API Summaries

This appendix is divided into two sections: attributes and functions. Each section describes how to use the APIs. Use the information in this appendix as an addendum to the *XView Reference Manual* by O'Reilly & Associates.

Attributes

This section lists attributes alphabetically and summarizes each attribute's purpose, type, default, and procs.

CANVAS_IM_PREEDIT_FRAME

Provides a pointer to the preedit popup window's frame.

Type: `Frame`
Default: none
Procs: `get`

Only the following attributes can be set with predictable results for the frame object returned by `CANVAS_IM_PREEDIT_FRAME`: `XV_LABEL`, `XV_SHOW`, `XV_X`, `XV_Y`, `XV_WIDTH`, `XV_HEIGHT`.

CURSOR_STRING_WCS

Creates a text drag and drop cursor. If the wide character string exceeds three characters (not bytes), then only the first 3 characters are displayed, and "the more arrow" is shown inside the cursor.

Type: `wchar_t *`
Default: none
Procs: `create, get`

FILE_CHOOSER_APP_DIR_WCS

Adds an application-specific pathname to the Go To history menu's fixed space.

Type: `wchar_t *` pair
Default: none
Procs: `create, set`

FILE_CHOOSER_CUSTOMIZE_OPEN_WCS

Allows the client to use the Open window in certain contexts: Insert, Choose, Import.

Type: `wchar_t *, wchar_t *, enum`
Default: none
Procs: `create`

FILE_CHOOSER_DIRECTORY_WCS

Specifies the current working directory that is displayed in Open, Save, and Save As windows.

Type: `wchar_t *`
Default: Current working directory name
Procs: `create, get, set`

FILE_CHOOSER_DOC_NAME_WCS

In a Save window, specifies the default document name ("Untitled1").
In a Save As window, specifies the current document name.

Type: `wchar_t *`
Default: Untitled1 (in a Save window)
Procs: `create, get, set`

FILE_CHOOSER_FILTER_STRING_WCS

Sets or gets an ex-like regular expression string. The string's files are filtered before being displayed in the list. If not filter string is specified, all entries are assumed to match. "`..`" is always assumed to match.

Type: `wchar_t *`
Default: none
Procs: `create`, `get`, `set`

Caution – Do not use this attribute for multibyte characters in filenames. Results are unpredictable. This limitation will be removed in future releases.

FILE_CHOOSER_NOTIFY_FUNC_WCS

Invokes a callback when the user selects a file from an Open, Save, or Save As window. Expected return values are `XV_OK` or `XV_ERROR`. The form of the callback depends on whether the `FILE_CHOOSER_TYPE` attribute is set to `FILE_CHOOSER_OPEN`, `FILE_CHOOSER_SAVE`, or `FILE_CHOOSER_SAVE_AS`.

Type: Function pointer
Default: NULL
Procs: `create`, `get`, `set`

FILE_CHOOSER_WCHAR_NOTIFY

Specifies that the callbacks registered by `FILE_LIST_CHANGE_DIR_FUNC`, `FILE_LIST_FILTER_FUNC`, and `FILE_LIST_COMPARE_FUNC` receive wide character strings, not character strings.

Type: `Boolean`
Default: FALSE
Procs: `create`, `get`, `set`

FILE_LIST_DIRECTORY_WCS

Sets or gets the directory currently displayed in the list.
If `FILE_LIST_DIRECTORY` is set to `NULL`, the directory will be empty.

Type: `wchar_t *`
Default: Name of the current working directory
Procs: `create`, `get`, `set`

FILE_LIST_DOTDOT_STRING_WCS

Allows the client to modify the string used by the file list package to denote the "`..`" entry.

Type: `wchar_t *`
Default: `"...Go up one folder..."`
Procs: `create, get, set`

FILE_LIST_FILTER_STRING_WCS

Sets or gets an ex-like regular expression string. The string's files are filtered before being displayed in the list. If not filter string is specified, all entries are assumed to match. "`..`" is always assumed to match.

Type: `wchar_t *`
Default: none
Procs: `create, get, set`

Caution – Do not use this attribute for multibyte characters in filenames. Results are unpredictable. This limitation will be removed in future releases.

FILE_LIST_WCHAR_NOTIFY

Specifies that the callbacks registered by `FILE_LIST_CHANGE_DIR_FUNC`, `FILE_LIST_FILTER_FUNC`, and `FILE_LIST_COMPARE_FUNC` receive wide character strings, not character strings.

Type: `Boolean`
Default: FALSE
Procs: `create, get, set`

FONT_CHAR_HEIGHT_WC

Returns the height (`int`) of a specified wide character (`wchar_t`).

Type: `int`
Default: none
Procs: `get`

FONT_CHAR_WIDTH_WC

Returns the width (`int`) of a specified wide character (`wchar_t`).

Type: `int`
Default: none
Procs: `get`

FONT_COLUMN_WIDTH

Returns the default screen column width in pixels, given a specified XView font set object.

Type: `int`
Default: none
Procs: `get`

FONT_LOCALE

Specifies the locale for a font set object.

Type: `char *`
Default the basic locale
Procs: `create`, `find`

FONT_NAMES

Specifies the XLFD font names of the fonts to be used to construct the font set object.

Type: `char **`
Default: none
Procs: `create`, `find`, `get`

FONT_SET_ID

Returns the X font set, when given an XView font set object.

Type: `XFontSet`
Defaults: none
Procs: `get`

FONT_SET_SPECIFIER

Specifies the name of a font set. The font set definition database will be queried with the specified value.

Type: `char *`
Default: none
Procs: `create`, `find`, `get`

FONT_STRING_DIMS_WCS

Given a wide character string and the address of a `Font_string_dims` structure, `xv_get()` fills in and returns a pointer to the structure describing the pixel dimensions of the string.

Type: `Font_string_dims *`
Default: none
Procs: `get`

FRAME_LABEL_WCS

Specifies the label used in the window manager's titlebar for the frame. This is the wide character version of `FRAME_LABEL`.

Type: `wchar_t *`
Default: `NULL`
Procs: `create`, `get`, `set`

FRAME_LEFT_FOOTER_WCS

Specifies the left-justified footer. This is the wide character version of `FRAME_LEFT_FOOTER`.

Type: `wchar_t *`
Default: `NULL`
Procs: `create`, `get`, `set`

FRAME_RIGHT_FOOTER_WCS

Specifies the right-justified footer. This is the wide character version of `FRAME_RIGHT_FOOTER`.

Type: `wchar_t *`
Default: `NULL`
Procs: `create`, `get`, `set`

HISTORY_ADD_FIXED_ENTRY_WCS

Adds a string to the fixed space in the list. A fixed string is added to the bottom of the fixed space in the list. Passing a label of NULL adds a blank menu item.

Type: wchar_t * pair
Default: none
Procs: create, set

HISTORY_ADD_ROLLING_ENTRY_WCS

Adds a string to the rolling space. Strings in the rolling space and stacked and "roll off" after a specified number of strings are added.

Type: wchar_t *
Default: none
Procs: create, set

HISTORY_LABEL_WCS

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

Type: Int pair
Default: none
Procs: get

HISTORY_NOTIFY_PROC_WCS

Invokes a callback-based on the user's selection from the Go To history menu.

Type: function pointer
Default: none
Procs: create, set

HISTORY_VALUE_WCS

Returns the VALUE from a specified space (defined as HISTORY_FIXED or HISTORY_ROLLING) for a specified row number. If the row does not exist, NULL is returned.

Type: `Int pair`
Default: `none`
Procs: `get`

ICON_LABEL_WCS

Specifies the icon label used for the frame. This is the wide character string version of `ICON_LABEL`.

Type: `wchar_t *`
Default: `NULL`
Procs: `create, get, set`

ICON_TRANSPARENT_LABEL_WCS

Draws the given wide character string into an icon. It does not affect any other pixels in the bounding box. This is the wide character version of `ICON_TRANSPARENT_LABEL`.

Type: `wchar_t *`
Default: `NULL`
Procs: `create, get, set`

MENU_ACCELERATOR_WCS

Sets an accelerator on a menu item when used in a `create` or `set` call. If an accelerator is changed with `xv_set()`, `FRAME_MENUS` must be set again before the accelerator takes effect. XView copies the wide character accelerator string, and `get` returns the accelerator string. Do not modify the return string.

Type: `wchar_t *`
Default: none
Procs: `create, get, set`

MENU_ACTION_ACCELERATOR_WCS

Creates a menu item with a given label, notify proc, and accelerator. If an accelerator is changed with `xv_set()`, `FRAME_MENUS` must be set again before the accelerator takes effect. XView copies the wide character accelerator string, but not the wide character label string.

Type: `wchar_t *, void (*)(), wchar_t *`
Default: none
Procs: `create, set`

MENU_ACTION_ITEM_WCS

Provides a shortcut for creating or modifying a menu item and associating it with a notify procedure. It takes two values: a wide character string and a notify procedure. This is the wide character version of `MENU_ACTION_ITEM`.

Type: `wchar_t *,(void *) ()`
Default: Not applicable
Procs: `create, set`

MENU_GEN_PIN_WINDOW_WCS

Uses a command window to create the pin window for a menu. The contents are based on the static (not dynamic) menu contents. The frame is the parent frame; the name in wide characters is the name of the pin window. All menu items must have a notify procedure; `MENU_NOTIFY_PROC` for the menu itself is ignored. This is the wide character version of `MENU_GEN_PIN_WINDOW`.

Type: `Frame, wchar_t *`
Default: no pin window
Procs: `create, set`

MENU_GEN_PROC_ITEM_WCS

Defines the generate procedure and wide character text string for a menu item. This is the wide character version of `MENU_GEN_PROC_ITEM`.

Type: `wchar_t *,(* Menu) ()`
Default: `NULL`
Procs: `create, set`

MENU_GEN_PULLRIGHT_ITEM_WCS

Provides a shortcut for creating a menu item (or a menu item's submenu) and associating it with a pull-right menu generate procedure. This is the wide character version of `MENU_GEN_PULLRIGHT_ITEM`.

Type: `wchar_t *,(* Menu) ()`
Default: none
Procs: `create, set`

MENU_PULLRIGHT_ITEM_WCS

Creates a string menu item with pull-right submenu. This is the wide character version of `MENU_PULLRIGHT_ITEM`.

Type: `wchar_t *, Menu`
Default: Not applicable
Procs: `create, set`

MENU_STRING_WCS

Sets or gets the string of a given menu item. This is the wide character version of `MENU_STRING`.

Type: `wchar_t *`
Default: `NULL`
Procs: `create, get, set`

MENU_STRING_ITEM_WCS

Defines the text string and value for a menu item. This is the wide character version of `MENU_STRING_ITEM`.

Type: `wchar_t *, Xv_opaque`
Default: `NULL`
Procs: `create, set`

MENU_STRINGS_WCS

Sets the strings for multiple menu items. This is the wide character version of `MENU_STRINGS`.

Type: list of `wchar_t *`
Default: Not applicable
Procs: `create, set`

MENU_STRINGS_AND_ACCELERATORS_WCS

Creates menu items with the given labels and accelerators. If an accelerator is changed with `xv_set()`, `FRAME_MENUS` must be set again before the accelerator takes effect. XView copies the wide character accelerator strings, but not the wide character label strings.

Type: list of `<label (wchar_t *), accelerator (wchar_t *)> pairs, terminated by NULL`
Default: none
Procs: `create, set`

MENU_TITLE_ITEM_WCS

Set the title string of a menu. Can only be used with menus that do not originate from pull-right items or pulldown menu buttons. This is the wide character version of `MENU_TITLE_ITEM`.

Type: `wchar_t *`
Default: No title
Procs: `create, set`

NOTICE_BUTTON_WCS

Specifies a wide character string to be displayed in a button and a value to use if the button is selected. This is the wide character version of `NOTICE_BUTTON`.

Type: `wchar_t *, int`
Default: none
Procs: `create, set`

NOTICE_BUTTON_NO_WCS

Specifies a wide character string associated with the NO (cancel) button of a notice. The value returned if this button is selected is `NOTICE_NO`. This is the wide character version of `NOTICE_BUTTON_NO`.

Type: `wchar_t *`
Default: none
Procs: `create, set`

NOTICE_BUTTON_YES_WCS

Specifies a wide character string associated with the YES (confirm) button. The value returned if this button is selected is NOTICE_YES. This is the wide character version of NOTICE_BUTTON_YES.

Type: wchar_t *
Default: none
Procs: create, set

NOTICE_MESSAGE_STRING_WCS

Specifies the text to print in a notice. The value of this attribute is a wide character string. This is the wide character version of NOTICE_MESSAGE_STRING.

Type: wchar_t *
Default: NULL
Procs: create, set

NOTICE_MESSAGE_STRINGS_WCS

Specifies the text to print in a notice. The value of this attribute is a NULL- terminated list of wide character strings. This is the wide character version of NOTICE_MESSAGE_STRINGS.

Type: list of wchar_t *
Default: NULL
Procs: create, set

NOTICE_MESSAGE_STRINGS_ARRAY_PTR_WCS

Specifies the text to print in a notice. The value of this attribute is a variable pointing to a NULL-terminated array of wide character strings. This is the wide character version of NOTICE_MESSAGE_STRINGS_ARRAY_PTR.

Type: wchar_t **
Default: NULL
Procs: create, set

PANEL_CHOICE_STRING_WCS

Sets the wide character string for a choice item. The first argument to the attribute is the index of a choice item, the second argument is the wide character string. This is the wide character version of PANEL_CHOICE_STRING.

Type: int, wchar_t *
Default: NULL
Procs: create, get, set
Objects: Panel_choice_item

PANEL_CHOICE_STRINGS_WCS

The value of this attribute is a NULL terminated list of wide character strings. Each string will be the label of a choice. This is the wide character version of PANEL_CHOICE_STRINGS.

Type: list of wchar_t *
Default: NULL
Procs: create, get, set
Objects: Panel_choice_item

PANEL_ITEM_IC_ACTIVE

Specifies whether a panel text item should allow or disallow input method conversion. The default value is set according to the value of the WIN_USE_IM attribute of the parent window.

Type: Boolean
Default: TRUE
Procs: create, get, set
Objects: Panel_text_item, Panel_slider_item, Panel_list_item, Panel_numeric_text_item

PANEL_LABEL_STRING_WCS

Specifies the wide character string for an item's label. This is the wide character version of PANEL_LABEL_STRING.

Type: wchar_t *
Default: NULL
Procs: create, get, set

PANEL_LIST_INSERT_STRINGS_WCS

Inserts the specified wide character strings into the scrolling list before the specified row. The first argument to the attribute is a row number, and the second argument is a pointer to a NULL-terminated array of wide character strings. This is the wide character version of PANEL_LIST_INSERT_STRINGS.

Type: int, wchar_t **
Default: Not applicable
Procs: create, set
Objects: Panel_list_item

PANEL_LIST_ROW_VALUES_WCS

Gets row values from or sets row values in PANEL_LIST. Takes the row number, a pointer to a Panel_list_row_values_wcs structure, and a count of the number of rows in the array. Panel_list_row_values_wcs is defined as:

```
typedef struct {
    wchar_t *string_wcs;
    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_wcs;
```

Type: int, struct pointer, int
Default: none
Procs: create, get, set

PANEL_LIST_STRING_WCS

Assign the specified wide character string (second argument) to the specified row (third argument). xv_get() will return a pointer to the string in the specified row. This is the wide character version of PANEL_LIST_STRING.

Type: int, wchar_t *
Default: NULL
Procs: create, get, set
Objects: Panel_list_item

PANEL_LIST_STRINGS_WCS

Similar to its companion attribute, PANEL_LIST_STRING, except that it takes a NULL-terminated list of wide character strings as its value. This is the wide character version of PANEL_LIST_STRINGS.

Type: list of wchar_t *
Return type: wchar_t *
Default: NULL
Procs: create, get, set
Objects: Panel_list_item

PANEL_LIST_TITLE_WCS

The title of a scrolling list as a wide character string. This is the wide character version of PANEL_LIST_TITLE.

Type: wchar_t *
Default: NULL
Procs: create, get, set
Objects: Panel_list_item

PANEL_MASK_CHAR_WC

When the user enters a character, the character specified as the value of PANEL_MASK_CHAR_WC will be displayed in place of each character the user has typed. Use the space character for no character echo (caret does not advance). Use the NULL character to disable masking. If PANEL_MASK_CHAR_WC or PANEL_MASK_CHAR is specified for a particular panel text item, then preedit input will be disabled for this panel text item. This is the wide character version of PANEL_MASK_CHAR.

Type: wchar_t
Default: none
Procs: create, get, set
Objects: Panel_text_item, Panel_numeric_text_item

PANEL_MAX_TICK_STRING_WCS

Specifies the wide character string which appears underneath the maximum tick mark on horizontal sliders, or to the right of the maximum tick mark on vertical sliders. PANEL_MAX_TICK_STRING_WCS is ignored if PANEL_TICKS is 0. The width of the slider can be adjusted to insure that there is enough space to accommodate both the minimum and maximum tick strings.

Type: wchar_t *
Default: NULL
Procs: create, get, set
Objects: Panel_slider_item, Panel_gauge_item

PANEL_MAX_VALUE_STRING_WCS

Maximum value string in wide characters 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. This is the wide character version of PANEL_MAX_VALUE_STRING.

Type: wchar_t *
Default: NULL
Procs: create, get, set
Objects: Panel_slider_item

PANEL_MIN_TICK_STRING_WCS

Wide character string that appears underneath the minimum tick mark on vertical sliders. PANEL_MIN_TICK_STRING_WCS is ignored if PANEL_TICKS is 0. The width of the slider can be adjusted to insure that there is enough room to accommodate both the minimum and maximum tick strings. This is the wide character version of PANEL_MIN_TICK_STRING.

Type: wchar_t *
Default: NULL
Procs: create, get, set
Objects: Panel_slider_item, Panel_gauge_item

PANEL_MIN_VALUE_STRING_WCS

Minimum value wide character 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. This is the wide character version of PANEL_MIN_VALUE_STRING.

Type: wchar_t *
Default: NULL
Procs: create, get, set
Objects: Panel_slider_item

PANEL_NOTIFY_PROC_WCS

Procedure to call when a scrolling list is activated. This is the wide character version of PANEL_NOTIFY_PROC.

Argument: See below
Default: none
Procs: create, get, set
Objects: Panel_item

Callback:

```
int
notify_proc_name(list_item, wide_char_string, client_data, op,
        event, row)
    Panel_item list_item;     /* PANEL_LIST item */
    wchar_t    *wide_char_string; /* string associated with row */
    Xv_opaque  client_data;   /* Client data of the row*/
    Panel_list_op   op;       /* Select, validate or
             * delete operation */
    Event   *event;
    int     row;        /* Scroll list row number*/
```

PANEL_NOTIFY_STRING_WCS

The value is a wide character string that triggers notification when typed into a text item. Applies only when PANEL_NOTIFY_LEVEL is PANEL_SPECIFIED. This is the wide character version of PANEL_NOTIFY_STRING.

Type: `wchar_t *`
Default: \r\t\n (carriage return, tab, newline)
Procs: `create, get, set`
Objects: `Panel_multiline_text_item, Panel_numeric_text_item, Panel_text_item`

PANEL_VALUE_WCS

Indicates the current value of a panel item as a wide character string if the value is a string. This is the wide character version of `PANEL_VALUE`.

Type: `wchar_t *`
Default: `NULL`
Procs: `create, get, set`
Objects: `Panel_choice_item, Panel_gauge_item, Panel_list_item, Panel_multiline_text_item, Panel_numeric_text_item Panel_slider_item, Panel_text_item`

PANEL_VALUE_DISPLAY_LENGTH

Maximum number of columns to display in a text item. The length of the value display cannot be less than the combined width of the left and right "more text" buttons. Note that this definition is different from the XView 3.1 environment.

Type: `int`
Default: 80
Procs: `create, get, set`
Objects: `Panel_list_item, Panel_slider_item, Panel_numeric_text_item, Panel_text_item`

PANEL_VALUE_STORED_LENGTH

Maximum number of bytes to be stored in the value string for text items. The panel value of a panel text item will be converted to multibyte to check whether it has the reached the limit set by this attribute.

Type: `int`
Default: 80
Procs: `create, get, set`
Objects: `Panel_list_item,`

```
Panel_multiline_text_item,
Panel_numeric_text_item,
Panel_text_item
```

PANEL_VALUE_STORED_LENGTH_WCS

Maximum number of wide characters to be stored in the value string for text items.

Type: `int`
Default: 80
Procs: `create, get, set`
Objects:
```
Panel_list_item,
Panel_multiline_text_item,
Panel_numeric_text_item,
Panel_text_item
```

PATH_LAST_VALIDATED_WCS

Returns the last pathname that has passed validation in the text field. If no pathname has passed validation, `NULL` is returned. The return value is set to `NULL` after setting `PATH_IS_DIRECTORY`.

`PATH_LAST_VALIDATED_WCS` returns the expanded version of the pathname, in contrast to `PANEL_VALUE`, which returns the current contents of the field.

Type: `wchar_t *`
Default: none
Procs: `get`

PATH_RELATIVE_TO_WCS

Specifies an absolute path to which any relative path input is appended to complete the pathname.

Type: `wchar_t *`
Default: none
Procs: `create, get, set`

SELN_REQ_CHARSIZE

Specifies the number of characters in the selection.

Type: `int`
Default: none
Procs: `selection_ask()`, `selection_init_request()`, `selection_query()`

SELN_REQ_CONTENTS_WCS

Specifies a pointer to a `wchar_t` string containing the selection's wide character contents.

Type: `wchar_t *`
Default: none
Procs: `selection_ask()`, `selection_init_request()`, `selection_query()`

SELN_REQ_FIRST_WC

Gives the number of characters that precede the first character of the selection.

Type: `int`
Default: none
Procs: `selection_ask()`, `selection_init_request()`, `selection_query()`

SELN_REQ_LAST_WC

Gives the character index of the last character of the selection.

Type: `int`
Default: none
Procs: `selection_ask()`, `selection_init_request()`, `selection_query()`

SERVER_IMAGE_BITMAP_FILE_WCS

Specifies a filename in wide characters for the file containing the X11 bitmap to be created. Refer to the Xlib documentation on `XReadBitmapFile()` for the file format. This is the wide character version of `SERVER_IMAGE_BITMAP_FILE`.

Type: `wchar_t *`
Default: none
Procs: `create`

TEXTSW_ACTION_CHANGED_DIRECTORY_WCS

The current working directory for the process has changed to the directory named. The name of the directory is specified in wide characters. This is the wide character version of `TEXTSW_ACTION_CHANGED_DIRECTORY`.

Type: `wchar_t *`
Default: none
Procs: `notify_proc`

TEXTSW_ACTION_EDITED_FILE_WCS

The file named by the provided wide character string has been edited. This is the wide character version of `TEXTSW_ACTION_EDITED_FILE`.

Type: `wchar_t *`
Default: none
Procs: `notify_proc`

TEXTSW_ACTION_LOADED_FILE_WCS

The text subwindow is used to view the file named. The filename is specified with a wide character string by the provided wide character string.

Type: `wchar_t *`
Default: none
Procs: `notify_proc`

TEXTSW_CONTENTS_WCS

Specifies the text for a text subwindow as a wide character string. `xv_get()` needs additional parameters. This is the wide character version of `TEXTSW_CONTENTS`.

Type: `wchar_t *`
Default: `NULL`
Procs: `create, get, set`

The following example shows how to use TEXTSW_CONTENTS_WCS to read text from a text subwindow.

```
wchar_t     *buf
int     buf_len;    /* character base */
Textsw_index    start_pos, next_pos;   /* character base */

next_pos = xv_get(textsw, TEXTSW_CONTENTS_WCS,
                                        start_pos, buf, buf_len);
```

TEXTSW_FILE_WCS

For xv_create() and xv_set(), specifies the name of the file to load in wide character format. For xv_get(), returns the name of the file loaded in wide character or NULL, if no file was loaded.

Type: wchar_t *
Default: NULL
Procs: create, get, set

TEXTSW_FILE_CONTENTS_WCS

Initializes the text subwindow contents from a file, yet still edits the contents in memory as if specified using TEXTSW_FILE_CONTENTS_WCS. The filename is specified in wide character. This is the wide character version of TEXTSW_FILE_CONTENTS.

Type: wchar_t *
Default: NULL
Procs: create, set

TEXTSW_FIRST_WC

Specifies the zero-based index in characters of the first displayed character. This is the wide character version of TEXTSW_FIRST.

Type: int
Default: none
Procs: get

TEXTSW_INSERT_FROM_FILE_WCS

Inserts the contents of a file, whose name is given as a wide character string, into a textsw at the current insertion point.

Type: `wchar_t *`
Default: `NULL`
Procs: `create, set`

TEXTSW_INSERTION_POINT_WC

Specifies the character-based index of the current insertion point. This is the wide character version of `TEXTSW_INSERTION_POINT`.

Type: `Textsw_index`
Default: none
Procs: `create, get, set`

TEXTSW_LENGTH_WC

`TEXTSW_LENGTH_WC` returns the length in characters of the `textsw`'s contents. This is dissimilar to `TEXTSW_LENGTH`, which returns the length in bytes.

Type: int
Default: none
Procs: `get`

WIN_ERROR_MSG_WCS

Specifies a wide character error message string. This is the wide character version of `WIN_ERROR_MSG`.

Type: `wchar_t *`
Default: `NULL`
Procs: `create`

WIN_IC

Sets or gets the input context (IC) handle associated with a given window. This will only affect windows which have `WIN_USE_IM` set to `TRUE`.

Type: `int`
Default: none
Procs: `create, get, set`

WIN_IC_ACTIVE

Specifies whether the input context (IC) associated with the given window is to be active. If WIN_IC_ACTIVE is TRUE, events will be directed to the input method. The attribute should be set to TRUE whenever the user expects to use the input method. WIN_IC_ACTIVE can also be set to FALSE when it is necessary to temporarily disable the input method (IM) for a particular window.

Type: Boolean
Default: TRUE
Procs: create, set, get

WIN_IC_COMMIT_STRING

If the window has an input context (IC) associated with it, querying this attribute will return the currently committed string for this window. The committed string is any input string that may have been pending before WIN_IC_RESET was set to TRUE.

Type: char *
Default: NULL
Procs: get

WIN_IC_COMMIT_STRING_WCS

If the window has an input context (IC) associated with it, querying this attribute will return the currently committed string for this window. The committed string is any input string that may have been pending before WIN_IC_RESET was set to TRUE. This is the wide character version of WIN_IC_COMMIT_STRING.

Type: wchar_t *
Default: NULL
Procs: get

WIN_IC_CONVERSION

Specifies whether input method conversion mode is on for a given window. This will only affect windows which have WIN_USE_IM set to TRUE.

Type: Boolean
Default: FALSE
Procs: create, get, set

WIN_IC_PREEDIT_CARET

Specifies the preedit caret callback and the associated callback data for a given window. The specified function is called when the input method wants the client to move the text insertion caret. This will only affect windows which have WIN_USE_IM set to TRUE.

Type: void (*preedit_start_proc)(), XPointer
Default: window dependent
Procs: create, set

WIN_IC_PREEDIT_DONE

Specifies the preedit done callback and the associated callback data for a given window. The specified function is called when input method conversion is turned off. This will only affect windows which have WIN_USE_IM set to TRUE.

Type: void (*preedit_done_proc)(), XPointer
Default: window dependent
Procs: create, set

WIN_IC_PREEDIT_DRAW

Specifies the preedit draw callback and the associated callback data for a given window. The specified function is called when the input method wants the client to draw, insert, delete, or replace preedit text in the preedit region. This will only affect windows which have WIN_USE_IM set to TRUE.

Type: void (*preedit_draw_proc)(), XPointer
Default: window dependent
Procs: create, set

WIN_IC_PREEDIT_START

Specifies the preedit start callback and the associated callback data for a given window. The specified function is called when input method conversion is turned on. This will only affect windows which have WIN_USE_IM set to TRUE.

Type: `void (*preedit_start_proc)()`, XPointer
Default: window dependent
Procs: `create`, `set`

WIN_IC_RESET

Resets the input context of a given window to its initial state. Any input pending on that input context is deleted. The input method clears the preedit area and updates the status area accordingly. If `WIN_IC_RESET` is set to `TRUE`, the input method conversion mode is turned off.

Type: `Boolean`
Default: none
Procs: `set`

WIN_IC_STATUS_DONE

Specifies the status done callback function and associated callback data for a given window. The status area is an output-only region which is intended to present the internal state of the input method to the user. The specified function is called when the input context loses focus or is destroyed. This will only affect windows which have `WIN_USE_IM` set to `TRUE`.

Type: `void (*status_done_proc)()`, XPointer
Defaults: private default function
Procs: `create`, `set`

WIN_IC_STATUS_DRAW

Specifies the status draw callback function and associated callback data for a given window. The status area is an output-only region which is intended to present the internal state of the input method to the user. The specified function is called when the input method wants the client to update the status area. This will only affect windows which have `WIN_USE_IM` set to `TRUE`.

Type: `void (*status_draw_proc)()`, XPointer
Defaults: private default function
Procs: `create`, `set`

WIN_IC_STATUS_START

Specifies the status start callback function and associated callback data for a given window. The status area is an output-only region which is intended to present the internal state of the input method to the user. The specified function is called when the input context is created or receives focus. This will only affect windows which have WIN_USE_IM set to TRUE.

Type: void (*status_start_proc)(), XPointer
Defaults: private default function
Procs: create, set

WIN_USE_IM

Specifies whether the input method will be enabled (TRUE) or disabled (FALSE) for a given window. WIN_USE_IM can be set on a frame, canvas, panel, text subwindow, or tty subwindow. The value of WIN_USE_IM can only be set with xv_create(). If WIN_USE_IM is not explicitly set, subwindows will inherit their value of WIN_USE_IM from the parent.

Type: Boolean
Default: TRUE
Procs: create, get

WIN_X_IM_STYLE_MASK

Specifies the input style mask for the input method, which will be specified as either preedit or status style. The mask is specified using XIMStyle masks in <X11/Xlib.h>. The value can be queried against any window object.

If WIN_USE_IM is TRUE, this attribute affects the input context associated with a window. XView attribute settings override input style command line options and input style resource settings.

Type: unsigned long
Default: locale-specific
Procs: xv_create(), xv_get()

XV_IM

Returns the XIM handle of the Xlib connection with the input method. XV_IM returns NULL if a connection to the input method has not been established. The server package attempts to make a connection during xv_init() only if the locale-specific resource xview.needIM.<locale> is TRUE. By default, xview.needIM.<locale> is FALSE.

The xview.needIM.<locale> resource is typically specified as TRUE in $OPENWINHOME/lib/locale/<locale>/xview/defaults, if the locale supports an input method. For the C locale and most western European locales, an input method may not be required; therefore, the default value of FALSE will suffice.

Type: XIM
Default: NULL
Procs: get

XV_IM_STYLES

Returns the XIMStyles list of supported styles. This attributes can be queried on any window or server object.

Type: XIMStyles
Default: none
Procs: xv_get()

XV_LABEL_WCS

Specifies a frame's header label or an icon's label, or simply associates a wide character name with an object. XView copies the strings on set. This is the wide character version of XV_LABEL.

Type: wchar_t *
Default: NULL
Procs: create, get, set

Functions

This section lists functions alphabetically.

defaults_get_locale()

Returns the current locale setting in the XView resource database lookup, as specified by the first argument for `defaults_set_locale()`.

```
char *
defaults_get_locale()
```

defaults_set_locale()

Turns the locale-sensitive XView resource database lookup on and off. If both `<locale>` and `<locale_attr>` are `NULL`, a call to this function turns off the locale-sensitive resource lookup (default). If `<locale>` or `<locale_attr>` is non-`NULL`, the locale-sensitive XView resource database lookup is turned on. `<locale>` must be a valid locale name, and `<locale_attr>` must be `XV_LC_BASIC_LOCALE`, `XV_LC_DISPLAY_LANG`, `XV_LC_NUMERIC`, or `XV_LC_TIME_FORMAT`.

```
void
defaults_set_locale(locale, locale_attr)
    char   *locale;          /* Locale value itself */
   int     locale_attr;      /* Locale category */
```

panel_get_value_wcs()

Macro that returns the `PANEL_VALUE_WCS` for the specified panel item. This is the wide character version of `panel_set_value()`.

```
wchar_t *
panel_get_value_wcs(item)
  Panel_item     item;
```

panel_set_value_wcs()

Macro that sets `PANEL_VALUE_WCS` for the specified panel item. This is the wide character version of `panel_get_value()`.

```
panel_set_value_wcs (item, value)
  Panel_item     item;
  wchar_t *      value;
```

textsw_add_mark_wc()

Adds a new mark at a character-based position. Flags can be either TEXTSW_MARK_DEFAULTS or TEXTSW_MARK_MOVE_AT_INSERT. This is the wide character version of textsw_add_mark().

```
Textsw_mark
textsw_add_mark_wc(textsw, position, flags)
          Textsw          textsw;
          Textsw_index    position;   /* character base */
          unsigned        flags;
```

textsw_append_file_name_wcs()

Returns 0 if textsw is editing a file and appends the name of the file at the end of name. This is the wide character version of textsw_append_file_name().

```
int
textsw_append_file_name_wcs(textsw, name)
Textsw          textsw;
wchar_t         *name;
```

textsw_delete_wcs()

Removes the span of characters beginning with first and ending one before last_plus_one. Returns 0 if the operation fails; otherwise, it returns the number of characters (not bytes) deleted. This is the wide character version of textsw_delete().

```
Textsw_index   /* character base */
textsw_delete_wcs (textsw, first, last_plus_one)
          Textsw         textsw;
          Textsw_index   first, last_plus_one; /*character base*/
```

textsw_edit_wcs()

Erases a character, a word, or a line, depending on whether unit is TEXTSW_UNIT_IS_CHAR, TEXTSW_UNIT_IS_WORD, or TEXTSW_UNIT_IS_LINE. If direction is 0, characters after the current insertion point are affected; otherwise, characters before the current insertion point are erased. The operation is repeated count times. The function returns 0 on failure. This is the wide character version of textsw_edit().

```
Textsw_index    /* character base */
textsw_edit_wcs (textsw, unit, count, direction)
          Textsw         textsw;
          unsigned       unit, count, direction;
```

textsw_erase_wcs()

Equivalent to `textsw_delete_wcs()`, but does not affect the global shelf. Returns 0 if the operation fails, otherwise, returns the number of characters (not bytes) actually deleted. This is the wide character version of `textsw_erase()`.

```
Textsw_index   /* character base */
textsw_erase_wcs (textsw, first, last_plus_one)
          Textsw         textsw;
          Textsw_index   first, last_plus_one; /* character base*/
```

textsw_find_wcs()

Beginning at the position addressed by `first`, searches for the pattern specified by `buf` of length `buf_len`. Searches forward if flag is 0; otherwise, it searches backward. Returns -1 if there is no match; otherwise, the matching span is placed in indices addressed by `first` and `last_plus_one`. This is the wide character version of `textsw_find()`.

```
int
textsw_find_wcs(textsw, first, last_plus_one, buf,
                buf_len, flags)
          Textsw         textsw;
          Textsw_index   *first, *last_plus_one;/*char. base*/
          wchar_t        *buf;
          unsigned       buf_len; /* character base */
          unsigned       flags;
```

textsw_find_mark_wc()

Returns the current position of mark in characters. If this operation fails, the function returns `TEXTSW_INFINITY`. This is the wide character version of `textsw_find_mark()`.

```
Textsw_index             /* character base */
textsw_find_mark_wc(textsw, position, flags)
          Textsw         textsw;
          Textsw_mark    position;
```

textsw_index_for_file_line_wc()

Returns the character index of the first character in the line given by `line`. If this operation fails, the function returns `TEXTSW_CANNOT_SET`. This is the wide character version of `textsw_index_for_file_line()`.

```
Textsw_index            /* character base */
textsw_index_for_file_line_wc(textsw, line)
          Textsw        textsw;
          int           line;
```

textsw_insert_wcs()

Insert wide characters from `buf` into `textsw` at the current insertion point. The number of characters (not bytes) actually inserted is returned. This will equal `buf_len` unless there was memory allocation failure. If there was a failure, the return value is 0. This is the wide character version of `textsw_insert()`.

```
Textsw_index            /* character base */
textsw_insert_wcs (textsw, buf, buf_len)
          Textsw        textsw;
          wchar_t       *buf;
          int           buf_len; /* character base */
```

textsw_match_wcs()

Searches for a block of text in the contents of a textsw. This is the wide character version of `textsw_match_bytes()`.

```
int
textsw_match_wcs(textsw, first, last_plus_one, start_sym
           start_sym_len, end_sym, end_sym_len, field_flag)

          Textsw       textsw;
          Textsw_index *first, *last_plus_one;/*character base*/
          wchar_t      *start_sym, *end_sym;
          int          start_sym_len, end_sym_len; /*char.base*/
          unsigned     field_flag;
```

textsw_normalize_view_wc()

Repositions the text so that the character at `position` (character base) is visible at the top of the subwindow. This is the wide character version of `textsw_normalize_view()`.

```
void
textsw_normalize_view_wc(textsw, position)
          Textsw         textsw;
          Textsw_index   position; /* character base */
```

textsw_possibly_normalize_wc()

If the character at the character-based `position` is already visible, this function does nothing. If the character is not visible, this function repositions the text so that the character is visible and at the top of the subwindow. This is the wide character version of `textsw_possibly_normalize()`.

```
void
textsw_possibly_normalize_wc(textsw, position)
          Textsw         textsw;
          Textsw_index   position;/* character base */
```

textsw_replace_wcs()

Replaces the character span from first to `last_plus_one` with the characters in `buf`. The return value is the number of characters inserted or deleted. The number is positive if characters are inserted, negative if characters are deleted. (The number is also negative if the original string is longer than the one that replaces it.) If this operation fails, it returns a value of `NULL`.

```
Textsw_index             /* character base */
textsw_replace_wcs(textsw, first, last_plus_one,
                           buf, buf_len)
          Textsw          textsw;
          Textsw_index    first, last_plus_one; /*char. base */
          wchar_t        *buf;
          unsigned        buf_len; /* character base */
```

textsw_set_selection_wcs()

Sets the selection to begin at `first` and include all characters up to `last_plus_one`. A `type` value of 1 indicates primary selection; 2 indicates secondary selection. This is the wide character version of `textsw_set_selection()`.

```
void
textsw_set_selection_wcs(textsw,first,last_plus_one, type)
          Textsw        textsw;
          Textsw_index  first, last_plus_one; /*character base */
          unsigned      type;
```

textsw_store_file_wcs()

Stores the contents of `textsw` to the file named by `filename`. If needed, a message box is displayed at x, y. This is the wide character version of `textsw_store_file()`.

```
unsigned
textsw_store_file_wcs (textsw, filename, x, y)
          Textsw          textsw;
          wchar_t        *filename;
          int             x, y;
```

ttysw_input_wcs()

Appends `len` number of wide characters from `buf` into input queue of the tty. The function returns the number of wide characters accepted. This is the wide character version of `ttysw_input()`.

```
int
ttysw_input_wcs (tty, buf, len)
          Tty            tty;
          wchar_t        *buf;
          int            len;
```

ttysw_output_wcs()

Appends `len` number of wide characters from `buf` into output queue of the tty. Characters are sent through the terminal emulator to the tty. It returns the number of wide characters accepted. This is the wide character version of `ttysw_output_wcs()`.

```
          int
          ttysw_output_wcs (tty, buf, len);
          Tty            tty;
          wchar_t        *buf;
          int            len;
```