `extension_data` field is reserved for use by the `FILE_LIST` package.
`statbuf` is the address of the current `stat` structure for the file.
`matched` is an indicator of whether or not the file name matched the regular expression and is one of `FILE_LIST_MATCHED` or `FILE_LIST_NOT_MATCHED`.
`xfrm` field contains the file name in the collated representation as returned by the `strxfrm(3)` function.

The return values for the callback are defined by `File_list_op`: `FILE_LIST_IGNORE` and `FILE_LIST_ACCEPT`. `FILE_LIST_IGNORE` means that the `FILE_LIST` package is not supposed to actively display this particular file name. `FILE_LIST_ACCEPT` tells the `File_list` package to actively display the entry.

Note: modifying the contents of the list from this callback it is *not* supported. To modify the list, use the attribute `FILE_LIST_CHANGE_DIR_FUNC` and wait for the `FILE_LIST_AFTER_CD` operation.

FILE_LIST_FILTER_MASK

Mask for FILE_LIST_FILTER_FUNC attribute.

Argument: int (of type File_list_filter_mask)
Valid Values:

FL_NONE_MASK
FL_MATCHED_FILES_MASK
FL_NOT_MATCHED_FILES_MASK
FL_MATCHED_DIRS_MASK
FL_NOT_MATCHED_DIRS_MASK
FL_DOTDOT_MASK
FL_ALL_MASK

Default: FL_MATCHED_FILES
Procs: create, get, set
Objects: File_list
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_LIST_MATCH_GLYPH

Sets or gets a glyph that is to be the default for all files that match the regular expression given by the FILE_LIST_FILTER_STRING attribute. This does not apply to directory entries. Note: this is strictly a convenience, this glyph may still be overridden with FILE_LIST_FILTER_FUNC.

Argument: Server_image
Procs: create, get, set
Objects: File_list
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_LIST_MATCH_GLYPH_MASK

Sets or gets a clip mask for the FILE_LIST_MATCH_GLYPH. The Server_image given must be of depth 1.

Argument: Server_image
Default: XV_NULL
Procs: create, get, set
Objects: File_list
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_LIST_ROW_TYPE

Takes a row number as argument. Returns a value of type File_list_row_type.

Argument: int
Valid Values:

FILE_LIST_DOTDOT_TYPE
FILE_LIST_DIR_TYPE
FILE_LIST_FILE_TYPE

Procs: get
Objects: File_list
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_LIST_SHOW_DIR

Tells the `FILE_LIST` package to display its current directory path in the Title item of the list (uses the `PANEL_LIST_TITLE` attribute).

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `File_list`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_LIST_SHOW_DOT_FILES

Specifies whether or not the `FILE_LIST` package should consider files beginning with the "." prefix.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `File_list`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_LIST_UPDATE

Tells the `FILE_LIST` to re-read its current directory and update its display.

Argument: None
Procs: `set`
Objects: `File_list`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FILE_LIST_USE_FRAME

Tells the `FILE_LIST` package to display any error messages in the footer of the current Frame. If the current Frame has no footer, or the attribute is `FALSE`, the `FILE_LIST` package will use `xv_error()`.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `File_list`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

FONT_CHAR_HEIGHT

Returns the height (an `int`) of a specified character (a `char`) of the font. This is actually the height of the bounding rectangle, good for any character of the font.

Return Type: `int`
Argument: `char`
Procs: `get`
Objects: `Xv_Font`
Usage:

```
Xv_font   font;
int       height;
height = (int)xv_get(font, FONT_CHAR_HEIGHT, 'm');
```

See Also: 16.2

FONT_CHAR_WIDTH

Returns the width (`int`) of a specified character (`char`) of the font.

Return Type: `int`
Argument: `char`
Procs: `get`
Objects: `Xv_Font`
See Also: 16.2

FONT_DEFAULT_CHAR_HEIGHT

Returns the default character height of the font. Does not take a value.

Return Type: `int`
Procs: `get`
Objects: `Xv_Font`
Usage:

```
Xv_font   font;
int       height;
height = (int)xv_get(font, FONT_DEFAULT_CHAR_HEIGHT);
```

See Also: 16.2

FONT_DEFAULT_CHAR_WIDTH

Returns the default character width of the font. Does not take a value.

Return Type: `int`
Procs: `get`
Objects: `Xv_Font`
See Also: 16.2

FONT_FAMILY

Specifies the name of a font family.

Argument: `char *`
Default: None
Procs: `create`, `find`, `get`
Objects: `Xv_Font`
See Also: 16.1.1

FONT_INFO

Returns a pointer to the X structure `XFontStruct` containing X-related information for the font.

Return Type: `XFontStruct *`
Default: N/A
Procs: `get`
Objects: `Xv_Font`
Usage:

```
#include <X11/Xlib.h>
Xv_font       font;
XFontStruct  *font_info;
font_info =  (XFontStruct *)xv_get(font, FONT_INFO);
```

See Also: 16.2

XView Attributes

FONT_NAME

Specifies the name of the font desired. This takes precedence over all other Font attributes. The list of valid names that can be used can be displayed using the xlsfonts command.

Argument: char *
Procs: create,find
Objects: Xv_Font
Usage:

```
Xv_font   font;

font = xv_create (frame, FONT,
  FONT_NAME,
"-adobe-courier-bold-r-normal--14-140-75-75-m-90-iso8859-1"
  NULL) ;
height = (int)xv_get(font, FONT_DEFAULT_CHAR_HEIGHT);
```

See Also: 16.1.4

FONT_PIXFONT

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*. Returns the pixfont representation of the font.

FONT_RESCALE_OF

Given an existing font and a rescale factor, the returned font will be a similar font in the specified scale.

Argument1: Xv_Font
Argument2: Window_rescale_state (see <*xview/window.h*>)
Procs: create, find
Objects: Xv_Font
Usage:

```
Xv_font   font1,  font2;

/*
 * Find a font similar to font1 but in the large
 * scale
 */
font2 = (Xv_FONT) xv_find (frame, FONT,
               FONT_RESCALE_OF, font1, WIN_SCALE_LARGE,
               NULL) ;
```

See Also: 16.1.3, FONT_SIZES_FOR_SCALE, FONT_SCALE

FONT_SCALE

Specifies the scale desired for a font. The scale settings map to certain pixel sizes (defaults are 10, 12, 14, and 19). If FONT_SIZE is used, it will take precedence over FONT_SCALE.

Argument: Window_rescale_state (see <*xview/window.h*>)
Default: WIN_SCALE_MEDIUM
Procs: create, find, get
Objects: Xv_Font
See Also: 16.1.3, FONT_RESCALE_OF, FONT_SIZES_FOR_SCALE

FONT_SIZE

Specifies the size of a font in pixels. Note that the valid values for size depend on what font sizes are available on the X server.

Argument: `int`
Default: `12`
Procs: `create`, `find`, `get`
Objects: `Xv_Font`
Usage:

```
Xv_font   font;

/*
 * This creates a lucida font with normal style
 * with its size = 14
 */
font = (Xv_FONT) xv_find (frame, FONT
               FONT_FAMILY_LUCIDA,
               FONT_STYLE, FONT_STYLE_NORMAL
               FONT_SIZE, 14,
               NULL) ;
```

See Also: 16.1.2

FONT_SIZES_FOR_SCALE

Specifies a set of four integral sizes (measured in *points*) to which a font can be scaled.

Argument 1: `int`, for fonts scaled in small size
Argument 2: `int`, for fonts scaled in medium size
Argument 3: `int`, for fonts scaled in large size
Argument 4: `int`, for fonts scaled in extra_large size
Default: `10`, `12`, `14`, `19`
Procs: `create`, `find`
Objects: `Xv_Font`
Usage:

```
Xv_font              font;

/*
 * This creates a lucida font with normal style
 * with its size = 19
 */
font = (Xv_FONT) xv_find (frame, FONT,
          FONT_FAMILY_LUCIDA,
          FONT_STYLE, FONT_STYLE_NORMAL,
          FONT_SIZES_FOR_SCALE, 12, 14, 19, 26,
          FONT_SCALE, WIN_SCALE_LARGE,
          NULL) ;
```

See Also: 16.1.3, `FONT_RESCALE_OF`, `FONT_SCALE`

XView Attributes

FONT_STRING_DIMS

Given a string and the address of a `Font_string_dims` structure (see *<xview/font.h>*), `xv_get()` fills it in with the width and height dimensions of the string. The pointer to the structure is returned by `xv_get`.

Return Type: `Font_string_dims *`
Argument1: `char *`
Argument2: `Font_string_dims *`
Procs: `get`
Objects: `Xv_Font`
Usage:

```
Xv_font            font;
Font_string_dims   dims;

(void)xv_get (font, FONT_STRING_DIMS, "Hello World", dims);

/*
 * At this point 'dims' will contain the width and height
 * (in pixels) of the entire string "Hello World" as
 * rendered in the font 'font'.
 */
```

See Also: 16.2

FONT_STYLE

Specifies a font style.

Argument: `char *`
Default: `FONT_STYLE_NORMAL`
Procs: `create`, `find`, `get`
Objects: `Xv_Font`
See Also: 16.1.1

FRAME_ACCELERATOR

Specifies a window-level accelerator. The character in the first argument is used to call the procedure in the second argument with the data in the third argument. On `xv_get`, the frame package searches through the linked list of accelerators, and returns a pointer to the `Frame_accelerator` structure whose code or keysym matches the specified code and keysym. Applications do not normally use `xv_get` with this attribute. Note: Certain key combinations are reserved for semantic actions within XView and should not be used for accelerators. These key combinations are listed in Appendix C, *Mouseless Model Keyboard Mappings*, of the *XView Programming Manual*.

On `create,set`:

Argument 1: `char`
Argument 2: `void (*)()`
Argument 3: `Xv_opaque`

On `get`:

Argument 1: `char`
Argument 2: KeySym
Procs: `create,get,set`
Objects: `Frame`

Callback:

```
void
accelerator_notify_proc(value, event)
    Xv_opaque   value;    /* from Argument 3 */
    Event      *event;
```

See Also: 6.14

FRAME_BACKGROUND_COLOR

This attribute is obsolete. To change a frame's background color, create a cms and set it on the frame.

FRAME_BUSY

Sets label to gray and changes cursor to hour-glass.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `Frame`
See Also: 4.5

FRAME_CLOSED

Controls the frame's mapped state (either open or iconic).

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `Frame`
See Also: 14.2, `XV_SHOW`

FRAME_CLOSED_RECT

Sets the *size* of the frame's icon. To comply with the normal window manager placement strategy, if only the size of the icon is being set and not its screen location, first get the current `FRAME_CLOSED_RECT` `rect` and use the retrieved `rect.r_top` and `rect.r_left` values in the call to set `FRAME_CLOSED_RECT`.

Argument: `Rect *`
Default: `rect.r_width` = 64
`rect.r_height` = 64
`rect.r_left` = window manager dependent value
`rect.r_top` = window manager dependent value
Procs: `create`, `get`, `set`
Objects: `Frame`
See Also: 4.2.3

FRAME_CMD_DEFAULT_PIN_STATE

This attribute controls the initial state of the pin when the frame goes from unmapped (withdrawn) to mapped state. It is valid for both mapped and unmapped frames. However, if the frame is currently mapped, the effects of the change will be visible only on the next transition from unmapped to mapped state.

Argument: `int`
Valid Values: {`FRAME_CMD_PIN_IN, FRAME_CMD_PIN_OUT`} defined in *<xview/frame.h>* .
Procs: `create,get,set`
Objects: `Frame_cmd`
See Also: 4.3.2, `FRAME_CMD_PIN_STATE`

XView Attributes

FRAME_CMD_PANEL

Gets the default panel in the command frame.

Return Type: `Panel`
Procs: `get`
Objects: `Frame`
See Also: 11.12

FRAME_CMD_PIN_STATE

This attribute returns the current state of the pin. It is valid for both mapped and unmapped frames. Though, for unmapped frames it will always return `FRAME_CMD_PIN_OUT`.

Argument: `int`
Valid Values: {`FRAME_CMD_PIN_IN, FRAME_CMD_PIN_OUT`}defined in *<xview/frame.h>*
Procs: `get`, `set`
Objects: `Frame_cmd`
See Also: 4.3.2, `FRAME_CMD_DEFAULT_PIN_STATE`

FRAME_CMD_PUSHPIN_IN

This attribute is obsolete. It is supported only for compatibility reasons. Applications should use `FRAME_CMD_DEFAULT_PIN_STATE` and `FRAME_CMD_PIN_STATE` instead. Indicates whether the pushpin is in or out.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `Frame`
See Also: `FRAME_CMD_PIN_STATE`

FRAME_DEFAULT_DONE_PROC

The default procedure is to set the subframe to `WIN_SHOW`, `FALSE`.

Default: `FRAME_DEFAULT_DONE_PROC`
Procs: `create`, `get`, `set`
Objects: `Frame`
See Also: 4.3.3, `FRAME_DONE_PROC`

FRAME_DONE_PROC

Names a procedure to be called when the command frame is dismissed by the user (by taking the pushpin out).

Argument: `void (*`*frame_done_proc*`)()`
Default: `FRAME_DEFAULT_DONE_PROC`
Procs: `create`, `get`, `set`
Objects: `Frame`
Callback:

```
void
frame_done_proc(frame)
     Frame  frame;
```

See Also: 4.3.3

FRAME_FOCUS_DIRECTION
The direction in which the Location Cursor is pointing.

Argument: `Frame_focus_direction`(enum)
Valid Values: {`FRAME_FOCUS_RIGHT`, `FRAME_FOCUS_UP`}
Procs: `create,set,get`
Objects: `Frame`
See Also: 6.13.4

FRAME_FOCUS_WIN
Returns the handle of the Location Cursor (focus) window.

Return Type: `Xv_window`
Procs: `get`
Objects: `Frame`
See Also: 6.13.4

FRAME_FOREGROUND_COLOR
This attribute is obsolete. To change a frame's foreground color, create a cms and set it on the frame.

FRAME_ICON
Identifies the base frame's icon.

Argument: `Xv_opaque`
Default: A default empty icon.
Procs: `create`, `get`, `set`
Objects: `Frame`
See Also: 4.2.3

FRAME_INHERIT_COLORS
This attribute is obsolete. To implement this functionality, use `WIN_INHERIT_COLORS` on the frame.

FRAME_LABEL
Specifies the label used in the window manager's titlebar for the frame. XView copies the string on `set`.

Argument: `char *`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Frame`
See Also: 4.2.2, `FRAME_SHOW_LABEL`

FRAME_LEFT_FOOTER
Specifies the left-justified footer. XView copies the string on set.

Argument: `char *`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Frame`
See Also: `FRAME_SHOW_FOOTER`, `FRAME_RIGHT_FOOTER`

FRAME_MAX_SIZE

This attribute is similar to FRAME_MIN_SIZE, but allows the application programmer to specify a maximum size the frame can be resized to by a user. All other aspects of this attribute are exactly the same as FRAME_MIN_SIZE.

Return Type: void
Argument 1: int (maximum width of frame)
Argument 2: int (maximum height of frame)
Default: The default value is 0. In other words there is no application specified minimum or maximum size. Keep in mind that the window manager may impose one though.
Procs: create,get,set
Objects: Frame
See Also: 4.3.5

FRAME_MENUS

For create and set, this replaces current menu list with the one passed on the avlist. For get, this returns the current list of menus. The list returned should not be modified by the application. The number of menus can be obtained with FRAME_MENU_COUNT.

Argument: Null terminated list of Menus. On get, this returns (Menu *).
Default: None
Procs: create, get, set
Objects: Frame
Usage:

```
xv_set(frame1, FRAME_MENUS, edit_menu, load_menu,
             NULL,
             NULL);
```

FRAME_MENU_COUNT

Returns the current number of menus registered on the frame using FRAME_MENUS, FRAME_MENU_ADD, or FRAME_MENU_DELETE.

Argument: int
Default: None
Procs: get
Objects: Frame
Usage:

```
int   menu_count;
menu_count = (int)xv_get(frame2, FRAME_MENU_COUNT);
```

FRAME_MENU_ADD

Appends to the list of accelerated menus on the frame.

Argument: Menu
Default: None
Procs: create, set
Objects: Frame
Usage:

```
xv_set(frame1,
       FRAME_MENU_ADD, print_menu,
       NULL);
```

FRAME_MENU_DELETE

Deletes from the list of accelerated menus on the frame.

Argument: `Menu`
Default: None
Procs: `create`, `set`
Objects: `Frame`
Usage:

```
xv_set(frame1,
FRAME_MENU_DELETE, print_menu,
NULL);
```

FRAME_MIN_SIZE

This attribute allows the application programmer to specify a minimum size a frame can be resized to by a user. The `FRAME_MIN_SIZE` attribute takes two integer parameters, specifying the minimum width and height of the `WM_NORMAL_HINTS` property. Note that the minimum size is only a hint to the window manager. Some window managers may choose to ignore certain application specified hints. Setting both the minimum width and height to 0 effectively removes any application controlled minimum restriction on size.

Return Type: void
Argument 1: `int` (minimum width of frame)
Argument 2: `int` (minimum height of frame)
Default: The default value is 0. In other words there is no application specified minimum or maximum size. Keep in mind that the window manager may impose one .
Defaults to a pre-determined number of rows and columns for a `FILE_CHOOSER`. It is recommended that clients who extend the `FILE_CHOOSER` update the `FRAME_MIN_SIZE` accordingly. Note: the client may increase the min size of the Frame, but decreasing it below the default values is not supported. Should not be set at create time for a File Chooser.
Procs: `create,get,set`
Objects: `Frame`
Usage: To get the previous set values of `FRAME_MIN_SIZE` use `xv_get()` and pass in two parameters:

```
int width, height;

(void)xv_get(frame, FRAME_MIN_SIZE, &width, &height);
```

Note that some window managers may choose to ignore changes to the `WM_NORMAL_HINTS` property on frames that are already mapped. Thus, depending on the window manager, it may be necessary to unmap and then remap the frame before the `FRAME_MIN_SIZE` values take effect.
See Also: 4.3.5

FRAME_NEXT_PANE

Set the input focus to the next pane that can accept input focus.

Argument: None
Valid Values: {`FRAME_FOCUS_RIGHT`, `FRAME_FOCUS_UP`}
Procs: `set`
Objects: `Frame`

XView Attributes

FRAME_NO_CONFIRM

Controls whether a notice is displayed when a frame is destroyed.

Argument: `Bool`
Default: `TRUE`
Procs: `create`, `get`, `set`
Objects: `Frame`
See Also: 4.2.4

FRAME_NTH_SUBFRAME

Gets the frame's `n`th (from 1) subframe. Returns `NULL` if requested `Subframe` does not exist.

Return Type: `Frame`
Argument: `int`
Procs: `get`
Objects: `Frame`
Usage:

```
Frame  frame;
frame = xv_get (base_frame, FRAME_NTH_SUBFRAME, 1);
```

See Also: 4.8, `FRAME_NTH_SUBWINDOW`

FRAME_NTH_SUBWINDOW

Gets the frame's `n`th (from 0) subwindow. Returns `NULL` if requested `Subwindow` does not exist.

Return Type: `Xv_Window`
Argument: `int`
Procs: `get`
Objects: `Frame`
See Also: 4.8, `FRAME_NTH_SUBFRAME`

FRAME_PREVIOUS_ELEMENT

When set on a canvas, the focus is set to the view's horizontal scrollbar, vertical scrollbar, or the last element in the canvas.

Argument: None
Procs: `set`
Objects: `Frame`

FRAME_PREVIOUS_PANE

Set the input focus to the previous pane that can accept input focus.

Argument: None
Procs: `set`
Objects: `Frame`

FRAME_RIGHT_FOOTER

Specifies the right-justified footer. XView copies the string on set.

Argument: `char *`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Frame`
See Also: `FRAME_SHOW_FOOTER` , `FRAME_LEFT_FOOTER`

FRAME_SHOW_FOOTER

Indicates whether the footer is visible. The footer will inherit the colormap segment and foreground color of the frame only if the colormap segment is a control colormap segment.

Argument: Bool
Default: FALSE (TRUE for a FILE_CHOOSER object)
Procs: create, get, set
Objects: Frame
See Also: 4.2.2, FRAME_LEFT_FOOTER, FRAME_RIGHT_FOOTER, CMS_CONTROL_CMS

FRAME_SHOW_HEADER

Indicates whether the header is visible. This is only a hint to the window manager. Some window managers may not honor this hint, some window managers may only honor this hint when the frame leaves the withdrawn state. Thus, to see the effect of setting this attribute to FALSE, the frame may need to be unmapped, and then mapped again.

Argument: Bool
Default: TRUE
Procs: create, get, set
Objects: Frame
See Also: 4.2.2

FRAME_SHOW_LABEL

Indicates whether the frame's label is displayed. This is equivalent to FRAME_SHOW_HEADER.

Argument: Bool
Default: TRUE
Procs: create, get, set
Objects: Frame

FRAME_SHOW_RESIZE_CORNER

Determines whether a frame has resize corners. This is only a hint to the window manager. Some window managers may not honor this hint, some window managers may only honor this hint when the frame leaves the withdrawn state. Thus, to see the effect of setting this attribute to FALSE, the frame may need to be unmapped, and then mapped again. This attribute has no effect on window managers that are not OPEN LOOK compliant.

Argument: Bool
Default: TRUE
Procs: create, get, set
Objects: Frame
See Also: 4.3.4

FRAME_WM_COMMAND_ARGC

Returns the number of command-line option strings stored on the frame.

Return Type: int
Default: 0
Procs: get
Objects: Frame
See Also: 4.11, FRAME_WM_COMMAND_ARGC_ARGV, FRAME_WM_COMMAND_ARGV, FRAME_WM_COMMAND_STRINGS, WIN_CMD_LINE, Chapter 12, *Interclient Communication*, in the *Xlib Programming Manual.*

XView Attributes

FRAME_WM_COMMAND_ARGC_ARGV

Lets an application set the command-line options that can be used to (re)start it. The options passed, in addition to XView options, are stored on a property called WM_COMMAND on the frame window. The options passed are stored by XView and will be added to the XView options on the WM_COMMAND property on the frame window, upon receiving a WM_SAVE_YOURSELF request from the session/window manager. The program *xprop* can be used to display a window's properties. Only one base frame window of the application needs to have this property set. This property is read possibly by a session manager to restart clients. Setting this attribute's arguments to NULL and -1 prevents any command-line option information from being saved on the frame. If there are two or more base frames in the application, the second and subsequent base frames should set their FRAME_WM_COMMAND_ARGC_ARGV attributes' arguments to NULL and -1 if they want to avoid multiple invocations of the same application by the session manager. The first argument is the number of strings passed in the second argument. The second argument is a pointer to an array containing the command-line option strings. The strings passed are copied and cached on the frame.

Argument1: int
Argument2: (char **)
Default: 0 for Argument1 NULL for Argument2
Procs: create,set
Objects: Frame
Usage:

```
Frame   base_frame, second_frame;
char   *argv[10];
int     argc = 0;

argv[argc++] = "-I"
argv[argc++] = "ls"
argv[argc++] = "-bold_font"
argv[argc++] = "courier-bold-14"

/*
 * This ensures that the above options are stored
 * on the base frame
 */
 xv_set(base_frame, FRAME_WM_COMMAND_ARGC_ARGV,
                          argc, argv, NULL);
/*
 * This ensures that no command-line information will
 * be stored on this frame.
 */
 xv_set(second_frame,
           FRAME_WM_COMMAND_ARGC_ARGV, NULL, -1, NULL);
```

See Also: 4.11, 20.9.8, FRAME_WM_COMMAND_ARGV, FRAME_WM_COMMAND_ARGC, FRAME_WM_COMMAND_STRINGS, WIN_CMD_LINE, Chapter 12, *Interclient Communication*, in the *Xlib Programming Manual*.

FRAME_WM_COMMAND_ARGV

Returns the array containing the command-line option strings stored on the frame. The strings in the array must not be modified by client programs. If the value returned is -1, this means that no command-line information is stored on the frame.

Return Type: (`char **`) or -1
Default: `NULL`
Procs: `get`
Objects: `Frame`
See Also: 4.11, `FRAME_WM_COMMAND_ARGC_ARGV`, `FRAME_WM_COMMAND_ARGC`, `FRAME_WM_COMMAND_STRINGS`, `WIN_CMD_LINE`, Chapter 12, *Interclient Communication*, in the *Xlib Programming Manual*.

FRAME_WM_COMMAND_STRINGS

Lets an application set the command-line options that can be used to (re)start it. The options passed, in addition to XView options are stored on a property called `WM_COMMAND` on the frame window. The options passed are stored by XView and will be added to the XView options on the `WM_COMMAND` property on the frame window, upon receiving a `WM_SAVE_YOURSELF` request from the session/window manager. (The program `xprop` can be used to display a window's properties.) Only one base frame window of the application needs to have this property set. This property is read, possibly by a session manager to restart clients. Setting this attribute to -1 prevents any command-line option information from being saved on the frame. If there are two or more base frames in the application, the second and subsequent base frames should set their `FRAME_WM_COMMAND_ARGC_ARGV` attributes to -1 if they want to avoid multiple invocations of the same application by the session manager. The strings passed are copied and cached on the frame.

Argument: `NULL`-terminated list of (`char *`) or, -1 followed by `NULL`
Default: None
Procs: `create,set`
Objects: `Frame`
Usage:

```
Frame   base_frame, second_frame;

/*  Ensure that the given options are stored on
 *  on the base frame
 */
xv_set(base_frame, FRAME_WM_COMMAND_STRINGS,
        "-I",
        "ls",
        "-bold_font",
        "courier-bold-14",
        NULL,
   NULL);

/* This ensures that no command-line information will
 * be stored on this frame.
 */
xv_set(second_frame, FRAME_WM_COMMAND_STRINGS,
                           -1, NULL, NULL);
```

See Also: 4.11, `FRAME_WM_COMMAND_ARGC_ARGV`, `FRAME_WM_COMMAND_ARGC`, `FRAME_WM_COMMAND_ARGV`, `WIN_CMD_LINE`, Chapter 12, *Interclient Communication*, in the *Xlib Programming Manual*.

FRAME_X_ACCELERATOR

Specifies a window-level accelerator. The keysym in the first argument is used to call the procedure in the second argument with the data in the third argument. On `xv_get`, the Frame package searches through the linked list of accelerators, and returns a pointer to the `Frame_accelerator` structure whose code or keysym matches the specified code and keysym. Applications do not normally use `xv_get` with this attribute. Note: Certain key combinations are reserved for semantic actions within XView and should not be used for accelerators. These key combinations are listed in Appendix C, *Mouseless Model Keyboard Mappings* of the *XView Programming Manual*.

Argument 1: KeySym
Argument 2: `void (`**accelerator_notify_proc*`)()`
Argument 3: `Xv_opaque`
Procs: `create,get,set`
Objects: `Frame`
Callback:

```
void
accelerator_notify_proc(value, event)
Xv_opaque   value;    /* from Argument 3 */
Event      *event;
```

See Also: 6.14

FULLSCREEN_ALLOW_EVENTS

When in a fullscreen grab and the pointer and/or the keyboard is "frozen," specifies how the events that are queued up in the server due to X grabs, are processed.

Argument: `int` (where the value is any of the `AllowEvents` modes in *<X11/X.h>*)
Procs: `create`, `set`
Objects: `Fullscreen`
See Also: `XAllowEvents()`, *Xlib Reference Manual*.

FULLSCREEN_ALLOW_SYNC_EVENT

When in a synchronous grab mode, where the keyboard and/or pointer is frozen by a client; specifies that normal event processing continue until the next mouse button or keyboard event.

Argument: No value
Procs: `create`, `set`
Objects: `Fullscreen`

FULLSCREEN_CURSOR_WINDOW

Restricts the cursor to a specified window. `xv_get` identifies the window.

Argument: `Xv_window`
Default: None
Procs: `create`, `get`
Objects: `Fullscreen`

FULLSCREEN_GRAB_KEYBOARD

Specifies whether keyboard should be grabbed.

Argument: `Bool`
Default: `TRUE`
Procs: `create`, `get`, `set`
Objects: `Fullscreen`

FULLSCREEN_GRAB_POINTER

Specifies whether pointer should be grabbed.

Argument: Bool
Default: TRUE
Procs: create, get, set
Objects: Fullscreen

FULLSCREEN_GRAB_SERVER

Specifies whether server should be grabbed.

Argument: Bool
Default: TRUE
Procs: create, get, set
Objects: Fullscreen

FULLSCREEN_INPUT_WINDOW

Specifies the window from which input is read; xv_get identifies that window. The server, keyboard, and pointer are grabbed for this window.

Argument: Xv_window
Default: Owner window
Procs: create, get
Objects: Fullscreen

FULLSCREEN_KEYBOARD_GRAB_KBD_MODE

Determines the grab mode for the keyboard when grabbing the keyboard.

Argument: Fullscreen_grab_mode
Default: FULLSCREEN_ASYNCHRONOUS
Procs: create, get, set
Objects: Fullscreen

FULLSCREEN_KEYBOARD_GRAB_PTR_MODE

Determines the grab mode for the pointer when grabbing the keyboard.

Argument: Fullscreen_grab_mode
Default: FULLSCREEN_ASYNCHRONOUS
Procs: create, get, set
Objects: Fullscreen

FULLSCREEN_OWNER_EVENTS

When a window grabs the server/keyboard/pointer, the value of this attribute determines the distribution of events to the the application's windows.

All events occurring outside all the application's windows, are reported to the grab window (see FULLSCREEN_INPUT_WINDOW).

For events occurring within the application's windows: If FULSCREEN_OWNER_EVENTS is TRUE, the events are reported to the window indicated by the pointer. If FULSCREEN_OWNER_EVENTS is FALSE, the events are reported to the grab window.

Argument: Bool
Default: FALSE
Procs: create, get, set
Objects: Fullscreen

XView Attributes

FULLSCREEN_PAINT_WINDOW

Specifies or gets the paint window that will be or is already in fullscreen.

Argument: `Xv_opaque`
Default: Owner window
Procs: `create`, `get`
Objects: `Fullscreen`

FULLSCREEN_POINTER_GRAB_KBD_MODE

Determines the grab mode for the keyboard when grabbing the pointer.

Argument: `Fullscreen_grab_mode`
Default: `FULLSCREEN_ASYNCHRONOUS`
Procs: `create`, `get`, `set`
Objects: `Fullscreen`

FULLSCREEN_POINTER_GRAB_PTR_MODE

Determines the grab mode for the pointer when grabbing the pointer.

Argument: `Fullscreen_grab_mode`
Default: `FULLSCREEN_ASYNCHRONOUS`
Procs: `create`, `get`, `set`
Objects: `Fullscreen`

FULLSCREEN_RECT

Returns a pointer to the rectangle containing the paint window that is currently fullscreen.

Argument: `Rect *`
Default: Owner window's bounding box
Procs: `get`
Objects: `Fullscreen`

FULLSCREEN_SYNC

Specifies whether to grab in synchronous (`TRUE`) or asynchronous (`FALSE`) mode.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `Fullscreen`

HELP_STRING_FILENAME

The name of a file containing a list of string pairs. The file is searched for in the directories listed in the environment variable `HELPPATH`. Each line in the file contains two words: the first word is the help string for which help is available, and the second word is of the form *file:target*, which XView uses to find the Spot Help text and More Help data. The first word must be less than 128 characters, and the second less than 64 characters. `HELP_STRING_FILENAME` is to be set on the paint window, or any of its owners, where the strings are to be painted.

Argument: `Char *`
Default: No string-help available
Procs: `create`, `get`, `set`
Objects: `Icon`
See Also: 23.2.3

HISTORY_ADD_FIXED_ENTRY

Add a string to the Fixed Space in the list. A Fixed String is always added to the bottom of the Fixed Space in the list. Passing a label of `NULL` will add a blank `Menu_item`. This attribute takes a pair of strings, the first of which is the label displayed in the Menu, and the second is the value of the menu item associated with this entry.

Argument 1: `char *`
Argument 2: `char *`
Default: None
Procs: `create`, `set`
Objects: `History_list`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

HISTORY_ADD_ROLLING_ENTRY

Add a string to the Rolling Space. Strings in the Rolling Space are stacked and will roll off after some specified number of strings are added. This attribute takes a pair of strings, the first of which is the label displayed in the Menu, and the second is the value of the menu item associated with this entry.

Argument 1: `char *`
Argument 2: `char *`
Default: None
Procs: `create`, `set`
Objects: `History_list`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

HISTORY_DUPLICATE_LABELS

Allow duplicate labels in the Rolling Space. Checks both Fixed and Rolling Space for duplicates.

Argument: `Bool`
Default: `TRUE`
Procs: `create`, `get`, `set`
Objects: `History_list`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

HISTORY_DUPLICATE_VALUES

Allow duplicate values in the Rolling Space. Checks both Fixed and Rolling Space for duplicates.

Argument: `Bool`
Default: `TRUE`
Procs: `create`, `get`, `set`
Objects: `History_list`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

HISTORY_FIXED_COUNT

Returns the number of entries currently in the Fixed Space.

Argument: `int`
Default: None
Procs: `get`
Objects: `History_list`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

HISTORY_INACTIVE

Sets/Gets the Inactive state for a specified row. The first argument is one of HISTORY_FIXED or HISTORY_ROLLING. The second argument is the row number, and the third argument is the state of the flag (set only). If the row does not exist, the xv_set call is ignored, or the xv_get call will return -1.

Argument 1: enum
Argument 2: int
Argument 3: int
Default: FALSE
Procs: get, set
Objects: History_list
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

HISTORY_LABEL

Returns the value or label from a specified space (defined HISTORY_FIXED or HISTORY_ROLLING), for a specified row number. If the specified row does not exist, NULL will be returned.

Argument 1: enum
Argument 2: int
Default: None
Procs: get
Objects: History_list
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

HISTORY_MENU_HISTORY_LIST

Specifies the HISTORY_LIST object that is associated with this HISTORY_MENU instance.

Argument: History_list
Default: NULL
Procs: create, get, set
Objects: History_menu
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

HISTORY_MENU_OBJECT

Retrieves the handle to the internal Menu. Note: the only supported use of this handle is to pass it to menu_show() or to another XView object (such as a PANEL_BUTTON) for it's menu argument.

Argument: No value
Procs: get
Objects: History_menu
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

HISTORY_NOTIFY_PROC

The callback invoked by the user's selection on the HISTORY_MENU.

Argument: int (*)()
Default: NULL
Procs: create, get, set
Objects: History_menu

Callback:

```
void
callback ( hm, label, value )
   History_menu  hm;
   char *        label;
   char *        value;
```

HISTORY_ROLLING_COUNT

Returns the number of entries currently in the Rolling Space.

Argument: `int`
Default: None
Procs: `get`
Objects: `History_list`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

HISTORY_ROLLING_MAXIMUM

Specifies the maximum strings in the Rolling Space before roll off actually occurs.

Argument: `int`
Default: 15
Procs: `create`, `get`, `set`
Objects: `History_list`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

HISTORY_VALUE

Returns the value or label from a specified space (defined `HISTORY_FIXED` or `HISTORY_ROLLING`), for a specified row number. If the specified row does not exist, `NULL` will be returned.

Argument 1: `int`
Argument 2: `int`
Default: None
Procs: `get`
Objects: `History_list`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

ICON_FONT

Specifies the icon's font.

Argument: `Xv_font`
Default: `lucida medium`
Procs: `create`, `get`, `set`
Objects: `Icon`

ICON_HEIGHT

Icon's height in pixels.

Argument: `int`
Default: 64
Procs: `create`, `get`, `set`
Objects: `Icon`

ICON_IMAGE
Sets or gets the remote image for icon’s image.

Argument: `Server_image`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Icon`
See Also: 14.2.0.1

ICON_IMAGE_RECT
Sets or gets the bounding box (`rect`) for the icon’s image.

Argument: `Rect *`
Default: Origin (`0`,`0`), width `64`, height `64`
Procs: `create`, `get`, `set`
Objects: `Icon`
See Also: 14.2

ICON_LABEL
Specifies the icon’s label.

Argument: `char *`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Icon`
See Also: 14.2.1

ICON_LABEL_RECT
Sets or gets the bounding box for the icon’s label. Relative to the icon, `0,0` is the upper-left corner of the icon.

Argument: `Rect *`
Default: Bottom left-hand corner of the icon
Procs: `create`, `get`, `set`
Objects: `Icon`
See Also: 14.2.1

ICON_MASK_IMAGE
The icon’s GC’s clipmask is set to this bitmap. If this is set, `ICON_TRANSPARENT` is set to `TRUE` as a side effect.

Argument: `Pixmap or Server_image`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `Icon`
See Also: 14.2.0.3

ICON_TRANSPARENT
Sets the background color of the icon to be the same as the workspace’s background color.

Argument: `int`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `Icon`
See Also: 14.2.0.2

ICON_TRANSPARENT_LABEL

Draws the given string into an icon using the foreground only. It does not affect any other pixels in the bounding box for each character. Creating, setting, and getting `ICON_TRANSPARENT_LABEL` is equivalent to creating, setting, and getting `ICON_LABEL` except that the string is drawn in the foreground color only.

Argument: `char *`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Icon`
See Also: 14.2.2

ICON_WIDTH

Icon's width in pixels.

Argument: `int`
Default: 64
Procs: `create`, `get`, `set`
Objects: `Icon`

MENU_ACCELERATOR

When used in a `create` or `set` call, sets an accelerator on a menu item. If an accelerator is *changed* with `set`, `FRAME_MENUS` must be set again before the accelerator will take effect (see `FRAME_MENUS`).

The accelerator string will be copied by XView.

`Get` will return the accelerator string. The returned string should not be modified.

Argument: `(char *)` The accelerator string. For example "Meta+f," "Meta+Ctrl+L."
Default: None
Procs: `create`, `get`, `set`
Objects: `Menu_item`

```
Menu   menu;

menu = xv_create(NULL, MENU
                       MENU_ITEM,
                       MENU_STRING, "Load",
                       MENU_NOTIFY_PROC, load_proc,
                       MENU_ACCELERATOR, "Meta+l",
                       NULL,
                       NULL);

Menu_item  load_item;

load_item = xv_create(NULL, MENU_ITEM,
                            MENU_STRING, "Load",
                            MENU_NOTIFY_PROC, load_proc,
                            MENU_ACCELERATOR, "Meta+l",
                            NULL);
```

MENU_ACTION_ACCELERATOR

Creates a menu item with a given label, notify procedure and accelerator. If an accelerator is *changed* with `set`, `FRAME_MENUS` must be set again before the accelerator will take effect (see `FRAME_MENUS`).

The menu item label string, argument 1, will not be copied by XView. The accelerator string, argument 3, will be copied by XView.

Argument 1: `(char *)` Menu item label
Argument 2: `void (*)()` Notify procedure to call for this accelerator.
Argument 3: `(char *)` The accelerator string for example, “Meta+f,” “Meta+Ctrl+L.”
Default: None
Procs: `create`, `set`
Objects: `Menu`
Usage:

```
xv_set(menu,
        MENU_ACTION_ACCELERATOR,
            "Load", load_proc, "Meta+L",
        NULL);
```

MENU_ACTION_IMAGE

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

MENU_ACTION_ITEM

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

MENU_APPEND_ITEM

Appends an item to the end of menu.

Argument: `Menu_item`
Default: N/A
Procs: `set`
Objects: `Menu`
See Also: 11.6.3

MENU_CLASS

Gets an enumerated type that identifies the menu class, as set by the package. This may be one of `MENU_CHOICE`, `MENU_COMMAND`, or `MENU_ TOGGLE`.

Argument: `Menu_class`
Default: Defined by package
Procs: `get`
Objects: `Menu`

MENU_CLIENT_DATA

Specifies an arbitrary value to be attached to a menu or a menu item.

Argument: `caddr_t`
Default: None
Procs: `create`, `get`, `set`
Objects: `Menu`, `Menu_item`
See Also: 11.17.3

MENU_COLOR

Specifies the color index to use for the foreground color for a menu or a menu item.

Argument: `int`
Default: `size`-1 (where `size` is the number of colors in the colormap segment associated with the window), for a menu.
Same as the color index specified in the menu item’s menu or for the window, for a menu item.
Procs: `create`, `get`, `set`
Objects: `Menu`, `Menu_item`

MENU_COL_MAJOR

If TRUE, string items in the menu will be sorted in column-major order (like ls (1)) instead of row-major order. This attribute does not apply unless the menu uses multiple columns.

Argument: Bool
Default: TRUE
Procs: create, get, set
Objects: Menu

MENU_DEFAULT

Default menu item as a position. The first menu item has position one. Note that a menu title is also a menu item.

Argument: int
Default: 1, without a menu title; 2, with a menu title
Procs: create, get, set
Objects: Menu

MENU_DEFAULT_ITEM

Default menu item as opaque handle.

Argument: Menu_handle
Default: Handle of first non-title item
Procs: create,get,set
Objects: Menu
See Also: 11.15

MENU_DESCEND_FIRST

If this attribute is specified in xv_find, then the search will be done "depth first." If it is not specified, the search will be "deferred"; that is, it will be done horizontally through the menu structure.

Argument: No value
Default: Deferred
Procs: find
Objects: Menu
See Also: 11.14

MENU_DONE_PROC

Specifies a callback procedure that is called when the menu group is dismissed.

Argument void (*menu_done_proc)()
Default: menu_return_value(), for a menu object; NULL, for a menu item object
Procs: create, get, set
Objects: Menu
Callback:

```
void
menu_done_proc(menu, result)
    menu       menu;
    Xv_opaque  result;
```

menu is the base (top) level menu, as specified in menu_show().
result is the return value from the menu notify procedure.

XView Attributes

If the menu notify procedure is user-specified, then `result` is invalid, since user-specified menu notify procedures do not return a value. If the menu notify procedure is `menu_return_value()`; then `result` is the value of the selected menu item. If the menu notify procedure is `menu_return_item()`, the `result` is the handle of the selected menu item. You can get notification that any menu in a menu group is done by attaching `MENU_DONE_PROC` to each menu. However, you will get better results with `menu_item` notify procedures.

See Also: 11.4

MENU_FEEDBACK

This attribute is for SunView compatibility. For more informations, refer to the manual *Converting SunView Applications*.

MENU_FIRST_EVENT

Gets the event which was initially passed into `menu_show`. The event's contents *can* be modified.

Return Type: `Event *`
Procs: `get`
Objects: `Menu`
See Also: 11.4

MENU_GEN_PIN_WINDOW

Creates a command window as the pin window based on the menu's contents. The frame (argument 1) is the parent frame; the name (argument 2) is the pin window's name. All menu items *must* have notify procedures; `MENU_NOTIFY_PROC` for the menu itself is ignored.

Argument 1: `Frame`
Argument 2: `char *`
Default: No pin window
Procs: `create`, `set`
Objects: `Menu`
See Also: 11.12

MENU_GEN_PROC

Names a client-provided procedure that is called to generate a menu or menu item.

Argument: `void (`**menu_gen_proc*`)()`
Default: None
Procs: `create`, `get`, `set`
Objects: `Menu`, `Menu_item`
Callback:

```
Menu
menu_gen_proc(m, op)
      Menu           m;
      Menu_generate  op;
```

This menu generating procedure is called whenever a menu item that has the `MENU_GEN_PROC` attribute set and the menu needs to be displayed or traversed. It should return a handle to a menu that has either been dynamically created or statically stored. The *op* argument tells the state of the menu when the function is called. The argument *op* has one of the following values: `MENU_DISPLAY`, `MENU_DISPLAY_DONE`, `MENU_NOTIFY`, or `MENU_NOTIFY_DONE` as defined by **Menu_generate** in *openmenu.h*.

See Also: 11.9

MENU_GEN_PROC_IMAGE

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

MENU_GEN_PROC_ITEM

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

MENU_GEN_PULLRIGHT

Defines the generate procedure for the menu item's submenu.

Argument: `void (*menu_gen_proc)()`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Menu_item`
Callback:

```
Menu
menu_gen_proc(m, op)
     Menu          m;
     Menu_generate op;
```

This menu generating procedure is called whenever a menu item has `MENU_GEN_PROC` set and the menu needs to be displayed or traversed. It should return a handle to a menu that has either been dynamically created or statically stored.

See Also: 11.9

MENU_GEN_PULLRIGHT_IMAGE

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

MENU_GEN_PULLRIGHT_ITEM

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

MENU_IMAGE

Specifies the menu item's server image.

Argument: `Server_image`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Menu_item`
See Also: 11.6.1

MENU_IMAGE_ITEM

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

MENU_IMAGES

Creates menu items with the specified server images. The new menu items are appended to the menu.

Argument: list of `Server_image`
Default: None
Procs: `create`, `set`
Objects: `Menu`
See Also: 11.8

XView Attributes

MENU_INACTIVE

If `TRUE`, the menu item is grayed out and not selectable.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `Menu_item`
See Also: 11.13

MENU_INSERT

Inserts a new menu item after *n*th item in the menu.

Argument 1: `int`
Argument 2: `Menu_item`
Default: N/A
Procs: `create`, `get`, `set`
Objects: `Menu`

MENU_INSERT_ITEM

Inserts the menu item given as the next value after the menu item given as the first value.

Argument 1: `Menu_item`
Argument 2: `Menu_item`
Procs: `create`, `set`
Objects: `Menu`

MENU_ITEM

Allows you to create menu items *in-line* with the call to the `xv_create()` used to create your menu. Takes a *menu item*-specific attribute-value list that would otherwise be used in a separate call to `xv_create()` to create menu items with the `MENUITEM` package.

Argument: A-V list
Procs: `create`, `set`
Objects: `Menu`
Usage:

```
xv_create(NULL, MENU,
            MENU_ITEM,
               MENU_STRING, "foo",
               MENU_NOTIFY_PROC, foo_notify_proc,
               NULL,
            MENU_ITEM,
               MENU_STRING, "bar",
               MENU_NOTIFY_PROC, bar_notify_proc,
               NULL,
            NULL);
```

See Also: 11.6.1

MENU_LAST_EVENT

Gets the last event read by the menu. The event’s contents can be modified.

Return Type: `Event *`
Procs: `get`
Objects: `Menu`
See Also: 11.4

MENU_NCOLS

Specifies the number of columns in a menu.

Argument: `int`
Default: `1`
Procs: `create`, `get`, `set`
Objects: `Menu`
See Also: 11.11

MENU_NITEMS

Returns the number of items in a menu.

Return Type: `int`
Procs: `get`
Objects: `Menu`
See Also: 11.9

MENU_NOTIFY_PROC

Names a procedure to be called when the user selects a menu item.

Argument: `void (*`*menu_notify_proc*`)()`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Menu`, `Menu_item`
Callback:

```
void
menu_notify_proc(menu, menu_item)
      Menu         menu;
      Menu_item    menu_item;
```

The notify procedure is attached to menus and menu items using `MENU_NOTIFY_PROC`. This function is called whenever the user selects a menu item. The `menu` identifies which menu the `menu_item` belongs to.

See Also: 11.13

MENU_NOTIFY_STATUS

If the menu is attached to a menu button that is part of an unpinned pop-up window, then the window is dismissed if `MENU_NOTIFY_STATUS` is `XV_OK`. If `MENU_NOTIFY_STATUS` is set to `XV_ERROR`, then the window is not dismissed. You probably will only need to `xv_get()` or `xv_set()` this attribute from within a notify procedure. When a notify procedure exits for an unpinned command frame, XView internally uses the value of this attribute to determine whether or not the command frame is dismissed. XView sets the value of this attribute to `XV_OK` before calling the notify procedure.

Argument: `int`
Default: `XV_OK`
Valid Values: {`XV_OK`, `XV_ERROR`}
Procs: `get`, `set`
Objects: `Menu`
See Also: 11.16

XView Attributes

MENU_NROWS

Sets or gets the number of rows in a menu.

Argument:	`int`
Default:	The number of menu items in the menu
Procs:	`create`, `get`, `set`
Objects:	`Menu`
See Also:	11.11

MENU_NTH_ITEM

Gets the *n*th menu item (*n* starts at `1`).

Return Type:	`Menu_item`
Argument:	`int`
Procs:	`get`
Objects:	`Menu`
See Also:	11.9

MENU_PARENT

On a Menu object, this attribute is only valid from within a notify or generate procedure. With a submenu the return value is the Menu Item from which the submenu was pulled-right. For a top-level menu the `xv_get()` returns `NULL`. On a Menu_item object, the return value is the handle of the enclosing Menu.

Return Type:	`Menu` or `Menu_item`
Procs:	`get`
Objects:	`Menu`, `Menu_item`
See Also:	11.9.1

MENU_PIN

Determines whether the menu will have pushpin.

Argument:	`Bool`
Default:	`FALSE`
Procs:	`create`, `get`, `set`
Objects:	`Menu`
See Also:	11.12

MENU_PIN_PROC

Names a procedure called if a user chooses the pin menu item. The default procedure displays a window whose layout is similar to the menu it replaces.

Argument: `void (*menu_pin_proc)()`
Default: `menu_default_pin_proc()`
Procs: `create`, `get`, `set`
Objects: `Menu`
Callback:

```
void
menu_pin_proc(menu, x, y)
      Menu   menu;
      int    x, y;
```

This client-supplied procedure is called whenever the user attempts to pin-up a menu. Attached to menus using the attribute, `MENU_PIN_PROC`. `x` and `y` are the fullscreen coordinates of the upper-left corner of the pin window.

See Also: 11.12

MENU_PIN_WINDOW

The handle of the command frame representing the pin window for the menu when pinned-up. Use `MENU_GEN_PIN_WINDOW` to have XView manage this frame automatically. When using `MENU_GEN_PIN_WINDOW`, the value returned by `xv_get()` for `MENU_PIN_WINDOW` will be `NULL` until a menu is initially pinned.

Argument: `Frame_cmd`
Default: None
Procs: `create`, `get`, `set`
Objects: `Menu`
See Also: 11.12

MENU_PULLRIGHT

Item's pullright menu.

Argument: `Menu`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Menu_item`
See Also: 11.8

MENU_PULLRIGHT_IMAGE

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

MENU_PULLRIGHT_ITEM

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

MENU_RELEASE

Specifies that the menu item gets destroyed when its parent menu is destroyed. This is the default for menu items created in-line.

Argument: No value
Default: Destroy item, if in-line; do *not* destroy item if `Append` or `Replace`.
Procs: `create`, `set`
Objects: `Menu_item`
See Also: 11.17

MENU_RELEASE_IMAGE

Specifies that the string or `Server_image` associated with the item is freed when the item is destroyed.

Argument: No value
Default: Do not release text string or server image.
Procs: `create`, `set`
Objects: `Menu_item`
See Also: 11.8

MENU_REMOVE

Removes the *n*th item from the menu (*n* starts at 1).

Argument: `int`
Procs: `set`
Objects: `Menu`
See Also: 11.9

XView Attributes

MENU_REMOVE_ITEM

Removes the specified menu item.

Argument: `Menu_item`
Procs: `set`
Objects: `Menu`

MENU_REPLACE

Replaces the *n*th menu item (argument 1) with the menu item specified in argument 2 (*n* starts at 1).

Argument 1: `int`
Argument 2: `Menu_item`
Procs: `create`, `set`
Objects: `Menu`

MENU_REPLACE_ITEM

Replaces the menu item given as first value with the one given as the second value in the menu (the old item is *not* replaced in any other menus in which it may appear).

Argument 1: `Menu_item`
Argument 2: `Menu_item`
Procs: `create`, `set`
Objects: `Menu`

MENU_SELECTED

This attribute is valid for both a menu and a menu item. On a menu, it returns the selected menu item number. On a menu item, it returns `TRUE` if the item is selected, otherwise it returns `FALSE`.

Argument: `int` or `Bool`
Default: `FALSE`
Procs: `get`
Objects: `Menu, Menu_item`

MENU_SELECTED_ITEM

Returns the selected menu item.

Argument: `Menu_item`
Default: None
Procs: `get`
Objects: `Menu`
See Also: 11.15

MENU_STRING

Sets or gets the menu item's string. The string is *not* copied when this attribute is set.

Argument: `char *`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Menu_item`
See Also: 11.6.1

MENU_STRING_ITEM

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

MENU_STRINGS

Creates menu items with the specified strings. The new menu items are appended to the menu. The strings are *not* copied when this attribute is set.

Argument: list of `char *`
Procs: `create`, `set`
Objects: `Menu`
See Also: 11.5

MENU_STRINGS_AND_ACCELERATORS

Creates menu items with a given label and accelerator. If an accelerator is changed with `set`, `FRAME_MENUS` must be set again before the accelerator will take effect (see `FRAME_MENUS`).

The accelerator strings will be copied by XView. The menu item label strings will not be copied by XView.

Argument: Null terminated list of label(`char *`), accelerator(`char *`) pairs
Default: None
Procs: `create`, `set`
Objects: `Menu`

```
xv_set(menu, MENU_NOTIFY_PROC, file_proc,
             MENU_STRINGS_AND_ACCELERATORS,
                          "Load", "Meta+l",
                          "Print", "Meta+p",
                          "Include","Meta+Ctrl+i",
                          NULL,
             NULL);
```

MENU_TITLE

Specifies that the item is the menu's title. Returns `TRUE` or `FALSE` on `get`.

Return Type: `Bool`
Argument: No value
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `Menu_item`

MENU_TITLE_ITEM

Creates a string title item. For OPEN LOOK compliance, this attribute must *not* be used with menus that originate from pullright items or pulldown menu buttons. The string is not copied when this attribute is set.

Argument: `char *`
Procs: `create`, `set`
Objects: `Menu`
See Also: 11.5

MENU_TYPE

Returns `MENU_MENU` or `MENU_ITEM`. Informs you whether the object is a menu or a menu item.

Argument: `Menu_attribute` (an enum)
Default: `MENU_MENU`
Procs: `get`
Objects: `Menu, Menu_item`

MENU_VALID_RESULT

If TRUE, then a zero return value represents a legitimate value. This attribute is used only when the menu notify procedure that is invoked is the public procedure `menu_return_value()`.

Argument: `Bool`
Default: FALSE
Procs: `create`, `get`, `set`
Objects: `Menu`

MENU_VALUE

Sets or gets the item's value. This attribute is used only when the menu notify procedure that is invoked is the public procedure `menu_return_value()`.

Argument: `Xv_opaque`
Default: NULL
Procs: `create`, `get`, `set`
Objects: `Menu_item`

NOTICE_BLOCK_THREAD

Specifies the type of notice desired. If TRUE, the notice, when mapped (via XV_SHOW), will block the thread of execution.

Argument: `Bool`
Default: TRUE
Procs: `create,set`
Objects: `Xv_Notice`
See Also: 12.2

NOTICE_BUSY_FRAMES

Specifies the frames or windows to appear busy during notice pop-up. This applies only when NOTICE_LOCK_SCREEN is FALSE.

Argument: NULL-terminated list, `Frame`
Default: NULL
Procs: `create,set`
Objects: `Xv_Notice`
See Also: 12.2.1

NOTICE_BUTTON

Specifies a string to be displayed in a button and a value to use if the button is selected.

Return Type: `int` (`notice_prompt()` returns the value of the button selected)
Argument 1: `char *`
Argument 2: `int`
Default: None
Procs: `create,set,notice_prompt()`
Objects: `Xv_Notice`
See Also: 12.1.1

NOTICE_BUTTON_NO

Specifies a string associated with the `NO` button. The value returned if this button is selected is `NOTICE_NO`.

Return Type: `int` (`notice_prompt()` returns the value of the button selected)
Argument: `char *`
Default: None
Procs: `create,set,notice_prompt()`
Objects: `Xv_Notice`
See Also: 12.1.1

NOTICE_BUTTON_YES

Specifies a string to associate with the `YES` (confirm) button. The value returned when this button is selected is `NOTICE_YES`.

Return Type: `int` (`notice_prompt()` returns the value of the button selected)
Argument: `char *`
Default: None
Procs: `create,set,notice_prompt()`
Objects: `Xv_Notice`
See Also: 12.1.1

NOTICE_EVENT_PROC

Specifies the function to be called when a notice button is clicked on. This applies only when `NOTICE_LOCK_SCREEN` is `FALSE`.

Argument: `void (*`*my_notice_proc*`)()`
Default: `NULL`
Procs: `create,set`
Objects: `Xv_Notice`
Callback:

```
void
my_notice_proc(notice, value, event)
      Xv_Notice   *notice;
      int         value;
      Event       *event;
```

See Also: 12.2.1.1

NOTICE_FOCUS_XY

Specifies the `x,y` position from which the notice shadow emanates. The `x,y` position is relative to the *owner* window. This applies to a notice object only when `NOTICE_LOCK_SCREEN` is `TRUE`.

Argument 1: `int`
Argument 2: `int`
Default: Current mouse position
Procs: `create,set,notice_prompt()`
Objects: `Xv_Notice`
See Also: 12.2.2

XView Attributes

NOTICE_FONT

Specifies the font to be used in the notice.

Argument: `Xv_Font`
Default: The font of the owner frame
Procs: `create,get,notice_prompt()`
Objects: `Xv_Notice`

NOTICE_LOCK_SCREEN

Specifies the type of notice desired. If `TRUE`, the notice locks up the screen when mapped.

Argument: `Bool`
Default: `FALSE`
Procs: `create,get,set`
Objects: `Xv_Notice`
See Also: 12.2

NOTICE_MESSAGE_STRING

Specifies the text to print in a notice. The argument is a single `NULL`-terminated string, which may contain \n as a line break.

Argument: `(char *)`
Default: `NULL`
Procs: `create,set,notice_prompt()`
Objects: `Xv_Notice`
See Also: 12.1

NOTICE_MESSAGE_STRINGS

Specifies the text to print in a notice. Argument is a `NULL`–terminated list of strings, which may contain \n as a line break.

Argument: `list of char *`
Default: `NULL`
Procs: `create,set,notice_prompt()`
Objects: `Xv_Notice`
See Also: 12.1

NOTICE_MESSAGE_STRINGS_ARRAY_PTR

Specifies the text to print in a notice. The argument is a variable pointing to a `NULL`-terminated array of strings, which may contain \n as a line break.

Argument: `char**`
Default: `NULL`
Procs: `create,set,notice_prompt()`
Objects: `Xv_Notice`
See Also: 24.4.2, 12.1

NOTICE_NO_BEEPING

Allows a client to specify that no beeping should take place, regardless of the default resource database setting. The default for this option is `FALSE`; that is, beep the number of times specified in the database.

Argument: `int`
Default: `TRUE`
Procs: `create,get,set,notice_prompt()`
Objects: `Xv_Notice`
See Also: 12.2.2.1

NOTICE_STATUS

Specifies the address of the return code of the notice when it pops-down. If not specified, the notice return code is stored in the notice object and can be obtained by doing `(int)xv_get(notice NOTICE_STATUS)`. In `xv_set()` we pass in an `(int *)`, and the `xv_get()` returns an `(int)`.

Return Type: `(int)`
Argument: `(int *)`
Default: See description
Procs: `create,get,set`
Objects: `Xv_Notice`
See Also: 12.1.1

NOTICE_TRIGGER

Specifies an XView event ID or type, other than clicking on mouse buttons, that will cause the notice to pop down. The notice pops down and the thread of execution continues. This applies to a notice object only when `NOTICE_LOCK_SCREEN` is `TRUE`. When this event occurs, the value returned is `NOTICE_TRIGGERED`. The event parameter to `notice_prompt` contains specifics about the event that triggered it.

Argument: `int` (for example, `MS_LEFT`)
Default: N/A
Procs: `create,set,notice_prompt()`
Objects: `Xv_Notice`
See Also: 12.2.2.1, `NOTICE_TRIGGER_EVENT`

NOTICE_TRIGGER_EVENT

Specifies the address of the XView Event structure that caused the notice to pop down, the thread of execution continues. This is used with the `NOTICE_TRIGGER` attribute and applies only when `NOTICE_LOCK_SCREEN` is `TRUE`.

Argument: `(Event *)`
Default: N/A
Procs: `create,set`
Objects: `Xv_Notice`
See Also: 12.2.2.1, `NOTICE_TRIGGER`

OPENWIN_ADJUST_FOR_HORIZONTAL_SCROLLBAR

Reserves space in openwin-class objects (e.g., canvas) for a horizontal scrollbar. On `xv_set`, adjusts (extends) the height of a subwindow that does *not* have a horizontal scrollbar to align properly with one that does.

Argument: `Bool`
Default: `TRUE`
Procs: `create`, `get`, `set`
Objects: `Openwin`

OPENWIN_ADJUST_FOR_VERTICAL_SCROLLBAR

Reserves space in openwin-class objects (e.g., canvas) for a vertical scrollbar. On `xv_set`, adjusts (extends) the width of a subwindow that does *not* have a vertical scrollbar to align properly with one that does.

Argument: `Bool`
Default: `TRUE`
Procs: `create`, `get`, `set`
Objects: `Openwin`

OPENWIN_AUTO_CLEAR

If TRUE, exposed areas of windows are cleared (i.e., painted the background color) before the repaint procedure is called.

Argument: Bool
Default: TRUE unless subwindow's canvas is retained
Procs: create, get, set
Objects: Openwin
See Also: 5.3

OPENWIN_HORIZONTAL_SCROLLBAR

Returns the handle of the horizontal scrollbar associated with the specified view.

Return Type: Scrollbar
Argument: Xv_Window
Procs: create,get,set
Objects: Openwin
Usage:

```
Scrollbar  sb;
Xv_Window  view;
Openwin    openwin;

sb = (Scrollbar)xv_get(openwin,
          OPENWIN_HORIZONTAL_SCROLLBAR, view);
```

See Also: OPENWIN_VERTICAL_SCROLLBAR

OPENWIN_NO_MARGIN

If TRUE, the view window's two pixel bottom and right margins are turned off.

Argument: Bool
Default: FALSE
Procs: create, get, set
Objects: Openwin

OPENWIN_NTH_VIEW

Gets the handle of a specified openwin view window. Openwin view windows are numbered from zero.

Return Type: Xv_window
Argument: int
Procs: get
Objects: Openwin
See Also: 5.6.3.2

OPENWIN_NVIEWS

Gets the number of views contained in the open window.

Return Type: int
Default: 1
Procs: get
Objects: Openwin

OPENWIN_SHOW_BORDERS

Displays openwin borders. This must remain on for openwin objects to have scrollbars attached to them.

Argument: Bool
Default: TRUE (FALSE for scrollable panels)
Procs: create, get
Objects: Openwin

OPENWIN_SPLIT

Takes as its value a list of attribute-value pairs beginning with OPENWIN_SPLIT_.

Argument: A-V list
Procs: create, set
Objects: Openwin
Callback:

```
void
openwin_split_destroy_proc(openwin)
        Openwin    openwin;
```

OPENWIN_SPLIT_ used to install this procedure. This function is called when the user *joins* two views.

See Also: 5.6.2

OPENWIN_SPLIT_DESTROY_PROC

Names a procedure to call when a split openwin is joined. This attribute can only be set in an OPENWIN_SPLIT attribute-value list.

Argument void (**openwin_split_destroy_proc*)()
Default: NULL
Procs: create, get, set
Objects: Openwin
See Also: 5.6.3.1, OPENWIN_SPLIT

OPENWIN_SPLIT_DIRECTION

Sets the direction of the split either horizontally or vertically. This attribute can only be set in an OPENWIN_SPLIT attribute-value list.

Argument: Openwin_split_direction
Default: OPENWIN_SPLIT_HORIZONTAL
Procs: create, set
Objects: Openwin
See Also: OPENWIN_SPLIT

OPENWIN_SPLIT_INIT_PROC

Names a procedure to call when a split window is created. This attribute can only be set in an OPENWIN_SPLIT attribute-value list.

Argument void (**openwin_split_init_proc*)()
Default: NULL
Procs: create, get, set
Objects: Openwin
See Also: 5.6.3.1, OPENWIN_SPLIT

OPENWIN_SPLIT_POSITION

Sets the position (in pixels) of the view. This attribute can only be set in an `OPENWIN_SPLIT` attribute-value list.

Argument: `int`
Default: None
Procs: `create`,`set`
Objects: `Openwin`
See Also: `OPENWIN_SPLIT`

OPENWIN_SPLIT_VIEW

Specifies which view to split. Its value is the handle of the view you want to split. This attribute can only be set in an `OPENWIN_SPLIT` attribute-value list.

Argument: `Xv_window`
Procs: `set`
Objects: `Openwin`
See Also: `OPENWIN_SPLIT`

OPENWIN_SPLIT_VIEW_START

Specifies which part of the data (measured in scrollbar-units) is displayed at the start of the view (*top* for vertical; *left* for horizontal). This attribute can only be set in an `OPENWIN_SPLIT` attribute-value list.

Argument: `int`
Default: Continue from previous view
Procs: `create`,`get`,`set`
Objects: `Openwin`
See Also: `OPENWIN_SPLIT`

OPENWIN_VERTICAL_SCROLLBAR

Returns the handle of the vertical scrollbar associated with the specified view.

Return Type: `Scrollbar`

Argument: `Xv_Window`
Procs: `create`,`get`,`set`
Objects: `Openwin`
Usage:

```
Scrollbar  sb;
Xv_Window  view;
Openwin    openwin;

sb = (Scrollbar)xv_get(openwin,
          OPENWIN_VERTICAL_SCROLLBAR, view);
```

See Also: `OPENWIN_HORIZONTAL_SCROLLBAR`

OPENWIN_VIEW_ATTRS

Distributes modifications across all views in a given openwin. Note that this does not affect canvas paint windows.

Argument: A-V list
Procs: `create`, `set`
Objects: `Openwin`
See Also: `CANVAS_PAINTWINDOW_ATTRS`

PANEL_ACCEPT_KEYSTROKE

Specifies whether the panel background or panel item should consume keyboard events. When an item wants keystrokes, set `PANEL_ACCEPT_KEYSTROKE` to `TRUE` on the panel item in the item's init routine. Setting `PANEL_ACCEPT_KEYSTROKE` to `TRUE` on the panel background is not recommended, since this is not supported by OPEN LOOK. This mode is maintained for SunView1 compatibility. **Warning**: this attribute should *only* be used from within a panel item extension.

Argument: `Bool`
Default: `FALSE` for panel background. Default for panel items depends on the state of `OpenWindows.KeyboardCommands`. If it is SunView1 or Basic, then only `PANEL_TEXT`, `PANEL_NUMERIC_TEXT`, and `PANEL_MULTILINE_TEXT` accept keyboard input focus. If it is Full, then all panel items accept keyboard input focus.
Procs: `create`, `get`, `set`
Objects: `Panel`, `Panel_item`
See Also: 7.19.6

PANEL_BACKGROUND_PROC

Names an event-handling procedure called when an event falls on the background of the panel.

Argument: `void (*`*panel_background_proc*`)()`
Default: `panel_default_handle_event`
Procs: `create`, `get`, `set`
Objects: `Panel`
Callback:

```
void
panel_background_proc(panel, event)
        Panel    panel
        Event   *event
```

Event-handling procedure called when an event falls on the background of the panel (e.g., not on any panel items).

See Also: 7.19.6

PANEL_BORDER

Adds a border around the panel. In a 3D implementation, this border is two pixels wide and presents a "chiseled" appearance. In 2D, the border is one pixel wide. Since the border is rendered directly on the Panel, it is the job of the application to make sure that no Panel_item's overlap the border.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get` `set`
Objects: `Panel`
See Also: 7.3.2

PANEL_BLINK_CARET

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

XView Attributes

PANEL_BUSY

Sets a button's or a drop target item's busy state. While a button or drop target is busy, it will not accept further input (e.g., SELECT-down). By default, a button or Drop Target Item will be in the busy state while in its notify procedure and be cleared upon exiting. The busy state can be maintained after exiting the notify procedure by setting `PANEL_BUSY` to `TRUE` from within the notify procedure. Setting `PANEL_BUSY` back to `FALSE`, at a later time, clears the button's busy state. This attribute can also be used for similar functionality in panel item extensions.

Argument: `Bool`
Default: Set to `FALSE` prior to entering button's or drop target's notify procedure
Procs: `get,set`
Objects: `Panel_button_item`, `Panel_item`, `Panel_drop_target_item`
See Also: 7.9.1

PANEL_CARET_ITEM

Specifies the panel item that currently has the input focus.

Argument: `Panel_item`
Default: First item that can accept keyboard input
Procs: `create`, `get`, `set`
Objects: `Panel`
See Also: 7.15.1

PANEL_CHILD_CARET_ITEM

Specifies what embedded (child) panel item to set the keyboard focus to when the application sets `PANEL_CARET_ITEM` to the parent panel item. `PANEL_CHILD_CARET_ITEM` should be `NULL`, if there are no embedded (child) panel items, or if the parent panel item itself can take the keyboard focus.

Argument: `Panel_item`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Panel_item` with an embedded (child) panel item.

PANEL_CHOICE_COLOR

Specifies the foreground color index (argument 2) for the specified choice (argument 1).

Argument 1: `int` (choice index)
Argument 2: `int` (color_index)
Default: Panel foreground color
Procs: `create,get,set`
Objects: `Panel_choice_item`
Usage:

```
xv_set(pchoice, PANEL_CHOICE_COLOR, 1, RED, NULL);
```

See Also: 7.10.6, 21.5

PANEL_CHOICE_FONT

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

PANEL_CHOICE_FONTS

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

PANEL_CHOICE_IMAGE

Specifies the image (argument 2) for the specified choice (argument 1).

Argument 1: `int`
Argument 2: `server_image`
Default: None
Procs: `create`, `get`, `set`
Objects: `Panel_choice_item`
See Also: 7.10.2

PANEL_CHOICE_IMAGES

Specifies the image for each of several choices.

Argument: `NULL`-terminated list of `Server_image`s
Default: `NULL`
Procs: `create`, `set`
Objects: `Panel_choice_item`
See Also: 7.10

PANEL_CHOICE_NCOLS

Specifies the number of columns to use in the layout of panel choices.

Argument: `int`
Default: 1 for vertical layout (`PANEL_LAYOUT`)
Procs: `create`, `get`, `set`
Objects: `Panel_choice_item`
See Also: 7.10.1

PANEL_CHOICE_NROWS

Specifies the number of rows to use in the layout of panel choices.

Argument: `int`
Default: 1 for horizontal layout (`PANEL_LAYOUT`)
Procs: `create`, `get`, `set`
Objects: `Panel_choice_item`
See Also: 7.10.1

PANEL_CHOICE_RECT

The rectangle that encloses the specified choice. The first argument is the index of the choice (0 = first choice). This attribute is not valid for `PANEL_CHOICE_STACK` objects.

Argument: `int`
Return Type: `Rect *`
Procs: `get`
Objects: `Panel_choice_item`
See Also: 7.10.7

PANEL_CHOICE_STRING

Specifies the string (argument 2) for the specified choice (argument 1). The string is copied when this attribute is set.

Argument 1: `int`
Argument 2: `char *`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Panel_choice_item`
See Also: 7.10

PANEL_CHOICE_STRINGS

Specifies the string for each choice. You must specify at least one choice. The least you can specify is a single NULL-terminated string. The strings are copied when this attribute is set.

Argument: NULL-terminated list of `char *`
Default: NULL
Procs: `create`, `set`
Objects: `Panel_choice_item`
See Also: 7.10, 25.2.1

PANEL_CHOICE_X

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

PANEL_CHOICE_XS

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

PANEL_CHOICE_Y

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

PANEL_CHOICE_YS

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

PANEL_CHOOSE_NONE

Allows scrolling lists or choice items to have no currently selected item. Not applicable if PANEL_CHOOSE_ONE is FALSE.

Argument: `Bool`
Default: TRUE
Procs: `create`, `get`, `set`
Objects: `Panel_choice_item`, `Panel_list_item`
See Also: 7.10.2

PANEL_CHOOSE_ONE

If TRUE, creates an exclusive scrolling list or choice. If FALSE, creates a non-exclusive scrolling list or choice. PANEL_CHOOSE_ONE is used as part of the PANEL_TOGGLE macro. When creating a toggle, it is recommended that you use the macro instead of the attribute.

Argument: `Bool`
Default: TRUE
Procs: `create`, `get`
Objects: `Panel_choice_item`, `Panel_list_item`
See Also: 7.11.3

PANEL_CLIENT_DATA

Specifies an arbitrary value to be attached to a panel or to individual panel items.

Argument: `caddr_t`
Default: None
Procs: `create`, `get`, `set`
Objects: `Panel`, `Panel_item`
See Also: 7.19.1

PANEL_CURRENT_ITEM

Returns the handle of the currently active panel item. An item is considered current when there is a mouse button down event pending on that item (for example, after a SELECT-down over a text scrolling button, but before the corresponding SELECT-up). `NULL` implies no panel item is currently active.

Argument: `Panel_item`
Procs: `get`
Objects: `Panel`

PANEL_DEFAULT_ITEM

Sets the default panel item in the panel or gets the handle of the default panel item. Only buttons are valid as default items. Application programmers use this attribute to indicate the default button in a popup frame. Then, using the `DefaultAction` key, usually RETURN, causes the default button to be activated.

Argument: `Panel_item`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Panel`

PANEL_DEFAULT_VALUE

Identifies the default choice in `PANEL_CHOICE_STACK` items.

Argument: `int` or `unsigned`
Default: First choice
Procs: `create`, `get`, `set`
Objects: `Panel_choice_item`

PANEL_DIRECTION

Identifies the horizontal or vertical orientation of a slider or gauge.

Argument: `Panel_setting`
Valid Values: {`PANEL_HORIZONTAL`, `PANEL_VERTICAL`}
Default: `PANEL_HORIZONTAL`
Procs: `create`, `get`, `set`
Objects: `Panel_slider_item`, `Panel_gauge_item`
See Also: 7.13

PANEL_DISPLAY_LEVEL

Specifies the number of choices to display. Values are one of `PANEL_ALL`, `PANEL_CURRENT`, or `PANEL_NONE`. `PANEL_DISPLAY_LEVEL` is used as part of the `PANEL_CHOICE_STACK` macro; it is recommended that you use the macro instead of the attribute.

Argument: `Panel_setting`
Valid Values: {`PANEL_ALL`, `PANEL_CURRENT`, `PANEL_NONE`}
Default: `PANEL_ALL`
Procs: `create`, `get`
Objects: `Panel_choice_item`
See Also: 7.10.1

XView Attributes

PANEL_DISPLAY_ROWS

Number of rows to display in the multiline text field.

Argument: `int`
Default: 5
Procs: `create,get,set`
Objects: `Panel_multiline_text_item`
See Also: 7.17

PANEL_DROP_BUSY_GLYPH

Busy drop target glyph, as defined by drag and drop specification.

Argument: `Server_image`
Default: Use normal drop target glyph
Procs: `create,get,set`
Objects: `Panel_drop_target_item`
See Also: 7.18.1.2

PANEL_DROP_DND

Drag and drop object (`DRAGDROP`) associated with panel drop target item. The `DRAGDROP` object is used to initiate a drag and drop operation. If no `PANEL_DROP_DND` exists, then the panel Drop Target Item does not support drags, and is termed an "empty" drop target (`PANEL_DROP_FULL = FALSE`).

Argument: `Drag_drop`
Default: `NULL`
Procs: `create,get,set`
Objects: `Panel_drop_target_item`
See Also: 7.18.1.3

PANEL_DROP_FULL

If `TRUE`, then the drop target item has draggable data set on the `PANEL_DROP_DND` object's selection items. The normal glyph is displayed. Setting `PANEL_DROP_FULL` to `TRUE` when no `PANEL_DROP_DND` is defined is not valid; `PANEL_DROP_FULL` will remain `FALSE`. If `FALSE`, then no draggable data is available (i.e., the drop target is empty). No glyph is displayed.

Argument: `Bool`
Default: `FALSE`
Procs: `create,get,set`
Objects: `Panel_drop_target_item`
See Also: 7.18.1.3

PANEL_DROP_GLYPH

Normal drop target glyph, as defined by drag and drop package.

Argument: `Server_image`
Default: None
Procs: `create,get,set`
Objects: `Panel_drop_target_item`
See Also: 7.18.1.2

PANEL_DROP_HEIGHT

Dimensions of drag target box, in pixels, excluding border and margin.

Argument: `int`
Default: 16
Procs: `create,get,set`
Objects: `Panel_drop_target_item`

PANEL_DROP_SEL_REQ

Selection requestor object (`SELECTION_REQUESTOR`) associated with the panel drop target item. The `Selection_requestor` object is used to receive a drop.

Argument: `Selection_requestor`
Procs: `get`
Objects: `Panel_drop_target_item`
See Also: 7.18.1.4

PANEL_DROP_SITE_DEFAULT

Specifies whether or not the panel drop target's drop site is the default drop site. Only one panel drop target in a frame may be the default drop target. See `DROP_SITE_DEFAULT` attribute for more information.

Argument: `Bool`
Default: `FALSE`
Procs: `create,get,set`
Objects: `Panel_drop_target_item`

PANEL_DROP_WIDTH

Dimensions of drag target box, in pixels, excluding border and margin.

Argument: `int`
Default: 12
Procs: `create,get,set`
Objects: `Panel_drop_target_item`

PANEL_EVENT_PROC

Event handler for panel items.

Argument: `void (`**panel_event_proc*`)()`
Default: Panel_item specific.
Procs: `create,get,set`
Objects: `Panel_item`
Callback:

```
void
panel_event_proc(item, event)
        Panel_item    item
        Event        *event
```

Client-specified event-handling procedure for handling events on panel items. This procedure is installed using `PANEL_EVENT_PROC`.

See Also: 7.19.7

PANEL_EXTRA_PAINT_HEIGHT

Defines the increment by which the panel grows in the `y` direction. It is used when `window_fit_height()` is called.

Argument: `int`
Default: 1 pixel
Procs: `create`, `get`, `set`
Objects: `Panel`
See Also: 7.5

XView Attributes

PANEL_EXTRA_PAINT_WIDTH

Defines the increment by which the panel grows in the `x` direction. It is used when `window_fit_width()` is called.

Argument: `int`
Default: 1 pixel
Procs: `create`, `get`, `set`
Objects: `Panel`
See Also: 7.5

PANEL_FEEDBACK

Specifies feedback to give when an item is selected. If `PANEL_DISPLAY_LEVEL` is `PANEL_ CURRENT`, the default value is `PANEL_NONE`; otherwise, it is `PANEL_MARKED`. `PANEL_FEED_BACK` is used as part of the `PANEL_CHECK_BOX` macro. It is recommended that you use the macro instead of the attribute.

Argument: `Panel_setting`
Default: See description
Procs: `create`, `get`, `set`
Objects: `Panel_choice_item`

PANEL_FIRST_ITEM

Gets the handle of the first item on a panel. The `PANEL_EACH_ITEM` macro (see Section 3, *Procedures and Macros*) can be used to iterate over each item in a panel, starting with this first item.

Argument: `Panel_item`
Procs: `get`
Objects: `Panel`
See Also: 7.7

PANEL_FIRST_PAINT_WINDOW

Returns a pointer to the first `Panel_paint_window` struct, which defines the paint window and view window associated with the first view in the scrollable panel.

Argument: `Panel_paint_window *`
Procs: `get`
Objects: `Scrollable_panel`

PANEL_FOCUS_PW

The paint window which currently has or last had the input focus. The `has_input_focus` `Panel_status` flag indicates whether or not the scrollable panel currently has the input focus. Initially, the focus paint window is set to the first paint window created in the scrollable panel.
Warning: This attribute should *only* be used from within a panel item extension.

Argument: `Xv_Window`
Procs: `get`
Objects: `Scrollable_panel`

PANEL_GAUGE_WIDTH

Specifies the length of the panel gauge "thermometer" bar, in pixels, regardless of its horizontal or vertical orientation.

Argument: `int`
Default: `100`
Procs: `create`, `get`, `set`
Objects: `Panel_gauge_item`
See Also: 7.14

PANEL_GINFO

The OLGX Graphics Information structure for the specified panel or panel item. This attribute is only of interest to those panel item extensions that are calling OLGX functions or using OLGX macros. **Warning**: This attribute should *only* be used from within a panel item extension.

Argument: `Graphics_info *`
Procs: `get`
Objects: `Panel,Panel_item`

PANEL_INACTIVE

If `TRUE`, panel item cannot be selected. Inactive items are displayed with gray-out pattern.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `Panel_item`
See Also: 7.9.1.1

PANEL_ITEM_CLASS

Gets the panel item *type* of the panel item specified.

Argument: `Panel_item_type`
Procs: `get`
Objects: `Panel_item`

PANEL_ITEM_COLOR

Specifies the colormap index to use for the item. A value of –1 implies the foreground color for the panel window.

Argument: `int`
Default: `-1`
Procs: `create`, `get`, `set`
Objects: `Panel_item`
See Also: 21.5.1

PANEL_ITEM_CREATED

The `Panel_item` has finished processing the `XV_END_CREATE` phase of the generic panel item package (i.e. `xv_panel_item_pkg`). All `Panel_items` are subclassed from `xv_panel_item_pkg`.
Warning: This attribute should *only* be used from within a panel item extension.

Argument: `Bool`
Procs: `get`
Objects: `Panel_item`

PANEL_ITEM_DEAF

The item does not want any events. This is used for items that contain one or more windows that are located within the item's rectangle. These windows are interposed on in order to process their events and maintain the panel's internal variables. An example of this is `PANEL_MULTILINE_TEXT`.
Warning: This attribute should *only* be used from within a panel item extension.

Argument: `Bool`
Procs: `get,set`
Objects: `Panel_item`

XView Attributes

PANEL_ITEM_LABEL_RECT

The rectangle describing the label portion of an item.
Warning: This attribute should *only* be used from within a panel item extension.

Argument: Rect *
Procs: get,set
Objects: Panel_item

PANEL_ITEM_MENU

Specifies the menu associated with the panel item.

Argument: Menu
Default: NULL
Procs: create, get, set
Objects: Panel_item

PANEL_ITEM_NTH_WINDOW

Returns the nth embedded window that receives events in the specified panel item. The first embedded window has index 0. This attribute functions exactly like OPENWIN_NTH_VIEW. This attribute is an "advanced" attribute; the general application programmer will not need to use this attribute.

Return Type: Xv_Window
Argument: int
Procs: get
Objects: Panel_item
Usage:

```
textsw_view = xv_get(mltxt_item,
                  PANEL_ITEM_NTH_WINDOW,0, NULL);
```

See Also: OPENWIN_NTH_VIEW

PANEL_ITEM_NWINDOWS

Returns the number of embedded windows in the specified panel item that receives events. Zero indicates that there are no embedded windows. This attribute functions exactly like OPENWIN_NVIEWS. This attribute is an "advanced" attribute; the general application programmer will not need to use this attribute.

Return Type: int
Procs: get
Objects: Panel_item
See Also: OPENWIN_NVIEWS

PANEL_ITEM_OWNER

Allows the client to obtain the parent object given the child. In essence, if PANEL_ITEM_OWNER ever returns a value (non-NULL), then the client should assume this is the proper handle.

Values retrieved or set on the child handle are not guaranteed to be forward compatible if those values are not handled through the parent object's API.

Argument: XV_object
Procs: get
Objects: Panel_item

PANEL_ITEM_RECT

Gets the rectangle surrounding the panel item.

Argument: `Rect *`
Procs: `get`
Objects: `Panel_item`

PANEL_ITEM_VALUE_RECT

The rectangle describing the value portion of an item.
Warning: This attribute should *only* be used from within a panel item extension.

Argument: `Rect *`
Procs: `get,set`
Objects: `Panel_item`

PANEL_ITEM_WANTS_ADJUST

The panel item wants `ACTION_ADJUST` events.
Warning: This attribute should *only* be used from within a panel item extension.

Argument: `Bool`
Procs: `get,set`
Objects: `Panel_item`

PANEL_ITEM_WANTS_ISO

This flag indicates that the panel item wants all ISO characters. Unmodified ASCII characters are not to be interpreted as mouseless keyboard commands. Currently, `PANEL_MULTILINE_TEXT` and `PANEL_TEXT` items have this flag set.
Warning: This attribute should *only* be used from within a panel item extension.

Argument: `Bool`
Default: `FALSE`
Procs: `get,set`
Objects: `Panel_item`

PANEL_ITEM_X

Specifies the `x` position (in pixels) where the last panel item was created. If no item was created, then it specifies the `x` position where the first item will be created.

Argument: `int`
Procs: `get`
Objects: `Panel`
See Also: 7.4.2

PANEL_ITEM_X_GAP

When set on a panel, the horizontal space, in pixels, between the last panel item created and the next panel item; when set on a panel item it is the horizontal space, in pixels, between the last panel item created and this panel item. This attribute allows applications to more easily use relative item positioning instead of absolute positioning. Relative item positioning is the recommended method of positioning panel items, since the size of items may change between invocations (e.g., different scales) or versions of the toolkit.

Argument: `int`
Default: `10`, for Panel; the panel's `PANEL_ITEM_X_GAP`, for panel item
Procs: `create,get,set`, for panel; `create,get`, for panel item
Objects: `Panel`, `Panel_item`
See Also: 7.3.1

XView Attributes

PANEL_ITEM_X_POSITION

The default x coordinate of the panel item.
Warning: This attribute should *only* be used from within a panel item extension.

Argument: `int`
Procs: `get,set`
Objects: `Panel`

PANEL_ITEM_Y

Specifies the `y` position (in pixels) where the last panel item was created. If no item was created, then it specifies the `y` position where the first item will be created.

Argument: `int`
Default: None
Procs: `get`
Objects: `Panel`
See Also: 7.4.2

PANEL_ITEM_Y_GAP

When set on a panel, this is the vertical space, in pixels, between the last panel item created and the next panel item when set on a panel item, `PANEL_ITEM_Y_GAP` is the vertical space, in pixels, between the last panel item created and this panel item. This attribute allows applications to more easily use relative item positioning instead of absolute positioning. Relative item positioning is the recommended method of positioning panel items, since the size of items may change between invocations (e.g., different scales) or versions of the toolkit.

Argument: `int`
Default: 13 for panel, the panel's `PANEL_ITEM_Y_GAP` for panel item
Procs: `create,get,set` for panel; `create,get` for panel item
Objects: `Panel, Panel_item`
See Also: 7.3.1

PANEL_ITEM_Y_POSITION

The default y coordinate of the panel item.
Warning: This attribute should *only* be used from within a panel item extension.

Argument: `int`
Procs: `get,set`
Objects: `Panel`

PANEL_JUMP_DELTA

Specifies the number of client units to adjust the value when a `JumpLeft` or `JumpRight` keyboard command is issued for a horizontal slider. Also specifies the number of client units to adjust the value when a `JumpUp` or `JumpDown` keyboard command is issued for a vertical slider or numeric text item. Used in conjunction with the OPEN LOOK Mouseless Model.

Argument: `int`
Default: 10
Procs: `create,get,set`
Objects: `Panel_slider_item,Panel_numeric_text_item`

PANEL_LABEL_BOLD

If TRUE, PANEL_MESSAGE_ITEM's label is rendered in bold.

Argument: Bool
Default: FALSE
Procs: create, get, set
Objects: Panel_message_item
See Also: 7.12

PANEL_LABEL_FONT

Defines the font to use in the label portion of a panel item.
Warning: Use of this attribute may cause your application to violate the OPEN LOOK Graphical User Interface Functional Specification.

Argument: Xv_Font
Default: Panel font
Procs: create,get,set
Objects: Panel_item

PANEL_LABEL_IMAGE

Specifies the image for an item's label.

Argument: Server_image
Default: NULL
Procs: create, get, set
Objects: Panel_item
See Also: 15.4.1

PANEL_LABEL_STRING

Specifies the string for the item's label. The string is copied when this attribute is set. By using \n, newlines can be placed into the strings to generate multi-line labels.

Argument: char *
Default: ""
Procs: create, get, set
Objects: Panel_item
Usage:

```
xv_create(panel, PANEL_TEXT,
        PANEL_LABEL_STRING, "Multi\nLine\nLabel:",
        NULL);
```

See Also: 7.9, 7.3.2

PANEL_LABEL_WIDTH

Specifies the width of the panel item label in pixels. For buttons, PANEL_LABEL_WIDTH does not include endcaps and menu marks (if any are present). PANEL_LABEL_WIDTH has no effect on a panel button until the panel button's PANEL_LABEL_STRING or PANEL_LABEL_IMAGE is set.

Argument: int
Default: Width of text string or server image
Procs: create, get, set
Objects: Panel_item
See Also: 7.9.3

PANEL_LABEL_X

Specifies the x coordinate of the *label* portion of panel items that have *values* associated with them (e.g., text items). Intended to be used when PANEL_VALUE_X is specified.

Argument: int (in pixels)
Default: PANEL_ITEM_X from parent panel
Procs: create, get, set
Objects: Panel_item
See Also: PANEL_VALUE_X

PANEL_LABEL_Y

Specifies the y coordinate of the *label* portion of panel items that have *values* associated with them (e.g., text items). Intended to be used only if PANEL_VALUE_Y is specified.

Argument: int (in pixels)
Default: PANEL_ITEM_Y from parent panel
Procs: create, get, set
Objects: Panel_item
See Also: PANEL_VALUE_Y

PANEL_LAYOUT

In a PANEL create call, this attribute controls the layout of panel items. When used in a create panel item call, this attribute controls the direction in which the item's components are laid out.

Argument: Panel_setting
Default: PANEL_HORIZONTAL
Procs: create, get, set
Objects: Panel, Panel_item
See Also: 7.3.1

PANEL_LINE_BREAK_ACTION

At the end of each line, wrap to the next line at either the character (PANEL_WRAP_AT_CHAR) or word (PANEL_WRAP_AT_WORD) level.

Argument: Panel_setting
Valid values: {PANEL_WRAP_AT_CHAR, PANEL_WRAP_AT_WORD}
Default: value of "text.lineBreak" default, or PANEL_WRAP_AT_WORD
Procs: create,get,set
Objects: Panel_multiline_text_item
See Also: 7.17

PANEL_LIST_CLIENT_DATA

Sets or gets up to 32 bits of user-entered data (argument 2) from a row number (argument 1) on the list.

Argument 1: int
Argument 2: Xv_opaque
Default: NULL
Procs: create, get, set
Objects: Panel_list_item
See Also: 7.11.4.1

PANEL_LIST_CLIENT_DATAS

Works much like its companion attribute, PANEL_LIST_CLIENT_DATA, except that it takes a NULL-terminated value list of client data as its value. Position in the value list determines the row to which the data will be associated.

Argument: NULL-terminated list of Xv_opaque
Default: NULL
Procs: create, set
Objects: Panel_list_item
See Also: 7.11.4.1

PANEL_LIST_DELETE

Deletes a row from the scrolling list. The list is adjusted automatically after the deletion.

Argument: int (row number)
Default: None
Procs: create, set
Objects: Panel_list_item
See Also: 7.11.2

PANEL_LIST_DELETE_INACTIVE_ROWS

Deletes all inactive rows from the list. Similar to the PANEL_LIST_DELETE_SELECTED_ROWS attribute.

Argument: No argument
Procs: set
Objects: Panel_list_item
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

PANEL_LIST_DELETE_ROWS

Deletes the specified number of rows, starting at the specified row. The first argument is the row number, and the second argument is the number of rows to delete.

Argument 1: int
Argument 2: int
Procs: create,set
Objects: Panel_list_item
See Also: 7.11.2

PANEL_LIST_DELETE_SELECTED_ROWS

Deletes all selected rows. Not valid in edit mode.

Argument: No value
Procs: set
Objects: Panel_list_item
See Also: 7.11.2

PANEL_LIST_DISPLAY_ROWS

Sets the number of rows in a list that will be displayed. This may not be set to less than three for a FILE_LIST.

Argument: int
Default: The number of items in the xv_create() call, up to a maximum of 5.
For a FILE_LIST the default is 10.
Procs: create, get, set
Objects: Panel_list_item, File_list
See Also: 7.11.1

XView Attributes

PANEL_LIST_DO_DBL_CLICK

Tells PANEL_LIST to interpret two select events that occur within the timeout value as a double-click instead of as a second select or a deselect (depending on the current mode of the list). The timeout value is specified with OpenWindows.MulticlickTimeout. If true, the PANEL_LIST will deliver a new op called PANEL_LIST_OP_DBL_CLICK instead of the normal PANEL_LIST_OP_SELECT or PANEL_LIST_OP_DESELECT.

Argument:	Bool
Default:	FALSE
Procs:	create, get, set
Objects:	Panel_list_item
See Also:	Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

PANEL_LIST_EXTENSION_DATA

Same as PANEL_LIST_CLIENT_DATA, except that it is reserved for package implementors. This is used by the FILE_LIST package.

PANEL_LIST_EXTENSION_DATAS

Same as PANEL_LIST_CLIENT_DATAS, except that it is reserved for package implementors. This is used by the FILE_LIST package.

PANEL_LIST_FIRST_SELECTED

Returns the row number of the first selected row. If no row is selected, it returns -1.

Return Type:	int
Argument:	No argument
Procs:	get
Objects:	Panel_list_item
See Also:	7.11.3

PANEL_LIST_FONT

Set the specified row (argument 1) to the specified font (argument 2). Note that the font specification using PANEL_LIST_FONT should follow the creation of the row (using PANEL_LIST_STRINGS).

Argument 1:	int (row number)
Argument 2:	Xv_Font
Default:	Use panel font
Procs:	create,get,set
Objects:	Panel_list_item
See Also:	7.11.1

PANEL_LIST_FONTS

Set the first *n* rows to the specified fonts. Note that fonts specified using PANEL_LIST_FONT should follow the creation of the rows (using PANEL_LIST_STRINGS).

Argument:	NULL-terminated list of Xv_Font
Default:	Use panel font
Procs:	create,set
Objects:	Panel_list_item
See Also:	7.11.1

PANEL_LIST_GLYPH

Takes a row number (argument 1) and `Server_image` (argument 2) to let you assign a glyph or icon to a row. The height of the glyph may not exceed the height of the scrolling list row. Also see `PANEL_LIST_ROW_HEIGHT`.

Return Type: `Server_image`
Argument 1: `int`
Argument 2: `Server_image`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Panel_list_item`
See Also: 7.11.1

PANEL_LIST_GLYPHS

Works the same as its companion attribute, `PANEL_LIST_GLYPH`, except that it takes a `NULL`-terminated value list as its value. The height of the glyph may not exceed the height of the scrolling list row. Also see `PANEL_LIST_ROW_HEIGHT`.

Argument: `NULL`-terminated list of `Server_image`
Default: `NULL`
Procs: `create`, `set`
Objects: `Panel_list_item`
See Also: 7.11.1

PANEL_LIST_INACTIVE

Grays out a row in a `PANEL_LIST`. Note that a row that is inactive cannot be selected at the same time. Also, mouseless navigation is not effected by the inactive state of the individual rows.

Argument 1: `int`
Argument 2: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `Panel_list_item`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual.*

PANEL_LIST_INSERT

Inserts a list item at a specified row number. This attribute allocates space, attaches a row number to the list, and inserts an empty string. Clients must set `PANEL_LIST_GLYPH` and/or `PANEL_LIST_STRING` at this row number to set the glyph item and/or string.

Argument: `int`
Procs: `create`, `set`
Objects: `Panel_list_item`
See Also: 7.11.2

PANEL_LIST_INSERT_DUPLICATE

Allow (`TRUE`) or disallow (`FALSE`) duplicate strings to be inserted into the scrolling list.

Argument: `Bool`
Default: `TRUE`
Procs: `create,get,set`
Objects: `Panel_list_item`
See Also: 7.11.2

XView Attributes

PANEL_LIST_INSERT_GLYPHS

Insert the specified glyphs into the scrolling list before the specified row. The first argument is a row number, and the second argument is a pointer to a NULL-terminated array of server images.

Argument 1: int
Argument 2: NULL-terminated list of Server_image
Procs: create,set
Objects: Panel_list_item
See Also: 7.11.2

PANEL_LIST_INSERT_STRINGS

Insert the specified strings into the scrolling list before the specified row. The first argument is a row number, and the second argument is a pointer to a NULL-terminated array of strings. The strings are copied when this attribute is set.

Argument 1: int
Argument 2: NULL-terminated list of char *
Procs: create,set
Objects: Panel_list_item
See Also: 7.11.2

PANEL_LIST_MASK_GLYPH

Tells the PANEL_LIST to use the given Server_image as a clip mask for the corresponding PANEL_LIST_GLYPH. The Server_image supplied must be of depth 1.

Argument: int
Argument: Server_image
Default: XV_NULL
Procs: create, get, set
Objects: Panel_list
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

PANEL_LIST_MASK_GLYPHS

Like PANEL_LIST_MASK_GLYPH, but takes a NULL terminated list of glyphs rather than a row and a single handle.

Argument: NULL-terminated list of Server_image
Procs: create, set
Objects: Panel_list
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

PANEL_LIST_MODE

Sets or gets the mode of the scrolling list. Setting the mode to PANEL_LIST_READ , when in edit mode is equivalent to selecting the “End Editing” menu item. Setting the mode to PANEL_LIST_EDIT when in read mode is equivalent to selecting the “Edit List” menu item.

Argument: Panel_list_mode
Valid Values: {PANEL_LIST_READ, PANEL_LIST_EDIT}
Default: PANEL_LIST_READ
Procs: get,set
Objects: Panel_list_item
See Also: 7.11.5

PANEL_LIST_NEXT_SELECTED

Returns the row number of the first selected row following the row specified. If no row after the row specified is selected, returns -1.

Argument: `int`
Procs: `get`
Objects: `Panel_list_item`
See Also: 7.11.3

PANEL_LIST_NROWS

Gets the total number of rows in the scrolling list.

Return Type: `int`
Procs: `get`
Objects: `Panel_list_item`

PANEL_LIST_ROW_HEIGHT

Specifies the height of each row in the scrolling list.

Argument: `int`
Default: Height of panel's font
Procs: `create`, `get`
Objects: `Panel_list_item`
See Also: 7.11.1

PANEL_LIST_ROW_VALUES

Improved performance method of getting/setting row values into the `PANEL_LIST`. Takes the row number, a pointer to a `Panel_list_row_values` array, and a count of how many rows in the array. `Panel_list_row_values` defined shown below.

Argument 1: `int`
Argument 2: `struct *`
Argument 3: `int`
Procs: `create`, `get`, `set`
Objects: `Panel_list`

```
typedef struct {
char * string;
Server_image glyph;
Server_image mask_glyph;
Xv_font font;
Xv_opaque client_data;
Xv_opaque extension_data;
unsigned inactive : 1;
unsigned selected : 1;
} Panel_list_row_values;
```

On `xv_get`, the arguments remain the same, the `struct` passed in gets filled in by the `PANEL_LIST` package. The return value is the number of rows that were successfully filled in.

PANEL_LIST_SCROLLBAR

Returns the scrollbar attached to a PANEL_LIST panel item.

Return Type: Scrollbar
Procs: get
Objects: Panel_list_item

PANEL_LIST_SELECT

Takes two values: a row number (argument 1) and a Bool (argument 2) that lets you select (TRUE) or deselect (FALSE) the specified row. If the scrolling list is not hidden (XV_SHOW is TRUE) then the specified row will be made visible, which may entail scrolling the scrolling list. To disable this scrolling, set XV_SHOW to FALSE before setting PANEL_LIST_SELECT and then reset XV_SHOW to TRUE afterwards.

Argument 1: int
Argument 2: Bool
Default: FALSE
Procs: set, create
Objects: Panel_list_item
See Also: 7.11.3

PANEL_LIST_SELECTED

Returns whether the specified row number is selected (TRUE) or not (FALSE).

Argument: int
Return Type: Bool
Procs: get
Objects: Panel_list_item

PANEL_LIST_SORT

Sort the list in forward (PANEL_FORWARD) or reverse (PANEL_REVERSE) alphabetical order. The FILE_LIST package consumes this attribute. Use FILE_LIST_COMPARE_FUNC in its place.

Argument: Panel_setting
Valid values: PANEL_FORWARD or PANEL_REVERSE
Procs: set
Objects: Panel_list_item

PANEL_LIST_STRING

Specifies the string (argument 2) of a specified row (argument 1). xv_get returns the pointer to the character string assigned to the row. The string is copied when this attribute is set.

Return Type: char *
Argument 1: int (row number)
Argument 2: char *
Default: Empty string
Procs: create, get, set
Objects: Panel_list_item
See Also: 7.11.1

PANEL_LIST_STRINGS

Works the same as its companion attribute, `PANEL_LIST_STRING`, except that it takes a `NULL`-terminated list of strings as its value. The strings are copied when this attribute is set.

Argument: `NULL`-terminated list of `char *`
Default: Empty string
Procs: `create`, `set`
Objects: `Panel_list_item`
See Also: 7.11.1

PANEL_LIST_TITLE

The title of the scrolling list. `PANEL_LIST_TITLE` makes a copy of the string. When the scrolling list is removed, the package will free the title string.

Argument: `char *`
Default: No title
Procs: `create,get,set`
Objects: `Panel_list_item`
See Also: 7.11.1

PANEL_LIST_WIDTH

Specifies the width of the scrolling list: -1 extends the scrolling list to the right edge of the panel. A value of 0 sets the width slightly wider than the widest row, and other values specify the width of the scrolling list in pixels. To get the width of a `PANEL_LIST` item's rectangle, use `XV_WIDTH`. To get the width of the label rectangle, use `PANEL_LABEL_WIDTH`.

Argument: `int`
Default: 0
Procs: `create,get,set`
Objects: `Panel_list_item`
See Also: 7.11.1

PANEL_MASK_CHAR

Specifies the character used to mask type-in characters. Use the space character for no character echo (caret does not advance). Use the `NULL` character to disable masking.

Argument: `char`
Default: `NULL`
Procs: `create`,`set`,`get`
Objects: `Panel_text_item`, `Panel_numeric_text_item`
See Also: 7.15

PANEL_MAX_TICK_STRING

String which appears underneath the maximum tick mark on horizontal sliders or gauges, or to the right of the maximum tick mark on vertical sliders or gauges. If `PANEL_TICKS` is 0, `PANEL_MAX_TICK_STRING` is ignored. The width of the slider or gauge may be adjusted to insure that there is enough space to accommodate both the minimum and maximum tick strings. The string is copied when this attribute is set.

Argument: `char *`
Default: No maximum tick string
Procs: `create,get,set`
Objects: `Panel_slider_item, Panel_gauge_item`
See Also: 7.13

XView Attributes

PANEL_MAX_VALUE

Specifies the maximum value of the slider, gauge, or numeric text item.

Argument:	`int`
Default:	`100`
Procs:	`create`, `get`, `set`
Objects:	`Panel_numeric_text_item`, `Panel_gauge_item`, `Panel_slider_item`
See Also:	7.13

PANEL_MAX_VALUE_STRING

Maximum value string for the slider. On horizontal sliders, the maximum value string appears to the right of the maximum end box. On vertical sliders, the maximum value string appears above the maximum end box. The string is copied when this attribute is set.

Argument:	`char *`
Default:	No maximum value string
Procs:	`create,get,set`
Objects:	`Panel_slider_item`
See Also:	7.13

PANEL_MIN_TICK_STRING

String which appears underneath the minimum tick mark on horizontal sliders or gauges, or to the right of the minimum tick mark on vertical sliders or gauges. If `PANEL_TICKS` is 0 `PANEL_MIN_TICK_STRING` is ignored. The width of the slider or gauge may be adjusted to insure that there is enough space to accommodate both the minimum and maximum tick strings. The string is copied when this attribute is set.

Argument:	`char *`
Default:	No minimum tick string
Procs:	`create, get, set`
Objects:	`Panel_slider_item, Panel_gauge_item`
See Also:	7.13

PANEL_MIN_VALUE

Specifies the minimum value of the slider, gauge, or numeric text item.

Argument:	`int`
Default:	`0`
Procs:	`create`, `get`, `set`
Objects:	`Panel_slider_item`, `Panel_numeric_text_item, Panel_gauge_item`
See Also:	7.13

PANEL_MIN_VALUE_STRING

Minimum value string for the slider. On horizontal sliders, the minimum value string appears to the left of the minimum end box. On vertical sliders, the minimum value string appears below the minimum end box. The string is copied when this attribute is set.

Argument:	`char *`
Default:	No minimum value string
Procs:	`create, get, set`
Objects:	`Panel_slider_item`
See Also:	7.13

PANEL_NCHOICES

Returns the number of choices available in a choice or toggle item.

Return Type: `int`
Procs: `get`
Objects: `Panel_choice_item`
See Also: 7.10

PANEL_NEXT_COL

This attribute is used when the panel layout is `PANEL_VERTICAL`. It specifies that the item is to start a new column and specifies the amount of white space (in pixels) between the last column and the next (new) column. If you specify –1, the default gap (`PANEL_ITEM_X_GAP`) is used.

Argument: `Panel_item`
Default: None
Procs: `create`
Objects: `Panel_item`
See Also: 7.3.1

PANEL_NEXT_ITEM

Gets the handle of the next item on a panel.

Argument: `Panel_item`
Procs: `get`
Objects: `Panel_item`
Usage:

```
next_item = xv_get(this_item, PANEL_NEXT_ITEM);
```

See Also: 7.7

PANEL_NEXT_ROW

This attribute is used when the panel layout is `PANEL_HORIZONTAL`. It specifies that the item is to start a new row and specifies the amount of white space (in pixels) between the last row and the next (new) row. If you specify –1, the default gap (`PANEL_ITEM_Y_GAP`) is used.

Argument: `Panel_item`
Default: None
Procs: `create`
Objects: `Panel_item`
See Also: 7.3.1

PANEL_NO_REDISPLAY_ITEM

This flag is useful if you intend to call `xv_super_set_avlist()` at the beginning of the set routine. Setting this attribute to `TRUE` prevents the parent panel from redisplaying while the attributes are being set. This attribute should be reset to `FALSE` after the call to `xv_super_set_avlist()`. **Warning**: This attribute should *only* be used from within a panel item extension.

Argument: `Bool`
Procs: `get,set`
Objects: `Panel`
See Also: 25.11.6

PANEL_NOTIFY_LEVEL

Specifies when to call the notify function. The valid values are PANEL_ALL, PANEL_NONE, PANEL_SPECIFIED, and PANEL_NON_PRINTABLE. For sliders, any setting other than PANEL_ALL results in notification only on SELECT-up.

Argument: Panel_setting
Valid Values: {PANEL_ALL, PANEL_NONE , PANEL_SPECIFIED, PANEL_NON_PRINTABLE}
Default: PANEL_SPECIFIED
Procs: create, get, set
Objects: Panel_numeric_text_item, Panel_multiline_text_item, Panel_slider_item, Panel_text_item
See Also: 7.13.2

PANEL_NOTIFY_PROC

Procedure to call when a panel item is activated.

Argument: Varies—see callbacks below
Default: NULL
Procs: create, get, set
Objects: Panel_item

Callback: (For Button Items)

```
void
notify_proc(item, event)
        Panel_item   item;
        Event       *event;
```

Client-specified callback routine invoked when the user activates the button (accepts preview).

Callback: (For Message Items)

```
void
notify_proc(item, event)
        Panel_item   item;
        Event       *event;
```

Client-specified callback routine invoked when the user activates the message (accepts preview).

Callback: (For Text Items, Numeric Text Items, and Multiline Text Items)

```
Panel_setting
notify_proc(item, event)
        Panel_item   item;
        Event       *event;
```

Client-specified callback routine invoked when the user activates the text (accepts preview). For text, numeric text, and multiline text items, the return value type is Panel_setting and is one of PANEL_INSERT, PANEL_NEXT, PANEL_NONE, or PANEL_PREVIOUS.

Callback: (For Exclusive Choice Items and Slider Items)

```
void
notify_proc(item, value, event)
        Panel_item   item;
        int          value;
        Event       *event;
```

Client-specified callback routine invoked when the user activates the exclusive choice or slider (accepts preview).

Callback: (For Non-exclusive Choice Items, toggles)

```
void
notify_proc(item, value, event)
        Panel_item      item;
        unsigned int    value;
        Event          *event;
```

Client-specified callback routine invoked when the user activates the non-exclusive choice (accepts preview). The `value` parameter is of type `unsigned int` because it represents a *mask* of choices that are selected. For example, if the first and third items are selected, then the first and third bits in the `value` parameter are on—this value happens to be five.

Callback: (For List Items)

```
int
notify_proc(item, string, client_data, op, event, row)
        Panel_item        item;
        char             *string;
        Xv_opaque         client_data;
        Panel_list_op     op;
        Event            *event;
        int               row;
```

Client-specified callback routine invoked when the user activates the list item (accepts preview).
`string` is the string associated with the `row`.
`client_data` is the client data associated with the `row`.
`op` indicates a select, validate, or delete operation. Of these, only `PANEL_LIST_OP_VALIDATE` requires a return value.
`row` indicates the row in the scrolling list. Returns `XV_OK` to validate a change or `XV_ERROR` to invalidate a change.

Callback: (For Drop Target Item)

```
int
notify_proc(item, event)
        Panel_item     item;
        Event         *event;
```

A return value of `XV_OK` indicates that XView should call `dnd_done()`. A return value of `XV_ERROR` indicates that XView should not call `dnd_done()`.

XView Attributes

Callback: (For Path Name Items)

```
Panel_setting
notify_proc(item, event, stat_buf)
  Path_name     item;
  Event *       event;
  struct stat * stat_buf;
```

Same as for PANEL_TEXT, except that the PATH_NAME package passes in the pointer to a stat structure as well as the item and event handles.

Callback: (For a File_list)

```
int
notify_proc( item, dir, file, client_data,
                          op, event, row )
   File_list  item;
   char      *dir;
   char      *file;
   Xv_opaque  client_data;
   Panel_list_op op;
   Event     *event;
   int        row;
```

Note: the client_data field is added either by the client explicitly setting PANEL_LIST_CLIENT_DATA(S) or by returning the value from the FILE_LIST_FILTER_FUNC.

The op PANEL_LIST_OP_DBL_CLICK, supports the PANEL_LIST package for applications that want to detect a double-click condition. This behavior is selected using the attribute PANEL_LIST_DO_DBL_CLICK which defaults to FALSE for the PANEL_LIST package (for compatibility reasons) but defaults to TRUE for the FILE_LIST package.

The FILE_LIST package intercepts the PANEL_LIST_OP_DBL_CLICK condition in cases where the user double-clicks on a directory name. The client only sees the double-click op if it occurred on a file that he may wish to open or save to. The client can still catch the directory change by installing a function via the FILE_LIST_CHANGE_DIR_FUNC attribute.

See Also: 7.10.5

PANEL_NOTIFY_STATUS

If the panel item is part of an unpinned command frame, then the window is dismissed if PANEL_NOTIFY_STATUS is XV_OK. If PANEL_NOTIFY_STATUS is set to XV_ERROR, then the window is not dismissed. You only xv_get() or xv_set() this attribute from within a notify procedure. When a notify procedure exits, for an unpinned command frame, XView internally uses the value of this attribute to determine whether or not the command frame is dismissed. XView sets the value of this attribute to XV_OK before calling the notify procedure. For the PATH_NAME package, see the following PANEL_NOTIFY_STATUS explanation.

Argument: int
Valid Values: {XV_OK, XV_ERROR}
Default: XV_OK
Procs: get, set
Objects: Panel_item
See Also: 7.9.1

PANEL_NOTIFY_STATUS

This attribute is overloaded by the Path Name package to represent whether or not the PANEL_NOTIFY_PROC succeeded in validating the the last directory it was called with. The return value from the notify procedure is of type Panel_setting and does not exactly match the same condition. Expected values are XV_OK or XV_ERROR in the case of an invalid path name. For a PATH_ITEM, see the preceding PANEL_NOTIFY_STATUS explanation.

Argument: int
Default: XV_OK
Procs: create, get, set

PANEL_NOTIFY_STRING

String of characters that triggers notification when one of the characters is typed in a text item. Applies only when PANEL_NOTIFY_LEVEL is PANEL_SPECIFIED. The string is copied when this attribute is set.

Argument: char *
Default: \n \r \t (i.e., newline, carriage return, and tab)
Procs: create, get, set
Objects: Panel_numeric_text_item, Panel_text_item, Panel_multiline_text_item
See Also: 7.15.3

PANEL_OPS_VECTOR

The panel operations vector, which is the method by which the panel package dispatches events to panel items. For more information, see Section 25.11, “The Wizzy Package” in the *XView Programming Manual*.

Argument: Panel_ops *
Procs: get,set
Objects: Panel_item

PANEL_PAINT

Controls the panel item’s painting behavior in xv_set calls. The specified value applies only to the xv_set call. As the xv_set call exits, it restores PANEL_PAINT to its previous value. Possible values are PANEL_CLEAR, PANEL_NO_CLEAR, or PANEL_NONE. PANEL_PAINT operates on Panel and Panel_item objects. The default for Panel item objects is the value of PANEL_PAINT for the Panel to which the Panel item is attached. This attribute is used strictly for performance enhancements and, normally, is not required.

Argument: Panel_setting
Default: PANEL_CLEAR for panel objects, the panel’s PANEL_PAINT for panel item objects
Procs: create, set
Objects: Panel,Panel_item
See Also: 7.19.3

PANEL_PRIMARY_FOCUS_ITEM

The current or last panel item that is or was a primary (First-Class) focus client.
Warning: This attribute should *only* be used from within a panel item extension.

Argument: Panel_item
Procs: get,set
Objects: Panel
Usage:

```
if (xv_get(panel_public, PANEL_PRIMARY_FOCUS_ITEM) == item_public)
      xv_set(panel_public, PANEL_PRIMARY_FOCUS_ITEM, NULL, NULL);
```

XView Attributes

This example should be executed when an item's remove procedure is called, if the item is a primary focus client.

PANEL_READ_ONLY

If TRUE, editing is disabled; if FALSE, editing is enabled. Note that xv_set does not apply to scrolling lists. PANEL_READ_ONLY mode defaults to TRUE for the FILE_LIST package as opposed to FALSE for the PANEL_LIST package.

Argument: Bool
Default: FALSE
TRUE for a FILE_LIST.
Procs: create, get, set
Objects: Panel_list_item, Panel_multiline_text_item, Panel_numeric_text_item, Panel_text_item, Panel_slider_item, File_list

PANEL_REPAINT_PROC

Specifies the name of the client-supplied panel background repaint procedure.

Argument: void (*_panel_repaint_proc_) ()
Default: NULL
Procs: create, get, set
Objects: Panel
Callback:

```
void
panel_repaint_proc(panel, pw, repaint_area)
    Panel      panel;
    Xv_Window  pw;      /* paint window */
    Rectlist   *repaint_area;
```

See Also: 7.19.2

PANEL_SHOW_RANGE

If TRUE, shows the minimum and maximum slider or gauge values.

Argument: Bool
Default: TRUE
Procs: create, get, set
Objects: Panel_slider_item, Panel_gauge_item
See Also: 7.13

PANEL_SHOW_VALUE

If TRUE, shows current slider value.

Argument: Bool
Default: TRUE
Procs: create, get, set
Objects: Panel_slider_item
See Also: 7.13

PANEL_SLIDER_END_BOXES

Shows or hides the slider end boxes.

Argument: Bool
Default: FALSE
Procs: create,get,set
Objects: Panel_slider_item
See Also: 7.13

PANEL_SLIDER_WIDTH

Specifies the length of the slider bar in pixels. The length is set whether the slider is horizontally or vertically oriented.

Argument: `int`
Default: `100`
Procs: `create`, `get`, `set`
Objects: `Panel_slider_item`
See Also: 7.13

PANEL_STATUS

Returns the state of the panel. See <*xview/panel.h*> for more information.
Warning: This attribute should *only* be used from within a panel item extension.

Argument: `Panel_status *`
Procs: `get`
Objects: `Panel`

PANEL_TEXT_SELECT_LINE

Selects and highlights the entire contents of the text field.

Argument: No value
Procs: `set`
Objects: `Panel_text_item`
Usage:

```
xv_set(ptext, PANEL_TEXT_SELECT_LINE, NULL);
```

See Also: 7.15.2

PANEL_TICKS

Specifies the number of evenly-spaced tick marks to be displayed on slider or gauge panel items.

Argument: `int`
Default: `0`
Procs: `create`, `get`, `set`
Objects: `Panel_slider_item, Panel_gauge_item`
See Also: 7.13

PANEL_TOGGLE_VALUE

Specifies the value (argument 2) of a particular toggle choice (argument 1).

Argument 1: `int`
Argument 2: `int`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Panel_choice_item`

PANEL_VALUE

Indicates the current value of a panel item. Its type varies depending on the type of panel item specified. See Chapter 7, *Panels*, in the *Xview Programming Manual* for details.

Argument: Varies with panel item type
Procs: `create`, `get`, `set`
Objects: `Panel_item`
See Also: 7.6

XView Attributes

PANEL_VALUE_DISPLAY_LENGTH

Maximum number of characters to display in a text string. Note that the length of the value display may not be less than the combined width of the left and right "more text" buttons. In 12-point font, this is four characters. This attribute is intended for use with fixed-width fonts. For variable-width fonts, use the attribute `PANEL_VALUE_DISPLAY_WIDTH`. For a Multiline Text Item, this specified the length of each row (line) in the Multiline Text Field, expressed in characters.

Argument:	`int`
Default:	`80` (for Slider, Text, and Numeric Text Items) `40` (for a Multiline Text Item)
Procs:	`create`,`get`,`set`
Objects:	`Panel_slider_item` , `Panel_numeric_text_item`, `Panel_text_item`, `Panel_multiline_text_item`
See Also:	7.17, 7.15

PANEL_VALUE_DISPLAY_WIDTH

The width, in pixels, of a text field's value. For a Multiline Text Item, this is the width of each row (line) in the Multiline Text Field.

Argument:	`int`
Default:	80 default character widths (for Text and Numeric Text Items). Width corresponding to value of `PANEL_VALUE_DISPLAY_LENGTH` (for a Multiline Text Item)
Procs:	`create,get,set`
Objects:	`Panel_text_item, Panel_numeric_text_item,` `Panel_multiline_text_item`
See Also:	7.17, 7.15

PANEL_VALUE_FONT

Specifies the font to use in the value portion of the Panel Item.
Warning: Use of this attribute may cause your application to violate the OPEN LOOK Graphical User Interface Functional Specification.

Argument:	`Xv_Font`
Default:	Panel's font
Procs:	`create,get,set`
Objects:	`Panel_numeric_text_item, Panel_slider_item, Panel_text_item`

PANEL_VALUE_STORED_LENGTH

Maximum number of characters allowed in string value for a text item. For Multiline Text Items, if `PANEL_VALUE_STORED_LENGTH` is greater than `PANEL_VALUE_DISPLAY_LENGTH` times `PANEL_DISPLAY_ROWS`, then a scrollbar will be visible on the Multiline Text Field. (An outstanding bug in `TEXTSW_MEMORY_MAXIMUM` prevents `PANEL_VALUE_STORED_LENGTH` from being set to less than 1024; thus a Multiline Text Item will always have a scrollbar.)

Argument:	`int`
Default:	Same as `TEXTSW_MEMORY_MAXIMUM` default (for Multiline Text Item) 80 (for a Text or Numeric Text Item)
Procs:	`create,get,set`
Objects:	`Panel_numeric_text_item`, `Panel_text_item`, `Panel_multiline_text_item, Panel_list_item`, `Panel`
See Also:	7.17, 7.15

PANEL_VALUE_UNDERLINED

Show or hide the text field value's underlining.

Argument:	`Bool`
Default:	`TRUE`
Procs:	`create, get, set`
Objects:	`Panel_text_item, Panel_numeric_text_item`

PANEL_VALUE_X

Specifies the `x` coordinate of the *value* portion of panel items that have separate *label* and *value* entities (e.g., text items).

Argument:	`int`
Default:	(See Chapter 7, *Panels*, in the *XView Programming Manual*.)
Procs:	`create`, `get`, `set`
Objects:	`Panel_item`
See Also:	7.4.3, `PANEL_LABEL_X`

PANEL_VALUE_Y

Specifies the `y` coordinate of the *value* portion of panel items that have separate *label* and *value* entities (e.g., Text Items).

Argument:	`int`
Default:	(See Chapter 7, *Panels*, in the *XView Programming Manual*.)
Procs:	`create`, `get`, `set`
Objects:	`Panel_item`
See Also:	7.4.3, `PANEL_LABEL_Y`

PATH_IS_DIRECTORY

Specifies if the `PATH_NAME` package should accept a path name as a directory or a full path to a file name. Changing this value from `FALSE` to `TRUE` has the side effect of setting the `PATH_LAST_VALIDATED` value to `NULL` if it is not a directory.

Argument:	`Bool`
Default:	`FALSE`
Procs:	`create`, `get`, `set`
Objects:	`Path_name`
See Also:	Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

PATH_LAST_VALIDATED

`PANEL_VALUE` returns the current contents of the Textfield, `PATH_LAST_VALIDATED` returns the last path name to pass validation in the Textfield. This means that you are always guaranteed of receiving a valid path name with this attribute. Note that this is always the expanded version of the path name, whereas `PANEL_VALUE` will return the current contents of the field.

Warning: if no path name has passed validation, this will return a `NULL`. Also, setting `PATH_IS_DIRECTORY` will cause this value to become `NULL`.

Argument:	`char *`
Default:	`NULL`
Procs:	`get`
Objects:	`Path_name`
See Also:	Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

PATH_RELATIVE_TO

Specifies an absolute path that any relative path input will be appended to in order to complete the path name. This means that if someone tries to validate the string *bin*, the PATH_NAME package will append *bin* to this string to determine what the actual path name should be validated as.

Argument: `char *`
Default: NULL
Procs: `create`, `get`, `set`
Objects: `Path_name`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

PATH_USE_FRAME

Specifies if the PATH_NAME package is allowed to put error messages into its parent Frame's footer.

Argument: `Bool`
Default: FALSE
Procs: `create`, `get`, `set`
Objects: `Path_name`
See Also: Appendix D, *Version 3.2 and the File Chooser*, of *XView Programming Manual*.

SCREEN_NUMBER

Specifies the number of the screen associated with object.

Argument: `int`
Default: `0`
Procs: `get`
Objects: `Screen`
See Also: 15.2

SCREEN_SERVER

Specifies the server associated with this screen. By default, its value is the server created by opening the display specified by the value of the DISPLAY environment variable.

Return Type: `Xv_Server`
Procs: `get`
Objects: `Screen`
See Also: 15.3.3

SCROLLBAR_COMPUTE_SCROLL_PROC

Converts physical scrollbar info to client object info. The `default_compute_scroll_proc` can be called to perform the functionality that the scrollbar package normally would. If a normalize procedure is not set, then the offset becomes the viewstart (after bounds checking) and the scrollbar package will scroll to this offset into the object.

Default: `scrollbar_default_compute_scroll_proc`
Procs: `create`, `get`, `set`
Objects: `Scrollbar`

Callback:

```
void
scrollbar_default_compute_scroll_proc(
       scroll_public, pos, available_cable,
                     motion, offset, object_length)
            Scrollbar        scroll_public
            Scrollbar        scroll_public
            int              pos;
            int              available_cable;
            Scroll_motion    motion;
            unsigned long   *offset;
            unsigned long   *object_length;
```

Computes default scrolling based on the scroll position. `pos` is the position from the start of the cable event. `length` is the length of the cable. `offset` and `object_length` are output parameters. The offset is given in client units to scroll by. The object length is in client units.

SCROLLBAR_DIRECTION

Sets orientation of the scrollbar as `SCROLLBAR_VERTICAL` or `SCROLLBAR_HORIZONTAL`.

Argument: `Scrollbar_setting`
Valid Values: {`SCROLLBAR_VERTICAL`, `SCROLLBAR_HORIZONTAL`}
Default: None
Procs: `create`, `get`
Objects: `Scrollbar`

SCROLLBAR_LAST_VIEW_START

Specifies the offset of the view into the object prior to the last scroll.

Argument: `int`
Default: `0`
Procs: `get`
Objects: `Scrollbar`

SCROLLBAR_MENU

Specifies a pointer to the scrollbar's menu. Clients can add items to the default menu but cannot remove items from it.

Argument: `Menu`
Default: Scrollbar creates a default menu
Procs: `get`
Objects: `Scrollbar`

SCROLLBAR_MOTION

Provides the scrolling motion that results in a `scrollbar_request` event.

Argument: `enum`
Procs: `get`
Return Values:

```
ABSOLUTE
POINT_TO_MIN (from here_to_top on menu)
PAGE_FORWARD
```

XView Attributes

```
LINE_FORWARD
MIN_TO_POINT (from top_to_here on menu)
PAGE_BACKWARD
LINE_BACKWARD
TO_END
TO_START
PAGE_ALIGNED
```

SCROLLBAR_NORMALIZE_PROC

Specifies a procedure that does normalization (sets the viewable area of the object that the scrollbar scrolls). This function should return `vstart`. It takes the offset given by the `compute_proc` and adjusts it. The scrollbar package will then scroll to this offset into the object. The scrollbar package will call the `scrollbar_compute_scroll_proc` and the `scrollbar_normalize_proc` in that order whenever any scrolling is done.

Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Scrollbar`
Callback:

```
scrollbar_normalize_proc(sb,voffset,motion,&vstart)
```

SCROLLBAR_NOTIFY_CLIENT

Used by the Notifier. Indicates the client that is notified when the scrollbar is scrolled. See Chapter 10, *Scrollbars*, in *XView Programming Manual*.

Argument: `Xv_opaque`
Default: Subwindow scrollbar is attached to contents
Procs: `create`, `get`, `set`
Objects: `Scrollbar`
See Also: 10.4.1

SCROLLBAR_OBJECT_LENGTH

Specifies the length of the scrollable object in scrollbar units. Value must be greater than or equal to zero.

Argument: `int`
Default: `0`
Procs: `create`, `get`, `set`
Objects: `Scrollbar`
See Also: 10.2

SCROLLBAR_PAGE_LENGTH

Specifies the length of a page in scrollbar units for page-scrolling purposes.

Argument: `int`
Default: `0`
Procs: `create`, `get`, `set`
Objects: `Scrollbar`
See Also: 10.2

SCROLLBAR_PIXELS_PER_UNIT

Specifies the number of pixels constituting a scrollbar unit. For example, when scrolling a list of icons, each unit might be 64 pixels.

Argument: `int`
Default: `1`
Procs: `create`, `get`, `set`
Objects: `Scrollbar`
See Also: 10.2

SCROLLBAR_SPLITTABLE

Indicates whether the object that contains the scrollbar is splittable.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `Scrollbar`
See Also: 5.6.1

SCROLLBAR_VIEW_LENGTH

Specifies the length of the viewing window in scrollbar units.

Argument: `int`
Default: `0`
Procs: `create`, `get`, `set`
Objects: `Scrollbar`
See Also: 10.2

SCROLLBAR_VIEW_START

Specifies the current offset into the scrollbar object in scrollbar units. The value must be greater than or equal to zero.

Argument: `int`
Default: `0`
Procs: `create`, `get`,`set`
Objects: `Scrollbar`
See Also: 10.2

SEL_APPEND_TYPE_NAMES

Same as `SEL_TYPE_NAMES` except that the new list is appended to the previously set type list.

Argument: List of string (`STRING, NULL`)
Default: N/A
Procs: `create, set`
Objects: `Selection_requestor`
Usage:

```
Selection_requestor  sel;
sel = xv_create(panel, SELECTION_REQUESTOR,
                SEL_REPLY_PROC, SelectionReplyProc,
                NULL);
xv_set(sel, SEL_APPEND_TYPE_NAMES, "TARGETS", NULL,
            NULL);
```

See Also: 18.2.3.1, `SEL_TYPE_NAMES`

SEL_APPEND_TYPES

Same as SEL_TYPES except that the new list is appended to the previously set type list.

Argument: List of Atoms (XA_STRING, NULL)
Default: N/A
Procs: create,set
Objects: Selection_requestor
Usage:

```
Selection_requestor  sel;
sel = xv_create(panel, SELECTION_REQUESTOR,
                SEL_REPLY_PROC, SelectionReplyProc,
                NULL);
xv_set(sel, SEL_APPEND_TYPES, XA_STRING, NULL, NULL);
```

See Also: 18.2.3.1, SEL_TYPES

SEL_CONVERT_PROC

Specifies the procedure that is to be called whenever a client requests the current value of the selection.

Argument: int (**convert_proc*) ()
Default: sel_convert_proc
Procs: create,get,set
Objects: Selection_owner
Callback:

```
int
convert_proc( sel, replyType, replyBuff, length, format )
    Selection_owner sel;
    Atom            *replyType;
    Xv_opaque       *replyBuff;
    unsigned long   *length;
    int             *format;
```

sel specifies the selection owner.
replyType specifies the type of the selection that has been requested. It should be explicitly set to an atom which describes the converted type of the selection (for example, TEXT might have the type XA_STRING).
replyBuff is a pointer to a buffer address which contains the converted data.
length specifies a pointer to the number of elements in replyBuff. The size of an element is defined by format. The convert procedure is called with length set to the maximum allowed buffer size.
format specifies a pointer to the data format. Valid values are 8, 12, or 32 for 8-bit, 16-bit, or 32-bit quantities, respectively.

See Also: 18.2.4

SEL_COPY

SEL_COPY indicates whether or not to make a copy of the SEL_DATA data. If set to FALSE it is up to the client to maintain the data.

Argument: Bool
Default: TRUE
Procs: create,set,get
Objects: Selection_item
See Also: 18.3.1

SEL_DATA

Used to initiate a blocking selection request. The arguments to this attribute are a pointer to a long which will be set to the number of elements in the returned buffer and a pointer to an integer which will be set to the data format. `xv_get()` returns a pointer to the selection data. Clients should free the returned buffer. If the requestor client has not registered a `reply_proc()` with the selection package and is requesting for `MULTIPLE` or `INCR`, the call will return with *length* set to `SEL_ERROR` and format set to zero. If the requestor client has registered a `reply_proc()` with the selection package and requesting for `MULTIPLE` or `INCR`, the package will call the client's `reply_proc()` with the converted data. `xv_get()` returns after the transaction has completed with *length* set to `XV_OK` and format set to the returned data format. If the request fails the `xv_get()` will return with *length* set to `SEL_ERROR` and `format` set to zero. For a `Selection_item` object, this attribute specifies a pointer to the selection data.

Argument: `Xv_opaque`
Default: N/A
Procs: `get` for `Selection_requestor` objects
`get` and `set` for `Selection_item` objects
Objects: `Selection_requestor`, `Selection_item`
Usage:

```
data = (char *) xv_get(sel_requestor, SEL_DATA,
                       &length, &format);
```

See Also: 18.3.1, 18.2.3.4

SEL_DONE_PROC

Specifies the procedure that is called after the requestor has received the selection or `NULL` if the owner is not interested in being called back.

Argument: `void` (**done_proc*) ()
Default: `NULL`
Procs: `create,get,set`
Objects: `Selection_owner`
Callback:

```
void
done_proc( sel, replyBuff, target )
    Selection_owner  sel;
    Xv_opaque        replyBuff;
    Atom             target;
```

`sel` specifies the selection owner.
`replyBuff` specifies the address which contains the converted data.
`target` specifies the target type returned by the convert procedure.

See Also: 18.2.7

SEL_FIRST_ITEM

Returns a selection item.

Argument: `Selection_item`
Default: N/A
Procs: `get`
Objects: `Selection_owner`
See Also: 18.3.1, `SEL_NEXT_ITEM`

XView Attributes

SEL_FORMAT

Specifies the data format.

Argument: Xv_opaque
Default: 8
Procs: create,get,set
Objects: Selection_item
See Also: 18.3.1

SEL_LENGTH

Specifies the number of 8, 16, 32-bit elements contained in the reply.

Argument: unsigned long
Default: Type-dependent
Procs: create,get,set
Objects: Selection_item
See Also: 18.3.1

SEL_LOSE_PROC

Used to register a procedure that is called back whenever the selection owner loses the selection that it holds.

Argument: void (**lose_proc*) ()
Default: NULL
Procs: create,get,set
Objects: Selection_owner
Callback:

```
void
lose_proc( sel )
    Selection_owner sel;
```

sel specifies the selection owner.

See Also: 18.2.6

SEL_NEXT_ITEM

Returns a selection item.

Argument: Selection_item
Default: N/A
Procs: get
Objects: Selection_owner
See Also: 18.3.1, SEL_FIRST_ITEM

SEL_OWN

Setting SEL_OWN causes the selection to be acquired (TRUE) or lost (FALSE). Owning a selection without an external conversion procedure or any selection items will generate a NULL-data reply to any incoming requests.

Argument: Bool
Default: FALSE
Procs: create,get,set
Objects: Selection_owner
See Also: 18.2.4.2

SEL_PROP_DATA

Specifies the data associated with a property. Should be used in conjunction with SEL_TYPE_INDEX.

Argument: Xv_opaque
Default: NULL
Procs: create,set
Objects: Selection_requestor
See Also: 18.6, SEL_TYPE_INDEX

SEL_PROP_FORMAT

Specifies the format of the data associated with a property. Should be used in conjunction with SEL_TYPE_INDEX.

Argument: int
Default: N/A
Procs: create,set
Objects: Selection_requestor
See Also: 18.6, SEL_TYPE_INDEX

SEL_PROP_INFO

Returns the property data. This attribute should be used from a conversion procedure. It returns the data that has been set on the selection notifier's property by the requestor client.

Argument: Sel_prop_info *
Default: N/A
Procs: get
Objects: Selection_owner
See Also: 18.7, SEL_TYPE_INDEX

SEL_PROP_LENGTH

Specifies the length of the data associated with a property. Should be used in conjunction with SEL_TYPE_INDEX.

Argument: unsigned long
Default: N/A
Procs: create,set
Objects: Selection_requestor
See Also: 18.6, SEL_TYPE_INDEX

SEL_PROP_TYPE

Specifies the type of the data associated with a property.

Argument: Atom
Default: N/A
Procs: create,set
Objects: Selection_requestor
See Also: 18.6, SEL_PROP_INFO, SEL_TYPE_INDEX

SEL_PROP_TYPE_NAME

Specifies the type of the data associated with a property. It takes a string argument which is interned into an atom. Should be used in conjunction with SEL_TYPE_INDEX.

Argument: Atom
Default: N/A
Procs: create,set
Objects: Selection_requestor
See Also: 18.6, SEL_PROP_INFO, SEL_TYPE_INDEX

SEL_RANK

Used to set the rank of the selection. Pre-defined atoms are `XA_PRIMARY` and `XA_SECONDARY`. The client should set the rank to the atom representing the selection.

Argument: `Atom`
Default: `XA_PRIMARY`
Procs: `create,get,set`
Objects: `Selection`
See Also: SEL_RANK and Chapter 12, *Interclient Communication*, in *Xlib Programming Manual*.

SEL_RANK_NAME

Used to set the rank of the selection. The package will intern the selection atom using this atom name (a string).

Argument: `*char`
Default: `XA_PRIMARY`
Procs: `create,get,set`
Objects: `Selection`
See Also: 18.2.2

SEL_REPLY_PROC

A procedure that is called when a response to a request comes in.

Argument: `void` (**reply_proc*) ()
Default: `NULL`
Procs: `create,get,set`
Objects: `Selection_requestor`
Callback:

```
void
reply_proc( sel_req, target, type, replyValue, length, format )
    Selection_requestor  sel_req;
    Atom                 target;
    Atom                 type;
    Xv_opaque            replyValue;
    unsigned long        length;
    int                  format;
```

If the selection conversion fails, this routine is called with `replyValue` set to an error code and `length` set to `SEL_ERROR`.
`sel_req` specifies the selection requestor.
`target` specifies the target type set by the convert procedure.
`type` specifies the specifies the type returned by the convert procedure.
`replyValue` specifies the data content returned.
`length` specifies the length of the data returned by the convert procedure.
`format` specifies the format of the data returned by the convert procedure.

If the selection content is larger than the server's maximum request size or if the selection owner has decided to transfer the selection data in increments, the selection package will send the data to the requestor in chunks.

See Also: 18.2.5

SEL_TIME

Specifies the time of the acquisition or the request attempt.

Argument: `struct timeval *`
Default: Last event time
Procs: `create, get, set`
Objects: `Selection`
See Also: 18.2.3

SEL_TIMEOUT_VALUE

Selection timeout value. This value indicates the number of seconds that a requestor or a selection owner waits for a response during the selection transfer.

Argument: `unsigned int`
Default: From the value of the resource `Selection.Timeout`
Procs: `create,get,set`
Objects: `Selection`
See Also: 18.2.3.3, 18.2.2

SEL_TYPE

Specifies an atom type that the client is requesting. For a selection item, specifies the type that the package will convert to.

Argument: `Atom`
Default: `XA_STRING`
Procs: `create,get,set`
Objects: `Selection_item`, `Selection_requestor`
See Also: 18.2.3.1, `SEL_TYPES`, `SEL_TYPE_NAME`, and Chapter 12, *Interclient Communication*, in *Xlib Programming Manual*.

SEL_TYPE_INDEX

Specifies an index to the `SEL_TYPES` or `SEL_TYPE` list. This attribute is also used to specify an index to the `SEL_TYPE_NAME` and `SEL_TYPE_NAMES` lists. This attribute is used in conjunction with `SEL_PROP_DATA`, `SEL_PROP_LENGTH`, `SEL_PROP_FORMAT`, `SEL_PROP_TYPE`, and `SEL_PROP_TYPE_NAME` to associate data with a property used by the requestor.

Argument: `int`
Default: N/A
Procs: `create,set`
Objects: `Selection_requestor`
Usage: (Multiple Request)

```
xv_set( sel_req,
      SEL_TYPES, XA_STRING, INSERT_SELECTION, NULL,
      SEL_TYPE_INDEX, 1,
      SEL_PROP_DATA, dataPointer,
      SEL_PROP_LENGTH, 20,
      NULL );
```

Usage: (Single Request)

```
xv_set( sel_req, SEL_TYPE, XA_STRING,
                 SEL_TYPE_INDEX, 0,
                 SEL_PROP_DATA, dataPointer,
                 SEL_PROP_LENGTH, 20,
          NULL );
```

See Also: 18.6

XView Attributes

SEL_TYPE_NAME

Same as SEL_TYPE except that the argument is the name of the requested selection type instead of an atom. The package will intern the requested atom name. For a selection item, specifies the type name that the package will convert to.

Argument: *char
Default: "STRING"
Procs: create,get,set
Objects: Selection_requestor Selection_item
See Also: 18.2.3.1

SEL_TYPE_NAMES

Specifies a NULL-terminated list of atom type names that the client is requesting.

Argument: List of string
Default: "STRING, NULL"
Procs: create,get,set
Objects: Selection_requestor
See Also: 18.2.3.1

SEL_TYPES

Specifies a NULL-terminated list of atom types that the client is requesting. The effect will be as if a sequence of SelectionRequest events is delivered to the selection owner, one for each atom. This attribute will initiate a MULTIPLE request.

Argument: List of Atoms
Default: XA_STRING,NULL
Procs: create,get,set
Objects: Selection_requestor
See Also: 18.2.3.1

SELN_*

Appendix A presents all of the SELN_ attributes.

SERVER_ATOM

SERVER_ATOM is equivalent to XInternAtom() (with the only_if_exists flag set to FALSE) except that it caches the results on the server object so that subsequent requests for the same atom will not require a round-trip to the X server.

Return Type: Atom
Procs: get
Usage:

```
Atom  atom;
atom = (Atom) xv_get(server_object,
                     SERVER_ATOM, "TIMESTAMP");
```

See Also: 15.3.3, SERVER_ATOM_NAME

SERVER_ATOM_NAME

SERVER_ATOM_NAME is equivalent to XGetAtomName() except that it caches the results on the server object so that subsequent requests will not require a roundtrip to the X server. The returned string is maintained by XView and should not be modified or freed. XView will free up all strings associated with atoms on that server when the server object is destroyed.

Return Type: char *
Procs: get
Usage:

```
char *atom_name;
atom_name (char *)xv_get(server_object, atom);
```

See Also: 15.3.3, SERVER_ATOM

SERVER_EXTENSION_PROC

Specifies the procedure used to handle server extension events.

Argument: void (**extension_proc*)()
Default: NULL
Procs: create, get, set
Objects: Server
Callback:

```
void
extension_proc(display, xevent, window)
     Display    *display;
     Event      *xevent;
     Xv_window   window;
```

See Also: 6.10

SERVER_EXTERNAL_XEVENT_MASK

This attribute, together with SERVER_EXTERNAL_XEVENT_PROC, allows a client to receive notification for X events destined for X windows which are not local to a client. For each non-local X window, the client can specify a different set of X events for which it needs notification. Additionally, an XView object handle is provided as an argument which is returned as a parameter during callback. For xv_create and xv_set this attribute takes these three arguments: an XID of a window, an event mask (see *<X11/X.h>* for details), and an XView object handle. SERVER_EXTERNAL_XEVENT_MASK for xv_get takes two arguments: an XID of a window, and an XView object handle and returns a mask.

Argument 1: unsigned long
Argument 2: unsigned long
Argument 3: Xv_opaque
Default: None
Procs: create,get,set
Objects: Server
Usage:

```
xv_set (server, SERVER_EXTERNAL_XEVENT_MASK,
                    RootWindow(dpy, 0),
                    ButtonPressMask | PropertyChangeMask,
                    frame,
                SERVER_EXTERNAL_XEVENT_PROC,
                    root_event_proc, frame,
                NULL);
```

See Also: 6.11, SERVER_EXTERNAL_XEVENT_PROC

XView Attributes

SERVER_EXTERNAL_XEVENT_PROC

This attribute, together with attribute SERVER_EXTERNAL_XEVENT_MASK allows a client to receive notification for X events destined for X windows which are not local to the client. A client can register a separate callback procedure for each XView object handle. For xv_create and xv_set this attribute takes two arguments. For xv_get it takes one argument: an XView object handle and returns a function name. A NULL first argument temporarily disables callbacks.

Argument 1: void (**callback_proc*)()
Argument 2: Xv_opaque
Default: None
Procs: create,get,set
Objects Server
Usage:

```
xv_set (server,
          SERVER_EXTERNAL_XEVENT_MASK,
                   RootWindow(dpy, 0),
                   ButtonPressMask | PropertyChangeMask,
                   frame,
          SERVER_EXTERNAL_XEVENT_PROC,
                   root_event_proc, frame,
          NULL);
```

Callback:

```
void
callback_proc (server, display, xevent, handle)
  Xv_server   server;
  Display    *display;
  XEvent     *xevent;
  Xv_opaque   handle;
```

See Also: 6.11, SERVER_EXTERNAL_XEVENT_MASK

SERVER_IMAGE_BITMAP_FILE

Specifies a file containing the X11 bitmap from which the server image is created.

Argument: char *
Default: None
Procs: create
Objects: Server
See Also: 15.4.1

SERVER_IMAGE_BITS

Specifies the SunView pixrect image bits for the server image. Use SERVER_IMAGE_X_BITS for standard X11 bitmaps.

Argument: short *
Default: Uninitialized
Procs: create, get, set
Objects: Server_image
See Also: 15.4.1

SERVER_IMAGE_CMS

Specifies the colormap segment to be used in converting the server image data (which are logical colormap indices) into pixel values in the pixmap used to draw the server image.

Argument: `cms`
Default: The default `cms` for the application
Procs: `create,get`
Usage:

```
Server_image image;

image = (Server_image)xv_create(NULL,
            SERVER_IMAGE,
            ...
            SERVER_IMAGE_CMS, cms,
            ...
            NULL);
```

See Also: 15.4.1

SERVER_IMAGE_COLORMAP

Specifies the name of the colormap segment to be used in converting the server image data (which are logical colormap indices) into pixel values in the pixmap used to draw the server image.

This is basically a backwards compatibility attribute, and where possible, `SERVER_IMAGE_CMS` should be used instead.

Argument: `char *`
Default: "`xv_default_cms`"
Procs: `get, create`
Usage:

```
Cms cms;
Server_image image;

cms = (Cms)xv_create(NULL, CMS,
    ...
    CMS_NAME, "palette",
    ...
    NULL);

image = (Server_image)xv_create(NULL, SERVER_IMAGE,
    ...
    SERVER_IMAGE_COLORMAP, "palette",
    ...
    NULL);
```

SERVER_IMAGE_DEPTH

Specifies the bit plane depth of the server image.

Argument: `int`
Default: `1`
Procs: `create`, `get`, `set`
Objects: `Server_image`
See Also: 15.4.1

SERVER_IMAGE_PIXMAP

Allows an existing pixmap to be associated with a server image. An xv_get using this attribute is equivalent to an xv_get of the XV_XID of the server image.

Argument: Pixmap
Default: None
Procs: create, get, set
Objects: Server_image
See Also: 15.4.1

SERVER_IMAGE_SAVE_PIXMAP

Allows the application to specify that the old pixmap must not be destroyed if a new pixmap is created as the result of changing any server image attributes or assigning a new pixmap directly. You should retain a handle to the old pixmap first.

Argument: Bool
Default: FALSE
Procs: create,get,set
Objects: Server_image
See Also: 15.4.1

SERVER_IMAGE_X_BITS

Specifies the bits to use in a server image. Bits are stored in an array of char.

Argument: char *
Default: None
Procs: create, set
Objects: Server_image
See Also: 15.4.1

SERVER_NTH_SCREEN

Specifies the screen with given number. Returns NULL if screen does not exist.

Argument 1: int
Argument 2: Screen
Default: N/A
Procs: create, get, set
Objects: Server
See Also: 15.2.1

SERVER_SYNC

Flushes the request buffer and waits for all events and errors to be processed by the server. If the argument is TRUE, then SERVER_SYNC discards all events on the input queue.

Argument: Bool
Default: N/A
Procs: set
Objects: Server

SERVER_SYNC_AND_PROCESS_EVENTS

Same as SERVER_SYNC, but this processes any events that arrive as a result of the XSync().

Argument: No value
Default: None
Procs: create, set
Objects: Server
See Also: 15.3.3

TEXTSW_ACTION_*

Appendix C, *Textsw Action Attributes*, describes the `textsw` action attributes. These attributes are only valid for use with a `textsw` notify procedure.

TEXTSW_AGAIN_RECORDING

If `FALSE`, changes to the `textsw` are not repeated when the user invokes `AGAIN`. Disabling reduces memory overhead.

Argument: `Bool`
Default: `TRUE`
Procs: `create`, `get`, `set`
Objects: `Textsw`

TEXTSW_AUTO_INDENT

If `TRUE`, automatically indents a new line to match the previous line.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `Textsw`

TEXTSW_AUTO_SCROLL_BY

Specifies the number of lines to scroll when type-in moves insert point below the view.

Argument: `int`
Default: `1`
Procs: `create`, `get`, `set`
Objects: `Textsw`

TEXTSW_BLINK_CARET

Determines whether the caret blinks (for better performance don't blink).

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `Textsw`

TEXTSW_BROWSING

If `TRUE`, prevents editing of displayed text. If another file is loaded in, browsing stays on.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `Textsw`

TEXTSW_CHECKPOINT_FREQUENCY

Specifies the number of edits between checkpoints. Set to `0` to disable checkpointing.

Argument: `int`
Default: `0`
Procs: `create`, `get`, `set`
Objects: `Textsw`

TEXTSW_CLIENT_DATA

Specifies the pointer to arbitrary client data.

Argument: `Xv_opaque`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Textsw`

TEXTSW_CONFIRM_OVERWRITE

Specifies confirmation of any request to write to an existing file.

Argument: `Bool`
Default: `TRUE`
Procs: `create`, `get`, `set`
Objects: `Textsw`

TEXTSW_CONTENTS

Specifies the text for a subwindow. `xv_get` needs additional parameters (see Chapter 8, *Text Subwindows*, in *XView Programming Manual*).

Argument: `char *`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Textsw`
See Also: 8.6.2

TEXTSW_CONTROL_CHARS_USE_FONT

If `FALSE`, control characters always display as an up arrow followed by a character instead of whatever glyph is in the current font.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `Textsw`

TEXTSW_DESTROY_VIEW

The current view will be destroyed.
Argument: No value

TEXTSW_DISABLE_CD

Stops `textsw` from changing current working directory and grays out associated items in the menu.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `Textsw`

TEXTSW_DISABLE_LOAD

Prevents files from being loaded into the `textsw` and grays out the associated items in the menu.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `Textsw`

TEXTSW_EDIT_COUNT

Monotonically increments count of the number of edits made to the `textsw`.

Argument: `int`
Procs: `get`
Objects: `Textsw`

TEXTSW_EXTRAS_CMD_MENU

Returns the Extras submenu of the TEXT PANE menu for the `textsw`.

Return Type: `Menu`
Procs: `get`
Objects: `Textsw`

TEXTSW_FILE

For `xv_create` and `xv_set`, specifies the name of the file to load; for `xv_get`, returns the name of the file loaded or `NULL` if no file was loaded.

Argument: `char *`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Textsw`
See Also: 8.4.1

TEXTSW_FILE_CONTENTS

Initializes the text subwindow contents from a file, yet still edits the contents in memory as if specified using `TEXTSW_FILE`.

Argument: `char *`
Default: `NULL`
Procs: `create`, `set`
Objects: `Textsw`
See Also: 8.6.1

TEXTSW_FIRST

Specifies the zero-based index of first displayed character.

Argument: `int`
Procs: `create`, `get`, `set`
Objects: `Textsw`
See Also: 8.4.1

TEXTSW_FIRST_LINE

Specifies the zero-based index of first displayed line.

Return Type: `int`
Procs: `get`
Objects: `Textsw`
See Also: 8.7.2

TEXTSW_FONT

Specifies the font to use in a text subwindow.

Argument: `Xv_Font`
Procs: `create`, `get`, `set`
Objects: `Textsw`
See Also: 8.1

TEXTSW_HISTORY_LIMIT

Specifies the number of user action sequences that can be undone.

Argument: `int`
Default: `50`
Procs: `create`, `get`, `set`
Objects: `Textsw`

TEXTSW_IGNORE_LIMIT

Specifies the number of edits `textsw` allows before vetoing `destroy`. Valid values are `0`, which means the `destroy` will be vetoed if there have been any edits, and `TEXTSW_ INFINITY`, which means the `destroy` will never be vetoed. Vetoing a destroy means a confirm notice is displayed when the `textsw` is about to be destroyed. This veto confirm message is displayed only if the ignore limit is set to `0`. A `textsw` is destroyed when the `textsw` or its enclosing frame is the object of an `xv_destroy()` call, which occurs when the application is quit from the Window Manager menu.

Argument: `int`
Default: `0`
Procs: `create`, `get`, `set`
Objects: `Textsw`

TEXTSW_INSERT_FROM_FILE

Inserts the contents of a file into a text subwindow at the current insertion point.

Argument: `char *`
Default: None
Procs: `set`, `create`
Objects: `Textsw`
See Also: 8.6.3

TEXTSW_INSERT_MAKES_VISIBLE

Controls whether insertion causes repositioning to make inserted text visible. Possible values are `TEXTSW_ALWAYS` or `TEXTSW_ IF_AUTO_SCROLL`.

Argument: `Textsw_enum`
Default: `TEXTSW_ALWAYS`
Procs: `create`, `get`, `set`
Objects: `Textsw`
See Also: 8.7.4.2

TEXTSW_INSERTION_POINT

Specifies the index of the current insertion point.

Argument: `Textsw_index`
Default: None
Procs: `create`, `get`, `set`
Objects: `Textsw`
See Also: 8.4.3.1

TEXTSW_LENGTH

Specifies the length of the `textsw`'s contents.

Argument: `int`
Procs: `get`
Objects: `Textsw`
See Also: 8.3

TEXTSW_LINE_BREAK_ACTION

Determines how the `textsw` treats file lines that are too big to fit on one display line. `TEXTSW_CLIP` clips the line when it gets too long to fit; you don't see any additional characters typed on the line when the line exceeds the width of the textsw. `TEXTSW_WRAP_CHAR` wraps the line around so the part of the line that exceeds the width of the textsw is displayed on the next line(s). Textsw does not insert a CR. It just displays it as if a CR were there. If this happens in the middle of the word, the textsw will display the line as if a CR were inserted in the middle of the word. `TEXTSW_WRAP_WORD` wraps at the word level, thus, it will not break a word up to wrap around. It will figure out where the word starts and display as if a CR were inserted before the word.

Argument: `Textsw_enum`
Valid Values: {`TEXTSW_CLIP`, `TEXTSW_WRAP_AT_CHAR`, `TEXTSW_WRAP_AT_WORD`}
Default: `TEXTSW_WRAP_AT_WORD`
Procs: `create`, `get`, `set`
Objects: `Textsw`
See Also: 8.7.1

TEXTSW_LOWER_CONTEXT

Specifies the minimum number of lines to maintain between insertion point and bottom of view. A value of –1 turns auto scrolling off.

Argument: `int`
Default: `0`
Procs: `create`, `get`, `set`
Objects: `Textsw`

TEXTSW_MEMORY_MAXIMUM

Specifies how much memory to use when not editing files (e.g., editing in memory). This attribute only takes effect at create time or after the window is reset with `textsw_reset`. The lower bound is 1K bytes, which is silently enforced.

Argument: `int`
Default: `20000`
Procs: `create`, `get`, `set`
Objects: `Textsw`
See Also: 8.4.8

TEXTSW_MODIFIED

Specifies whether the `textsw` has been modified.

Argument: `Bool`
Procs: `get`
Objects: `Textsw`
See Also: 8.4.1

TEXTSW_MULTI_CLICK_SPACE

Specifies the maximum number of pixels between successive mouse clicks to still have the clicks considered a multi-click.

Argument: `int`
Default: `4`
Procs: `create`, `get`, `set`
Objects: `Textsw`

TEXTSW_MULTI_CLICK_TIMEOUT

Specifies the maximum number of milliseconds between successive mouse clicks to still have the clicks considered a multi-click.

Argument: `int`
Default: `390`
Procs: `create`, `get`, `set`
Objects: `Textsw`

TEXTSW_NOTIFY_PROC

Names a notify procedure.

Argument: `void (*`*notify_proc*`) ()`
Default: `NULL`
Procs: `create`, `set`
Objects: `Textsw`
Callback:

```
void
notify_proc(textsw, avlist)
        Textsw       textsw
        Attr_avlist  avlist
```

Notify procedure installed by application using `TEXTSW_NOTIFY_PROC`.

See Also: 8.12

TEXTSW_READ_ONLY

If `TRUE`, prevents editing of the displayed text. If another file is loaded in, read-only status is turned off again.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `Textsw`

TEXTSW_STATUS

If set, specifies the address of a variable of type `Textsw_status`. A value that reflects what happened during the call to `xv_create` is then written into it. Note that this attribute must appear in the attribute list before the operation whose status you want to determine.

Argument: `Textsw_status *`
Default: None
Procs: `create`
Objects: `Textsw`
See Also: 8.2

TEXTSW_STORE_CHANGES_FILE

Controls whether the *target* filename given to `textsw_store()` to save the current contents to a file changes the name of the file being edited (`TEXTSW_FILE`).

Argument: `Bool`
Default: `TRUE`
Procs: `create`, `set`
Objects: `Textsw`
See Also: 8.5.1

TEXTSW_SUBMENU_EDIT

Returns the submenu associated with the text pane menu `Edit` submenu.

Argument: `Menu`
Procs: `get`
Objects: `Textsw`

TEXTSW_SUBMENU_FILE

Returns the submenu associated with the text pane menu `File` submenu.

Argument: `Menu`
Procs: `get`
Objects: `Textsw`

TEXTSW_SUBMENU_FIND

Returns the submenu associated with the text pane menu `Find` submenu.

Argument: `Menu`
Procs: `get`
Objects: `Textsw`

TEXTSW_SUBMENU_VIEW

Returns the submenu associated with the text pane menu `View` submenu.

Argument: `Menu`
Procs: `get`
Objects: `Textsw`

TEXTSW_UPPER_CONTEXT

Specifies the minimum number of lines to maintain between the start of the selection and top of view. A value of –1 means defeat the normal actions.

Argument: `int`
Default: `2`
Procs: `create`, `get`, `set`
Objects: `Textsw`

TTY_ARGV

The command, specified as an argument vector, that the tty subwindow executes. Using the value `TTY_ARGV_DO_NOT_FORK` lets a user start a tty subwindow without forking off a shell.

Argument: `char**`
Default: None
Procs: `create`, `set`
Objects: `Textsw`
See Also: 9.4, 9.1

TTY_CONSOLE

If `TRUE`, tty subwindow grabs console output.

Argument: `Bool`
Default: `FALSE`
Procs: `set`, `create`
Objects: `Tty`

XView Attributes

TTY_PAGE_MODE

If TRUE, output will stop after each page.

Argument: Bool
Default: FALSE
Procs: create, get, set
Objects: Tty

TTY_PID

The process ID of the program being run in the tty subwindow.

Argument: int
Default: None
Procs: create, get, set
Objects: Tty
See Also: 9.3

TTY_QUIT_ON_CHILD_DEATH

If TRUE, window_done is called on the parent frame of the tty window when its child terminates.

Argument: Bool
Default: FALSE
Procs: set, create
Objects: Tty
See Also: 9.3

TTY_TTY_FD

Gets the file descriptor of the pseudo-tty associated with the tty subwindow.

Argument: int
Procs: get
Objects: Tty
See Also: 9.4

WIN_ALARM

Rings the bell for that window.

Argument: No value
Default: off
Procs: set, create
Objects: Xv_Window
See Also: WIN_ALARM_DATA

WIN_ALARM_DATA

Gets the number of flashes and number of beeps used as alarm. (Currently always returns the same value.)

Argument: Xv_opaque
Default: 1 beep, 1 flash
Procs: get
Objects: Xv_Window
See Also: WIN_ALARM

WIN_BACKGROUND_COLOR

Specifies the background color of a window as an index into the colormap segment associated with the window.

Argument: `int`
Default: `0`
Procs: `create`, `get`, `set`
Objects: `Xv_Window`
See Also: 4.7, 21.3.1.1, `WIN_CMS`, `WIN_FOREGROUND_COLOR`

WIN_BACKGROUND_PIXMAP

Specifies the background pixmap of a window.

Argument: `Pixmap`
Default: `NULL`
Procs: `create`, `get`, `set`
Objects: `Xv_Window`

WIN_BELOW

Causes the window to be positioned below the sibling window given as the value. Restricted to windows with the same immediate parent (i.e., subwindows). Does not affect the `XV_X` of the window.

Argument: `Xv_Window`
Default: N/A
Procs: `set`, `create`
Objects: `Xv_Window`

WIN_BIT_GRAVITY

Sets the Xlib-specific "bit gravity" on the underlying X window associated with the object. See *<X11/X.h>* for a list of legal values.

Argument: `int`
Default: `NorthWestGravity`
Procs: `create`, `get`
Objects: `Xv_Window`
See Also: 5.3, `CANVAS_FIXED_IMAGE`, `CANVAS_RESIZE_PROC`

WIN_BORDER

Controls whether a window has a border.

Argument: `Bool`
Default: `TRUE`
Procs: `create`, `get`, `set`
Objects: `Xv_Window`

WIN_CLIENT_DATA

Specifies an arbitrary value to be attached to a window.

Argument: `caddr_t`
Default: None
Procs: `create`, `get`, `set`
Objects: `Xv_Window`
See Also: 7.19.1, `XV_KEY_DATA`

WIN_CMD_LINE

Lets an application set the command-line options that can be used to (re)start it. The options passed, in addition to XView options are stored on a property called `WM_COMMAND` on the frame window. (The program *xprop* can be used to display a window's properties). Only one base frame window of the application needs to have this property set. This property is read possibly by a session manager to restart clients. Setting this attribute to -1 prevents any command-line option information from being saved on the frame. If there are two or more base frames in the application, the second and subsequent base frames should set their `WIN_CMD_LINE` attributes to -1 if they want to avoid multiple invocations of the same application by the session manager.

The string passed is copied and cached on the frame.

Warning: Usage of this attribute will make the application non-ICCCM compliant. Use `FRAME_WM_COMMAND_ARGC_ARGV` instead. Intermixing the usage of `WIN_CMD_LINE` and `FRAME_WM_COMMAND_*` attributes has unpredictable results.

Argument:	`char *`
Default:	None
Procs:	`create,get,set`
Objects:	`Xv_Window`
See Also:	`FRAME_WM_COMMAND_ARGC_ARGV`, `FRAME_WM_COMMAND_ARGC`, `FRAME_WM_COMMAND_ARGV`, `FRAME_WM_COMMAND_STRINGS`, Chapter 12, *Interclient Communication*, in *Xlib Programming Manual*.

WIN_CMS

Window-based objects use colormap segments to get their colors. These objects get a default colormap segment when they are created, but you can assign a new one using this attribute.

Argument:	`Cms`
Default:	Depends on `WIN_INHERIT_COLORS`, typically inherits parent's `cms`
Procs:	`create`, `get`, `set`
	Only use `create` and `get` for a `FILE_CHOOSER`.
Objects:	`Xv_Window`
See Also:	21.5, 21.2.2, 21.1, `WIN_FOREGROUND_COLOR, WIN_BACKGROUND_COLOR`

WIN_CMS_DATA

This attribute is obsolete; where possible, the `Cms` package should be used. It specifies the data for the colormap segment associated with the window. The data is written into the currently allocated colormap segment. If a new segment is desired, set it using `WIN_CMS_NAME` first.

Argument:	`Xv_cmsdata *`
Default:	None
Procs:	`create`, `get`, `set`
Objects:	`Xv_Window`

WIN_CMS_NAME

This attribute is obsolete; where possible, the `Cms` package should be used. Specifies the colormap segment to be associated with the window.

Argument:	`char *`
Default:	None
Procs:	`create`, `get`, `set`
Objects:	`Xv_Window`
See Also:	21.2.2

WIN_COLLAPSE_EXPOSURES

Collapses contiguous multiple `Exposure` (and `GraphicsExpose`) events destined for the same window that are grouped by the `count` field in the X `Expose` event.

Argument: `Bool`
Default: `TRUE`
Procs: `create`, `get`, `set`
Objects: `Xv_Window`
See Also: 5.3

WIN_COLUMN_GAP

Specifies the gap between columns in the window.

Argument: `int`
Default: `0`
Procs: `create`, `get`, `set`
Objects: `Xv_Window`

WIN_COLUMN_WIDTH

Specifies the width of a column in the window.

Argument: `int`
Default: Font's default width
Procs: `create`, `get`, `set`
Objects: `Xv_Window`

WIN_COLUMNS

Specifies the window's width (including left and right margins) in columns relative to the width of the window's font. `WIN_COLUMNS` is not a valid `xv_create()` attribute for a `Textsw`. Issue a separate `xv_set()` call after the `xv_create()` in order to set this attribute on a `Textsw`.

Argument: `int`
Default: Varies based on font, typically 80 columns
Procs: `create`, `get`, `set`
Objects: `Xv_Window`

WIN_CONSUME_EVENT

Specifies that the window will accept an event of type specified. The event is appended to the current input mask.

Argument: Event code (defined in <*xview/win_input.h*>)
Default: Varies from package to package
Procs: `create`, `get`, `set`
Objects: `Xv_Window`
See Also: 6.3.2, 5.7.1, `WIN_CONSUME_X_EVENT_MASK`, `WIN_CONSUME_EVENTS`,`WIN_IGNORE_EVENT`, `WIN_IGNORE_EVENTS`, `WIN_X_EVENT_MASK`

WIN_CONSUME_EVENTS

Specifies a `NULL`-terminated list of event types that this window accepts. The event is appended to the current input mask.

Argument: List of XView events
Default: Varies from package to package
Procs: `create`, `set`
Objects: `Xv_Window`
See Also: 6.3.2, `WIN_CONSUME_X_EVENT_MASK`, `WIN_CONSUME_EVENT`, `WIN_IGNORE_EVENT`, `WIN_IGNORE_EVENTS`, `WIN_X_EVENT_MASK`

XView Attributes

WIN_CONSUME_X_EVENT_MASK

The input mask is specified using X event masks found in *<X11/X.h>*. The event is appended to the current input mask.

Argument: `unsigned long`
Default: Varies from package to package
Procs: `create`, `get`, `set`
Objects: `Xv_Window`
See Also: 6.3.1, `WIN_CONSUME_EVENT`, `WIN_CONSUME_X_EVENTS`, `WIN_IGNORE_EVENT`, `WIN_IGNORE_EVENTS`, `WIN_X_EVENT_MASK`

WIN_CURSOR

Specifies the window's cursor. You must supply the handle of an XView window as the parent parameter when getting `WIN_CURSOR`. Getting `WIN_CURSOR` on the root window returns `NULL`.

Argument: `Xv_Cursor`
Default: Default X server cursor
Procs: `create`, `get`, `set`
Objects: `Xv_Window`
Usage:

```
default_cursor = (Xv_cursor)xv_get(paint_win, WIN_CURSOR);

xv_set(paint_win, WIN_CURSOR, default_cursor, NULL);
```

WIN_DEPTH

This attribute is obsolete. Use `XV_DEPTH` instead. Specifies the pixel depth of the window.

Argument: `int`
Default: Function of the screen
Procs: `create`, `get`
Objects: `Xv_Window`
See Also: 21.8, 2.4.2

WIN_DYNAMIC_VISUAL

This attribute is obsolete. Use the attribute `XV_VISUAL` to specify the visual used in the creation of the window or colormap segment.

Argument: `Bool`
Default: `FALSE`
Procs: `create`
Objects: `Xv_Window`

WIN_EVENT_PROC

Specifies a callback procedure where window events are delivered.

Argument: `void (*`*event_proc*`)()`
Default: None
Procs: `create`, `get`, `set`
Objects: `Xv_Window`
Callback:

```
void
event_proc(window, event, arg)
   Xv_Window    window;
   Event       *event;
   Notify_arg   arg;
```

See Also: Chapter 20, *The Notifier*, in *XView Programming Manual*.

WIN_FIT_HEIGHT

Causes the window to shrink or expand its height according to the window's contents leaving a margin specified by the value given. Typically used with panels to fit panel items.

Argument: `int`
Default: Depends on window size
Procs: `create`, `get`, `set`
Objects: `Xv_Window`
See Also: `window_fit()` macro in Section 3, *Procedures and Macros*.

WIN_FIT_WIDTH

Causes the window to shrink or expand its width according to the window's contents leaving a margin specified by the value given.

Argument: `int`
Default: Depends on window size
Procs: `create,get,set`
Objects: `Xv_Window`
See Also `window_fit()` macro in Section 3, *Procedures and Macros*

WIN_FOREGROUND_COLOR

Specifies the foreground color of a window as an index into the colormap segment associated with the window.

Argument: `int`
Default: `size` –1 (where `size` is the number of colors in the colormap segment associated with the window)
Procs: `create`, `get`, `set`
Objects: `Xv_Window`
See Also: 4.7, 21.3.1.1, `WIN_CMS`, `WIN_BACKGROUND_COLOR`

WIN_FRAME

Returns the window's frame.

Return Type: `Frame`
Default: N/A
Procs: `get`
Objects: `Xv_Window`

WIN_FRONT

Indicates that the window should move to the front of the stacking order. Setting this attribute does *not* map the window. Use `XV_SHOW` to raise and map a window.

Argument: No value
Procs: `create`, `set`
Objects: `Xv_Window`

WIN_GRAB_ALL_INPUT

Specifies that the window will get all events regardless of location of the pointer. Performs a grab of the keyboard and the pointer.

Argument: `Bool`
Default: `FALSE`
Procs: `set`, `create`
Objects: `Xv_Window`
See Also: 6.6.1.2, `FULLSCREEN` attributes

WIN_HORIZONTAL_SCROLLBAR

This attribute is obsolete. Use OPENWIN_HORIZONTAL_SCROLLBAR instead. It is a handle to the horizontal scrollbar for that window.

Argument: Scrollbar
Default: NULL
Procs: create, get, set
Objects: Xv_Window

WIN_IGNORE_EVENT

Specifies that the window will not receive this event. (Certain events cannot be ignored.)

Argument: XView event code (see *<xview/win_input.h>*)
Default: None
Procs: create, set
Objects: Xv_Window
See Also: 6.3.2, WIN_CONSUME_EVENT, WIN_CONSUME_X_EVENT_MASK, WIN_CONSUME_EVENTS, WIN_IGNORE_EVENTS

WIN_IGNORE_EVENTS

Specifies a NULL-terminated list of events that this window will not receive. (Certain events cannot be ignored.)

Argument: List of event codes (see *<xview/win_input.h>*)
Default: None
Procs: set, create
Objects: Xv_Window
See Also: 6.3.2, WIN_CONSUME_EVENT, WIN_CONSUME_X_EVENT_MASK, WIN_CONSUME_EVENTS, WIN_IGNORE_EVENT

WIN_IGNORE_X_EVENT_MASK

Prevents the specified event masks from being delivered to the event handler. The input mask is specified using X event masks found in *<X11/X.h>*.

Argument: unsigned long
Default: Varies from package to package
Procs: create, get, set
Objects: Xv_Window
See Also: 6.3.1, WIN_CONSUME_X_EVENT_MASK, WIN_EVENT_PROC, WIN_IGNORE_EVENT, WIN_IGNORE_EVENTS, WIN_X_EVENT_MASK

WIN_INHERIT_COLORS

This attribute specifies whether the window should inherit the colors used by its parent. If TRUE, the window will inherit the WIN_CMS, XV_VISUAL, WIN_FOREGROUND_COLOR, and WIN_BACKGROUND_COLOR attributes of its parent. If FALSE, it will use the defaults for all of these attributes.

Argument: Bool
Default: TRUE
Procs: create,get,set
Objects: Xv_Window

WIN_INPUT_MASK

Specifies the window's input mask. This overwrites the current mask.

Argument: `Inputmask *`
Default: Varies from package to package
Procs: `create`, `get`, `set`
Objects: `Xv_Window`
See Also: 6.8

WIN_INPUT_ONLY

Specifies that a window is an input only window; although it has other window properties, it cannot be drawn into.

Argument: No value
Default: Input/Ouptut Window
Procs: `create`
Objects: `Xv_Window`

WIN_IS_CLIENT_PANE

When used with `xv_create()`, the attribute has no value; using the attribute sets it to `TRUE`. When used, the window is considered to be an OPEN LOOK GUI application-specific pane which can override resources, such as fonts. Currently, the only packages that support this are `text` and `tty/term`.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`
Objects: `Xv_Window`

WIN_KBD_FOCUS

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

WIN_MAP

Indicates whether to map or unmap the window. Setting this attribute to `TRUE` maps the window without changing the stacking order. This does not imply *raised* (to the top of the window tree). Use `XV_SHOW` to raise and map a window.

Argument: `Bool`
Default: `FALSE`
Procs: `create`, `get`, `set`
Objects: `Xv_Window`
See Also: `WIN_FRONT, XV_SHOW`

WIN_MENU

This attribute is obsolete. The window manager now specifies the window's menu.

WIN_MESSAGE_DATA

Used to access the data portion of a window's client message.

Return Type: char
Procs: `get`
Objects: `Xv_Window`
See Also: 6.7.1, `WIN_MESSAGE_FORMAT`, `WIN_MESSAGE_TYPE`

XView Attributes

WIN_MESSAGE_FORMAT

Used to access the format portion of a window's client message.

Return Type: unsigned char
Procs: get
Objects: Xv_Window
See Also: 6.7.1, WIN_MESSAGE_DATA, WIN_MESSAGE_TYPE

WIN_MESSAGE_TYPE

Used to access the type portion of a window's client message.

Return Type: Atom
Procs: get
Objects: Xv_Window
See Also: 6.7.1, WIN_MESSAGE_FORMAT, WIN_MESSAGE_DATA

WIN_MOUSE_XY

Warps the mouse pointer to the specified position. Returns a static Rect * on xv_get.

Argument: int, int
Default: N/A
Procs: create, get, set
Objects: Xv_Window
See Also: 13.3

WIN_NO_CLIPPING

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

WIN_PARENT

Specifies the window's parent in the window tree. This attribute effectively reparents a window on set.

Argument: Xv_Window
Default: None
Procs: create, get, set
Objects: Xv_Window

WIN_PERCENT_HEIGHT

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

WIN_PERCENT_WIDTH

This attribute is for SunView compatibility. For more information, refer to the manual *Converting SunView Applications*.

WIN_RECT

Specifies the bounding box of a window. Returns a static Rect on xv_get.

Argument: Rect *
Default: NULL
Procs: create, get, set
Objects: Xv_Window

WIN_RETAINED

Hint to the server to maintain backing store for this window. The server may not honor this request.

Argument: Bool
Default: FALSE
Procs: create, get, set
Objects: Xv_Window
See Also: 5.5

WIN_RIGHT_OF

Causes a sibling window to be laid out just to the right of the window given as the value. Restricted to windows that share the same immediate parent. Does not set XV_Y.

Argument: Xv_Window
Default: Arbitrary
Procs: create,set
Objects: Xv_Window
See Also: WIN_BELOW

WIN_ROW_GAP

Specifies the gap between rows in the window.

Argument: int
Default: 0
Procs: create, get, set
Objects: Xv_Window

WIN_ROW_HEIGHT

Specifies the height of a row in the window.

Argument: int
Default: Font's default height
Procs: create, get, set
Objects: Xv_Window

WIN_ROWS

Specifies the window's height (including top and bottom margins) in rows. WIN_ROWS is not a valid xv_create() attribute for a Textsw. Issue a separate xv_set() call after the xv_create() in order to set this attribute on a Textsw.

Argument: int
Default: 34
Procs: create, get, set
Objects: Xv_Window

WIN_SAVE_UNDER

Provides a hint to the server about whether or not the screen area beneath a window should be saved while the window, such as a pop-up menu, is mapped. This is not the same as WIN_RETAINED.

Argument: Bool
Default: FALSE
Procs: set
Objects: Xv_Window

WIN_SCREEN_RECT

Returns the bounding box of the screen containing the window. Data points to per-process static storage.

Argument: `Rect *`
Default: Screen dependent
Procs: `get`
Objects: `Xv_Window`

WIN_SET_FOCUS

Sets the input focus to this window, if possible. The X protocol restricts unmapped windows from holding the input focus. The window must select for `KBD_USE` and `KBD_DONE` events.

Argument: No value
Default: None
Procs: `create`,`set`
Objects: `Xv_Window`

WIN_SOFT_FNKEY_LABELS

Assigns labels for the soft function keys. The value for the `WIN_SOFT_FNKEY_LABELS` is a string of 12 labels separated by newline characters "\n".

Argument: `char *`
Procs: `get,set`
Usage:

```
canvas = (Canvas) xv_create (frame,CANVAS,
                    CANVAS_X_PAINT_WINDOW,TRUE,
                    NULL);

xv_set(canvas_paint_window(canvas),
         WIN_SOFT_FNKEY_LABELS,"Red0reen0lue0aroon
         Orchid0iolet0agenta0oral0urquoise
         Yellow0rick0lack0,
         WIN_EVENT_PROC, my_Event_proc,
         NULL);
```

In the above example, whenever the canvas gets the input focus, the soft function key labels will be updated to "Red Green Blue ...". To display the soft function key panel, select the Function Keys item from the Workspace Utilities menu.

See Also: 6.12.1

WIN_TOP_LEVEL

Returns whether the window is the child of another window or is a frame.

Return Type: `Bool`
Default: N/A
Procs: `get`
Objects: `Xv_Window`

WIN_TOP_LEVEL_NO_DECOR

Controls whether this window is or is not controlled by the window manager. (This controls the `OverrideRedirect` flag.)

Argument: `Bool`
Default: `FALSE`
Procs: `create`,`get`,`set`
Objects: `Xv_Window`
See Also: 4.10

WIN_TRANSPARENT

Specifies that the window's background pixmap should be be transparent (set to none). For more information, on the background pixmap refer to Chapter 4, *Frames*, of the *XView Programming Manual*.

Argument: No value
Default: Not transparent
Procs: `create`
Objects: `Xv_Window`

WIN_VERTICAL_SCROLLBAR

This attribute is obsolete. Use `OPENWIN_VERTICAL_SCROLLBAR` instead. Specifies that scrollbar orientation is vertical.

Argument: `Scrollbar`
Default: None
Procs: `create`, `get`, `set`
Objects: `Xv_Window`

WIN_WINDOW_GRAVITY

Defines how the window should be repositioned if its parent is resized. See *<X11/X.h>* for legal values.

Argument: `int`
Default: `NorthWestGravity`
Procs: `create`, `get`, `set`
Objects: `Xv_Window`
See Also: 5.3, `WIN_BIT_GRAVITY`

WIN_X_COLOR_INDICES

Translates the logical indices of the window's colormap segment (from `0` to `size-1`) into the actual indices into the colormap used by the window.

Argument: `unsigned long *`
Procs: `get`
Objects: `Xv_Window`
See Also: 21.3.0.1

WIN_X_EVENT_MASK

Expects an X event mask. Acts in the same manner as `WIN_CONSUME_X_EVENT_MASK`; but instead of appending to the current input mask, replaces it with a new mask.

Argument: `unsigned long`
Default: Varies from package to package
Procs: `create`, `get`, `set`
See Also: `WIN_CONSUME_EVENTS`, `WIN_CONSUME_X_EVENT_MASK`

XV_APP_NAME

Sets the string to be used by XView as the application's name. (Currently this is only for the help package). This also allows for more than one help file while still displaying the same name in the spot help header window.

Argument: `char *`
Default: `argv[0]`
Procs: `set, get`
Objects: `Server`
Usage:

```
server = xv_init(XV_INIT_ARGS, argc, argv, 0);
          xv_set(server,
             XV_APP_NAME,mailtool, NULL);

/*  or (for localized app name) */
          xv_set(server,
             XV_APP_NAME,gettext(mailtool), NULL);
```

XV_AUTO_CREATE

Specifies whether to create an object not found by `xv_find()`.

Argument: `Bool`
Default: `TRUE`
Procs: `find`
Objects: `Cms, Font, Menu_item`
See Also: 25.9.2

XV_BOTTOM_MARGIN

Specifies the margin at the bottom of an object.

Argument: `int`
Default: Varies with object
Procs: `create`, `get`, `set`

XV_DEPTH

Specifies the pixel depth of the object.

Argument: `int`
Default: Function of the screen
Procs: `create`, `get`
Objects: `Xv_Window`

XV_DISPLAY

Returns the X `Display` data structure. The structure returned is maintained by the toolkit and should not be freed.

Argument: `Display *`
Default: Display structure for specified object
Procs: `get`
Objects: All `Drawable` objects and its subcalsses
See Also: 15.1

XV_ERROR_PROC
The application's XView error handler is called as a result of `xv_error`() being called.

Argument: `int (*`*error_proc*`)()`
Default: None
Procs: `xv_init`
Objects: N/A
Callback:

```
int
error_proc(object, avlist)
     Xv_object   object;
     Attr_avlist avlist;
```

See Also: 24.2, `XV_X_ERROR_PROC`

XV_FOCUS_ELEMENT
Set focus on the first (0) or last (-1) element in the pane.

Argument: `int`
Valid Values: {0, -1}
Procs: `set`
Objects: `Xv_Window`

XV_FONT
Specifies an object's font. Use only with `xv_create()` if you are setting a panel's font.

Argument: `Xv_Font`
Default: `lucida medium`
Procs: `create`, `get`, `set`
Only use `create` and `get` for a `FILE_CHOOSER`.
Object: Generally, all

XV_HEIGHT
Specifies the height of an object in pixels.

Argument: `int`
Default: N/A
Defaults to a pre-determined number of rows and columns for a `FILE_CHOOSER` object. Should not be set at create time for a File Chooser.
Procs: `create`, `get`, `set`
Objects: All
See Also: 15.4

XV_HELP_DATA
Specifies the help string used by the help package to display on-line help. The text string has the form `file:keyword`. The help package looks for the key `keyword` in the file *$HELPPATH/file.info*. In the special case for a `Textsw`, set `XV_HELP_DATA` on the `Textsw`'s view which you can get with the following call:

```
xv_get(textsw, OPENWIN_NTH_VIEW, NULL);
```

Argument: `char *`
Default: No help available notice
Procs: `create`, `get`, `set`
Objects: All
See Also: 23.1

XView Attributes

XV_INIT_ARGC_PTR_ARGV

Interprets command-line `args`. Strips generic toolkit command-line arguments out of `argv` and decrements `argc` accordingly.

Argument 1: `int *`
Argument 2: `char **`
Default: None
Procs: `xv_init`
Objects: N/A
Usage:

```
main(argc, argv
    int   argc;
    char  **argv;
 {
    server = xv_init (XV_INIT_ARGC_PTR_ARGV,
                         &argc, argv, NULL);
 }
```

See Also: 2.6.1, `XV_INIT_ARGS`

XV_INIT_ARGS

Interprets command-line arguments. Does not strip generic toolkit command-line arguments out of `argv`.

Argument 1: `int`
Argument 2: `char **`
Default: None
Procs: `xv_init`
Objects: `N/A`
Usage:

```
main(argc, argv
    int    argc;
    char   **argv;
 {
    server = xv_init (XV_INIT_ARGS, argc, argv, NULL);
 }
```

See Also: 2.6.1, `XV_INIT_ARGC_PTR_ARGV`

XV_INSTANCE_NAME

The attribute `XV_INSTANCE_NAME` is used to associate an instance name with an XView object. The instance name is used to construct the resource name used by the resource manager to perform lookups. The resource name is constructed by concatenating the instance names of all objects in the current object's lineage, starting with the name of the application or whatever was passed in with the -*name* command-line option, ending with the XView attribute name. The XView attribute name remains in lowercase. `XV_INSTANCE_NAME` is normally used with `XV_USE_DB`.

Argument: `char *`
Default: `NULL`
Procs: `create`, `set`

Usage:

```
Frame       frame;
Panel       panel;
frame = (Frame)xv_create(NULL, FRAME,
                        XV_INSTANCE_NAME, "base_frame",
                        NULL);
panel = (Panel)xv_create(frame, PANEL,
                        XV_INSTANCE_NAME, "panel",
                        XV_USE_DB,
                            XV_WIDTH, 100,
                            XV_HEIGHT, 200,
                            NULL,
                        NULL);
```

Above, assume the name of the application is `app`, the resource names constructed for lookup of the width and height of the panel are:

```
app.base_frame.panel.xv_width
app.base_frame.panel.xv_height
```

Entries in the resource manager could look like:

```
app.base_frame.panel.xv_width:400
app.base_frame.panel.xv_height:500
```

If these entries were not present in the resource manager, the width and height of the panel would take the default values of 100 and 200, respectively.

See Also: 22.3.2.3, 17.3, XV_USE_DB

XV_KEY_DATA

Stores a 32-bit data value on an object. You may set multiple XV_KEY_DATA values on objects by using different keys. The key specified should be a unique number (see `xv_unique_key()` in Section 3, *Procedures and Macros*.)

Argument 1: `int`
Argument 2: `XV_opaque`
Default: None
Procs: `create`, `get`, `set`
Objects: All
Usage:

```
xv_create(panel, PANEL_BUTTON,
      PANEL_BUTTON_LABEL,    "Push Me"
      XV_KEY_DATA,           PANEL_ITEM_KEY,    "text",
      PANEL_NOTIFY_PROC,     my_notify_proc
NULL);
```

See Also: 7.19.1

XV_KEY_DATA_REMOVE

Causes the data associated with the specified key to be removed. If an XV_KEY_DATA_REMOVE_PROC is defined, it will be called.

Argument: `int`
Default: None
Procs: `set`
Objects: All
See Also: 7.19.1, XV_KEY_DATA, XV_KEY_DATA_REMOVE_PROC

XV_KEY_DATA_REMOVE_PROC

Names the function that is called whenever the object that has key data attached to it is destroyed. This function should free the data associated with the key.

Argument 1: `int`
Argument 2: `void(*`*xv_key_data_remove_proc*`)()`
Default: None
Procs: `create`, `get`, `set`
Objects All
Callback:

```
void
xv_key_data_remove_proc(object, key, data)
        Xv_object     object;
        int           key;
        caddr_t       data;
```

See Also: 7.19.1

XV_LABEL

Specifies a frame's header label or an icon's label or simply associates a name to an object. XView copies the strings on set.

Argument: `char *`
Default: `NULL`
For a `FILE_CHOOSER`, defaults to the string "Open", "Save" or "Save As", depending on the value of the `FILE_CHOOSER_TYPE` attribute. Note that the the proper format is `Application: Popupname`.
Procs: `create`, `get`, `set`
Objects: All
See Also: 4.2.2

XV_LC_BASIC_LOCALE

Specifies the basic locale category, which sets the country of the user interface. This XView attribute can only be used in `xv_init()`, but can be queried on any XView object which is a subclass of window or server class via `xv_get()`. The value that is set using these attributes must be a valid locale name in the system. Use in situations where you want to force a program to operate in a certain locale. For example, if a program only works in French, then the locale attribute can be set so that it cannot be switched to another language.
Warning: This attribute should only be used for localization operations.

Argument: `char *`
Default: "C"
Procs: `xv_init()`, `get`
Usage:

```
xv_init(...,
       XV_LC_BASIC_LOCALE, "fr",
       XV_LC_DISPLAY_LANG, "C",
       NULL);
```

See Also: 22.1.4, `XV_LOCALE_DIR`, Chapter 22, *Internationalization*, in *XView Programming Manual.*

XV_LC_DISPLAY_LANG

Specifies the display language locale category, which sets the language in which labels, messages, menu items, and help text are displayed. This XView attribute can only be used in `xv_init()`, but can be queried on any XView object which is a subclass of window or server class via `xv_get()`. The value that is set using these attributes must be a valid locale name in the system. Use in situations where you want to force a program to operate in a certain locale. For example, if a program only works in French, then the locale attribute can be set so that it cannot be switched to another language.
Warning: This attribute should only be used for localization operations.

Argument: `char *`
Default: "C"
Procs: `xv_init()`, `get`
See Also: 22.1.4, `XV_LOCALE_DIR`, Chapter 22, *Internationalization*

XV_LC_INPUT_LANG

Specifies the input language locale category, which sets the language used for keyboard input. This XView attribute can only be used in `xv_init()`, but can be queried on any XView object which is a subclass of window or server class via `xv_get()`. The value that is set using these attributes must be a valid locale name in the system. Use in situations where you want to force a program to operate in a certain locale. For example, if a program only works in French, then the locale attribute can be set so that it cannot be switched to another language.
Warning: This attribute should only be used for localization operations.

Argument: `char *`
Default: "C"
Procs: `xv_init()`, `get`
See Also: 22.1.4, `XV_LOCALE_DIR`, Chapter 22, *Internationalization*

XV_LC_NUMERIC

Specifies the numeric locale category, which defines the language used to format numeric quantities. This XView attribute can only be used in `xv_init()`, but can be queried on any XView object which is a subclass of window or server class via `xv_get()`. The value that is set using these attributes must be a valid locale name in the system. Use in situations where you want to force a program to operate in a certain locale. For example, if a program only works in French, then the locale attribute can be set so that it cannot be switched to another language.
Warning: This attribute should only be used for localization operations.

Argument: `char *`
Default: "C"
Procs: `xv_init()`, `get`
See Also: 22.1.4, `XV_LOCALE_DIR`, Chapter 22, *Internationalization*

XV_LC_TIME_FORMAT

Specifies the time format locale category, which defines the language used to format time and date. This XView attribute can only be used in `xv_init()`, but can be queried on any XView object which is a subclass of window or server class via `xv_get()`. The value that is set using these attributes must be a valid locale name in the system. Use in situations where you want to force a program to operate in a certain locale. For example, if a program only works in French, then the locale attribute can be set so that it cannot be switched to another language.
Warning: This attribute should only be used for localization operations.

Argument: `char *`
Default: "C"
Procs: `xv_init()`, `get`
See Also: 22.1.4, `XV_LOCALE_DIR`, Chapter 22, *Internationalization*

XV_LEFT_MARGIN

Specifies the margin at the left of the object.

Argument: `int`
Default: Varies with object
Procs: `create`, `get`, `set`

XV_LOCALE_DIR

This attribute specifies the location of the application's locale specific files. At present, these include the `app-defaults` and `LC_MESSAGES` directories.

The directory structure referenced by `XV_LOCALE_DIR` is:

```
<XV_LOCALE_DIR>/<locale>/LC_MESSAGES
<XV_LOCALE_DIR>/<locale>/app-defaults
```

`<locale>` is expanded differently depending on which internationalization feature is using `XV_LOCALE_DIR`.

Argument: `char *`
Default: *$OPENWINHOME/lib/locale*
Procs: `xv_init()`
See Also: 22.3.2.1, 22.2.1.6, 22.1.2.1

XV_MARGIN

Specifies the offset from the border of this object.

Argument: `int`
Default: Varies with object
Procs: `create`, `get`, `set`

XV_NAME

Specifies an optional name for an object. In some cases, this attribute is used internally by packages. For instance, the `SERVER` package sets the display connection name using this attribute.

Argument: `char *`
Default: Varies with object
Procs: `create`, `get`, `set`
Objects: All
See Also: 15.3.1

XV_OWNER

Returns the object's owner. The object returned varies among packages.

Return Type: `Xv_opaque`
Procs: `get`
Objects: All

XV_RECT

Specifies the object's bounding box; that is, the smallest rectangle that contains the object. The `Rect*` returned points to a per-process static storage and thus should not be freed.

Argument: `Rect *`
Default: None. Should not be set at create time for a File Chooser.
Procs: `create`, `get`, `set`
Objects: `Window` and its subclasses
See Also: 25.2.1

XV_RIGHT_MARGIN

Specifies the margin at the right of the object.

Argument: `int`
Default: Varies with object
Procs: `create`, `get`, `set`

XV_ROOT

Returns the root window for an object. (Object must be a window based.)

Argument: `Xv_object`
Default: Root window for default screen
Procs: `get`
Objects: `Window` and its subclasses

XV_SCREEN

Returns the screen object associated with this object.

Argument: `Xv_screen`
Default: Values for specified screen
Procs: `get`
Objects: `Window` and its subclasses
See Also: 15.2

XV_SHOW

Causes the object to be displayed or undisplayed. If object is a window, brings object to the top.

Argument: `Bool`
Default: Varies
For a `FILE_CHOOSER` object, it is not recommended to set `XV_SHOW` to `TRUE` at create time.
Procs: `create`, `get`, `set`
Objects: `Generic`
See Also: 4.3.1, `WIN_MAP, WIN_FRONT`

XV_TOP_MARGIN

Specifies the margin at the top of the object.

Argument: `int`
Default: Varies with object
Procs: `create`, `get`, `set`

XV_TYPE

Returns the package that belongs to an object.

Argument: `Xv_pkg *`
Default: N/A
Procs: `get`
Objects: `All`
See Also: 24.4.2

XV_USAGE_PROC

An application-defined routine that is called when an application is started with the *-help* option. Typically used to print a list of valid command-line options. This routine should not return. Applications that add additional command-line options to those already provided by XView should register an `XV_USAGE_PROC`. From within the `XV_USAGE_PROC`, the application should first print the application defined options and then call `xv_usage()` to allow XView to print the default command-line options.

Argument: `void (*`*xv_usage_proc*`)()`
Default: `xv_usage()` (XView defined usage procedure)
Procs: `xv_init`
Usage:

```
application_usage_proc(name)
char *name
{
    printf("...");
            xv_usage(name);
            exit(1);
}
main()
{
    xv_init(... XV_USAGE_PROC,
                    application_usage_proc, ...);
}
```

Callback:

```
void
usage_proc(application_name)
    char *application_name
```

See Also: 2.6.1

XV_USE_DB

The attribute `XV_USE_DB` can be used to specify a set of attributes that are to be searched in the X11 resource manager database. `XV_USE_DB` takes a `NULL`-terminated list of attribute-value pairs as its values. During program execution, each attribute in this `NULL`-terminated list of attributes is looked up in the X11 resource manager database. If the attribute is not found in the database, then the value specified in the attribute-value pair is used as the default value.

The list of customizable attributes is shown below:

```
CANVAS_HEIGHT                       CANVAS_MIN_PAINT_HEIGHT
CANVAS_MIN_PAINT_WIDTH              CANVAS_WIDTH
PANEL_CHOICE_NCOLS                  PANEL_CHOICE_NROWS
PANEL_DROP_HEIGHT                   PANEL_DROP_WIDTH
PANEL_EXTRA_PAINT_HEIGHT            PANEL_EXTRA_PAINT_WIDTH
PANEL_GAUGE_WIDTH                   PANEL_ITEM_X
PANEL_ITEM_X_GAP                    PANEL_ITEM_Y
PANEL_ITEM_Y_GAP                    PANEL_JUMP_DELTA
PANEL_LABEL_WIDTH                   PANEL_LABEL_X
PANEL_LABEL_Y                       PANEL_LIST_DISPLAY_ROWS
PANEL_LIST_ROW_HEIGHT               PANEL_LIST_WIDTH
PANEL_MAX_VALUE                     PANEL_MIN_VALUE
PANEL_NEXT_COL                      PANEL_NEXT_ROW
PANEL_SLIDER_WIDTH                  PANEL_TICKS
PANEL_VALUE_DISPLAY_LENGTH          PANEL_VALUE_DISPLAY_WIDTH
PANEL_VALUE_STORED_LENGTH           PANEL_VALUE_X
```

```
PANEL_VALUE_Y
WIN_COLUMNS                          WIN_DESIRED_HEIGHT
WIN_DESIRED_WIDTH                    WIN_ROWS
XV_HEIGHT                            XV_WIDTH
XV_X                                 XV_Y
```

Argument: NULL-terminated attribute-value list
Procs: create, set
Usage:

```
Frame   frame;

frame = (Frame)xv_create(NULL, FRAME,
                        XV_INSTANCE_NAME, "base_frame",
                        XV_USE_DB,
                            XV_WIDTH, 100,
                            XV_HEIGHT, 200,
                            NULL,
                        NULL);
```

In this example, the value of the attributes XV_WIDTH and XV_HEIGHT for the frame object is searched in the X11 resource manager database. If not found, XV_WIDTH and XV_HEIGHT will take the default values of 100 and 200 respectively. See XV_INSTANCE_NAME for information regarding the actual resource name used for the search.
For more detailed information, refer to the *Xlib Programming Manual*.

See Also: 22.3.2.2, 17.3, XV_INSTANCE_NAME, Chapter 22, *Internationalization*

XV_USE_LOCALE

This attribute enables the internationalization features in XView. When this attribute is FALSE (default), internationalization features in XView are turned off. All internationalized applications must set this attribute to be TRUE. This attribute is only valid in the xv_init() function call. The format is as follows:

Argument: Bool
Default: FALSE
Procs: xv_init
Usage:

```
xv_init(XV_USE_LOCALE, TRUE,...);
```

See Also: 22.1.2

XV_VISUAL

This attribute specifies the exact visual that will be used in the creation of the window or colormap segment. The value is a pointer to a visual structure that one would normally get from XMatchVisualInfo or XGetVisualInfo. This attribute applies only to WINDOW and cms objects.

Argument: Visual *
Default: Varies with server
Procs: create, get
See Also: 21.8

XView Attributes

XV_VISUAL_CLASS

This attribute specifies the class of the visual that will be used in the creation of the window or colormap segment. The value should be one of the following:

```
StaticGray
GrayScale
StaticColor
PseudoColor
TrueColor
DirectColor
```

If the server that the window is being created with does not support the visual class that is specified by this attribute, the library will use the default visual instead. This attribute applies only to `WINDOW` and `CMS` objects.

Argument:	`int`
Default:	Varies with server (usually PseudoColor)
Procs:	`create,get`
See Also:	21.8

XV_WIDTH

Specifies the width of the object in pixels.

Argument:	`int`
Default:	None Defaults to a pre-determined number of rows and columns for a `FILE_CHOOSER` object. Should not be set at create time for a File Chooser.
Procs:	`create`, `get`, `set`
Objects:	`Generic`
See Also:	15.4

XV_X

Specifies the `x` position of the object relative to its parent.

Argument:	`int`
Default:	Varies with object
Procs:	`create`, `get`, `set`
See Also:	7.4.2

XV_X_ERROR_PROC

Specifies the application's Xlib error handler (different from `XV_ERROR_PROC`). This function should return `XV_OK` if the error is to be ignored by XView. Return `XV_ERROR` if XView is to process the X error and exit.

Argument:	`int (*`*x_error_proc*`)()`
Default:	None
Procs:	`xv_init`
Callback:	

```
int
x_error_proc(display, event)
   Display     *display;
   XErrorEvent *event;
```

See Also:	24.3, `XV_ERROR_PROC`

XV_XID

Returns the XID of the specified object such as the pixmap associated with a `Server_ image` or the X Window associated with a canvas paint window.

Argument: `int`
Default: N/A
Procs: `get`
Objects: `Drawable` and its subclasses
See Also: 16.1

XV_Y

Specifies the y position of the object relative to its parent.

Argument: `int`
Default: Varies with object
Procs: `create`, `get`, `set`
Objects: `Generic`
See Also: 7.4.2

XView Attributes