Appendix B - Data Types

This appendix summarizes the data types used as arguments or return values in Motif toolkit and Motif Resource Manager functions. Xt and Xlib data types used by the routines are included. For each data type, the description states the header file that defines the type. Data types (which include simple typedefs as well as structures and enums) are listed alphabetically. Defined symbols (for example, constants used to specify the value of a mask or a field in a structure) or other data types used only to set structure members are listed with the data type in which they are used.

ArgList

An ArgList is used for setting resources in calls to widget creation routines. It is defined as follows in <*X11/Intrinsic.h*>:

```
typedef struct {
    String      name;
    XtArgVal    value;
} Arg, *ArgList;
```

The name *field is* typically a defined constant of the form *XtNresourcename* from either <*X11/Stringdefs.h*> or a widget public header file. It identifies the name of the argument to be set. The value *field is* an *XtArgVal*, a system-dependent typedef chosen to be large enough to hold a pointer to a function. It is often not large enough to hold a float or double.

Atom

To optimize communication with the server, a property is referenced by string name only once, and subsequently by a unique integer ID called an Atom. Predefined atoms are defined in <*X11/Xatom.h*> using defined symbols beginning with XA_; other atoms can be obtained from the server by calling the Xlib function `XInternAtom`(). The Motif toolkit supports an atom-caching mechanism with `XmInternAtom`(). Atoms are used by the Motif protocol routines.

Boolean

A typedef from <*X11/Intrinsic.h*> used to indicate True (1) or False (0). Use either the symbols TRUE or FALSE, defined in <*X11/Intrinsic.h*> or True or False, defined in <*X11/Xlib.h*>.

Cardinal

A typedef from <*X11/Intrinsic.h*> used to specify any unsigned integer value.

Colormap

An XID (server resource ID) from <*X11/X.h*> that identifies a Colormap resource maintained by the server. `XmGetColors`() *and* `MrmFetchColorLiteral`*()* use Colormap values.

Cursor
: A typedef in <*X11/X.h*> for an XID *(*server resource ID) that identifies a cursor resource maintained by the server. A Cursor is used to set the menu cursor in Motif. `XmTrackingEvent`() *and* `XmTrackingLocate`*()* also have a Cursor parameter.

Dimension
: A typedef from <*X11/Intrinsic.h*> used to specify an unsigned short quantity, typically used for window sizes.

Display
: A structure defined in <*X11/Xlib.h*> that contains information about the display the program is running on. Display structure fields should not be accessed directly; Xlib provides a number of macros to return essential values. In Xt, a pointer to the current Display is returned by a call to `XtDisplay`(). The Motif clipboard routines and string drawing routines, among others, use Display parameters. This data type should not be confused with the XmDisplay widget in Motif 1.2 and later.

GC
: A graphics context, which is defined in <*X11/Xlib.h*>. A GC is a pointer to a structure that contains a copy of the settings in a server resource. The server resource, in turn, contains information about how to interpret a graphics primitive. A pointer to a structure of this type is returned by the Xlib call `XCreateGC`() or the Xt call `XtGetGC`(). Motif string drawing routines use GC parameters. The members of this structure should not be accessed directly.

KeyCode
: *A* server-dependent code that describes a key that has been pressed. A *KeyCode* is defined as an unsigned character in <*X11/X.h*>. `XmTranslateKey`() takes a *KeyCode* argument.

KeySym
: A portable representation of the symbol on the cap of a key. The Motif toolkit use both virtual keysyms (osfkeysyms) and actual keysyms. The toolkit maps osfkeysyms to actual keysyms. Individual *KeySyms* are symbols defined in <*X11/keysymdef.h*>. The keycode-to-keysym lookup tables are maintained by the server, and hence a *KeySym* is actually an *XID*. `XmVaCreateSimpleOptionMenu`() and `XmTranslateKey`*()* take *KeySym* arguments.

Modifiers
: Any bitmask that describes modifier keys. The Modifiers type and its values are defined as follows in <*X11/Intrinsic.h*> and <*X11/X.h*>:

  typedef unsigned int Modifiers;

```
#define ShiftMask       (1<<0)
#define LockMask        (1<<1)
#define ControlMask     (1<<2)
#define Mod1Mask        (1<<3)
#define Mod2Mask        (1<<4)
#define Mod3Mask        (1<<5)
#define Mod4Mask        (1<<6)
#define Mod5Mask        (1<<7)
```

`XmTranslateKey`() takes an argument of type *Modifiers*.

MrmCode
: Indicates the type of a value returned by `MrmFetchLiteral`*()*. Codes are prefixed with MrmRtype and are defined in <*Mrm/MrmPublic.h*>.

MrmCount
: A typedef in <*Mrm/MrmPublic.h*> for specifying a count of items.

MrmHierarchy
: A pointer to an Mrm hierarchy opened with `MrmOpenHierarchy`*()* or `MrmOpenHierarchyPerDisplay`(). The type is defined in <*Mrm/MrmPublic.h*>. The functions associate one or more UID files with the hierarchy. An MrmHierarchy is a required argument of most of the Mrm functions.

MrmOsOpenParamPtr
: A structure of operating system-dependent settings used as an argument to `MrmOpenHierarchy`*()* and `MrmOpenHierarchyPerDisplay`() and defined in <*Mrm/MrmPublic.h*>. As of Motif 1.2, the settings are only useful to the UIL compiler.

MrmRegisterArg
: See MrmRegisterArgList.

MrmRegisterArgList
: A type used for registering application-defined procedures and identifiers with `MrmRegisterNames`*()* and `MrmRegisterNamesInHierarchy`(). It is defined as follows in <*Mrm/MrmPublic.h*>*:*

```
typedef struct {
    String          name;   /* case-sensitive name                      */
    XtPointer       value; /* value/procedure to associate with name    */
} MrmRegisterArg, *MrmRegisterArglist;
```

MrmType
: Indicates the class of a widget created with `MrmFetchWidget`*()* or `MrmFetchWidgetOverride`(). As of Motif 1.2, the types are not defined in any of the Mrm include files, although the OSF documentation states that they are defined in <*Mrm/Mrm.h*>.

Pixel
: An unsigned long integer (defined in <X11/Intrinsic.h>) that serves as a lookup key to a Colormap. If the visual type is PseudoColor, it is implemented as an index to a Colormap; for DirectColor, the RGB values are directly coded into the Pixel value. The Motif pixmap and color routines, as well as some Mrm functions, use Pixel values.

Pixmap
: *An XID* (server resource ID) that represents a two-dimensional array of pixels, used as an offscreen drawable. that is, a drawable with a specified width, height, and depth (number of planes), but no screen coordinates. The Motif pixmap routines, as well as some Mrm functions, use Pixmap parameters.

Position
: A typedef from <*X11/Intrinsic.h*> used to specify a short quantity used for x- and y-coordinates. The Motif string drawing routines, among others, use Position values.

Screen
: A structure that describes the characteristics of a screen (one or more of which make up a display). A pointer to a list of these structures is a member of the Display structure. A pointer to a structure of this type is returned by `XtScreen`() and `XGetWindowAttributes`(). The Motif pixmap routines, among others, as well as some of the Mrm functions, use Screen values. This data type should not be confused with the Screen object in Motif 1.2.

```
typedef struct {
    XExtData        *ext_data;      /* hook for extension to hang data   */
    struct _XDisplay *display;      /* back pointer to display structure */
    Window          root;           /* root window ID                    */
    int             width;          /* width of screen                   */
    int             height;         /* height of screen                  */
    int             mwidth;         /* width in millimeters              */
    int             mheight;        /* height in millimeters             */
    int             ndepths;        /* number of depths possible         */
    Depth           *depths;        /* list of allowable depths on screen*/
    int             root_depth;     /* bits per pixel                    */
    Visual          *root_visual;   /* root visual                       */
    GC              default_gc;     /* GC for the root visual            */
    Colormap        cmap;           /* default Colormap                  */
    unsigned long   white_pixel;
    unsigned long   black_pixel;    /* white and black pixel values      */
    int             max_maps;       /* max Colormaps                     */
    int             min_maps;       /* min Colormaps                     */
    int             backing_store;  /* Never, WhenMapped, Always         */
    Bool            save_unders;
    long            root_input_mask; /* initial root input mask          */
} Screen;
```

String
: A typedef for char *.

StringTable
: A pointer to a list of Strings.

Time
: A typedef for an unsigned long value (defined in <*X11/X.h*>) that contains a time value in milliseconds. The constant CurrentTime is interpreted as the time in milliseconds since the server was started. The Time data type is used in event structures and as an argument to some Motif clipboard, drag and drop, and text selection routines.

Visual
: A structure that defines a way of using color resources on a particular screen.

VoidProc
: The prototype for the procedure that copies data passed by name to the clipboard. `XmClipboardBeginCopy`() specifies a procedure of this type. It is defined as follows in <*Xm/CutPaste.h*>:

```
typedef void (*VoidProc) (Widget widget, int *data_id, int *private_id,
                          int   *reason)
```

VoidProc takes four arguments. The first argument, *widget*, is the widget passed to the callback routine, which is the same widget as passed to XmClipboardBeginCopy(). *The data_id* argument is the ID of the data item that is returned by XmClipboardCopy() *and private_id* is the private data passed to XmClipboardCopy(). *The reason* argument takes the value XmCR_CLIPBOARD_DATA_REQUEST, which indicates that the data must be copied to the clipboard, or XmCR_CLIPBOARD_DATA_DELETE, which indicates that the client can delete the data from the clipboard. Although the last three parameters are pointers to integers, the values are read-only and changing them has no effect.

Widget
: A structure returned by calls to create a widget, such as XtAppInitialize*()*, XtCreateWidget(), and XtCreateManagedWidget*()*, as well as the Motif widget creation routines. The members of this structure should not be accessed directly from applications; they should regard it as an opaque pointer. Type Widget *is* actually a pointer to a widget instance structure. Widget code accesses instance variables from this structure.

WidgetClass
: A pointer to the widget class structure, used to identify the widget class in various routines that create widgets or that return information about widgets. Widget class names have the form nameWidgetClass, with the exception of the widget-precursor classes, Object and RectObj, which have the class pointers objectClass and rectObjClass, respectively.

WidgetList
: A pointer to a list of Widgets.

Window
: A resource maintained by the server, and known on the client side only by an integer ID. In Xt, a widget's window can be returned by the XtWindow() macro. Given the window, the corresponding widget can be returned by XtWindowToWidget(). Conversely, given a widget, the window can be deduced through XtWindowOfObject(). The Motif clipboard and string drawing routines use Window values.

XEvent
: A union of all thirty event structures. The first member is always the type, so it is possible to branch on the type, and do event-specific processing in each branch. Both XmDragStart() and XmTrackingEvent() take XEvent parameters. An XButtonPressedEvent, which is one of the event structures in the union, is used by XmMenuPosition().

XFontSet
: Specifies all of the fonts needed to display text in a particular locale. The Motif font list entry routines can use XFontSet values.

XFontStruct
: Specifies metric information (in pixels) for an entire font. This structure (defined in *<X11/Xlib.h>*) is filled by means of the Xlib routines `XLoadQueryFont`() and `XQueryFont`*()*. `XListFontsWithInfo`() also fills it, but with metric information for the entire font only, not for each character. Some of the Motif font list routines use XFontStructs.

```
typedef struct {
    XExtData    *ext_data;          /* hook for extension to hang data    */
    Font        fid;                /* font ID for this font              */
    unsigned    direction;          /* direction the font is painted      */
    unsigned    min_char_or_byte2;  /* first character                    */
    unsigned    max_char_or_byte2;  /* last character                     */
    unsigned    min_byte1;          /* first row that exists              */
    unsigned    max_byte1;          /* last row that exists               */
    Bool        all_chars_exist;    /* flag if all characters have        */
                                    /* nonzero size                       */
    unsigned    default_char;       /* char to print for undefined character*/
    int         n_properties;       /* how many properties there are      */
    XFontProp   *properties;        /* pointer to array of additional     *\
                                    /* properties                         */
    XCharStruct  min_bounds;        /* minimum bounds over all            */
                                    /* existing char                      */
    XCharStruct  max_bounds;        /* maximum bounds over all            */
                                    /* existing char                      */
    XCharStruct *per_char;          /* first_char to last_char information */
    int         ascent;             /* logical extent of largest character */
                                    /* above baseline                     */
    int         descent;            /* logical descent of largest character */
                                    /* below baseline                     */
} XFontStruct;
```

The *direction* member is specified by one of the following constants from *<X11/X.h>:*

FontLeftToRight FontRightToLeft FontChange

XImage

Describes an area of the screen. This structure (defined in <X11/Xlib.h>) is used by `XmInstallImage`() and `XmUninstallImage`().

```
typedef struct _XImage {
    int                 width, height;      /* size of image in pixels          */
    int                 xoffset;            /* number of pixels offset in X direction
    */
    int                 format;             /* XYBitmap, XYPixmap, ZPixmap*/
    char                *data;              /* pointer to image data            */
    int                 byte_order;         /* data byte order: LSBFirst, MSBFirst
    */
    int                 bitmap_unit;        /* quant. of scan line 8, 16, 32    */
    int                 bitmap_bit_order; /* LSBFirst, MSBFirst                 */
    int                 bitmap_pad;         /* 8, 16, 32                      */
    int                 depth;              /* depth of image                 */
    int                 bytes_per_line;     /* accelerator to next line       */
    int                 bits_per_pixel;     /* bits per pixel (ZPixmap only)  */
    unsigned long       red_mask;           /* bits in z arrangement          */
    unsigned long       green_mask;
    unsigned long       blue_mask;
    char                *obdata;            /* hook for object routines to hang on
    */
    struct funcs {                          /* image manipulation routines
    */
        struct _XImage      *(*create_image)(void);
        int                 (*destroy_image)(struct XImage *);
        unsigned long       (*get_pixel)(struct XImage *, int, int);
        int                 (*put_pixel)(struct XImage *, int, int, unsigned int,
                                    unsigned int);
        struct _XImage      *(*sub_image)(struct XImage *, int, int, unsigned
                                    int, unsigned int);
        int                 (*add_pixel)(struct XImage *, long);
    } f;
} XImage;
```

The format *member is specified by one of the following constants defined in <X11/X.h>:*

```
XYBitmap    /* depth 1, XYFormat */
XYPixmap    /* pixmap viewed as stack of planes; depth == drawable depth
*/
ZPixmap     /* pixels in scan-line order; depth == drawable depth       */
```

byte_order and bitmap_bit_order are specified by either LSBFirst *or MSBFirst*, which are defined in <*X11/X.h*>.

XRectangle

Specifies a rectangle. This structure (defined in <*X11/Xlib.h*>) is used by the Motif string drawing routines and `XmGetDisplayRect`().

```
typedef struct {
    short           x, y;
    unsigned short  width, height;
} XRectangle;
```

XmAllocColorProc

The prototype for the per-screen color allocation procedure which is specified through the XmScreen resource XmNcolorAllocationProc. It is defined as follows in <Xm/Screen.h>:

```
typedef void (*XmAllocColorProc)(  Display     *display;
                                   /* connection to the X server                */
                                   Colormap  colormap;
                                   /* Colormap in which to allocate color */
                                   XColor     *color)
                                   /* color to allocate                         */
```

An XmAllocColorProc takes three arguments. The first *display* argument is the connection to the X server. The second argument is the Colormap in which to allocate the required color. The third color argument is where the required color is specified and returned.

XmAnyCallbackStruct

The generic Motif callback structure. It is defined as follows in <*Xm/Xm.h*>:

```
typedef struct {
    int         reason;     /* the reason that the callback was called   */
    XEvent      *event;     /* event structure that triggered callback   */
} XmAnyCallbackStruct;
```

XmArrowButtonCallbackStruct

The callback structure passed to ArrowButton callback routines. It is defined as follows in <*Xm/Xm.h*>:

```
typedef struct {
    int         reason;         /* the reason that the callback was called     */
    XEvent      *event;         /* event structure that triggered callback     */
    int         click_count;    /* number of clicks in multi-click sequence    */
} XmArrowButtonCallbackStruct;
```

XmButtonType
An enumerated type that specifies the type of button used in a simple menu creation routine. The valid values for the type are:

XmPUSHBUTTON	XmTOGGLEBUTTON
XmRADIOBUTTON	XmCHECKBUTTON
XmCASCADEBUTTON	XmTITLE
XmSEPARATOR	XmDOUBLE_SEPARATOR

XmButtonTypeTable
A pointer to a list of XmButtonType values.

XmClipboardPendingList
A structure used in calls to `XmClipboardInquirePendingItems`() to specify a data_id/private_id pair. It is defined as follows in <*Xm/CutPaste.h*>:

```
typedef struct {
    long DataId;
    long PrivateId;
} XmClipboardPendingRec, *XmClipboardPendingList;
```

XmColorProc
The prototype for the color calculation procedure used by `XmGetColorCalculation`() and `XmSetColorCalculation`(). It is defined as follows in <*Xm/Xm.h*>:

```
typedef void (*XmColorProc)(
                    XColor    *bg_color,   /* specifies the background color
*/
                    XColor    *fg_color,   /* returns the foreground color
*/
                    XColor    *sel_color, /* returns the select color
*/
                    XColor    *ts_color,   /* returns the top shadow color
*/
                    XColor    *bs_color)   /* returns the bottom shadow color
*/
```

An XmColorProc takes five arguments. The first argument, *bg_color*, is a pointer to an XColor structure that specifies the background color. The red, green, blue, and pixel fields in the structure contain valid values. The rest of the arguments are pointers to XColor structures for the colors that are to be calculated. The procedure fills in the red, green, and blue fields in these structures.

XmComboBoxCallbackStruct

The callback structure passed to ComboBox callback routines. It is defined as follows in <*Xm/Xm.h*>:

```
typedef struct {
    int         reason;         /* reason that the callback was called    */
    XEvent      *event;         /* event that triggered callback          */
    XmString    item_or_text;   /* the selected item                      */
    int         item_position;  /* the index of the item in the list      */
} XmComboBoxCallbackStruct;
```

XmCommandCallbackStruct

The callback structure passed to Command widget callback routines. It is defined as follows in *<Xm/Xm.h>:*

```
typedef struct {
    int         reason;     /* the reason that the callback was called    */
    XEvent      *event;     /* event structure that triggered callback    */
    XmString    value;      /* the string contained in the command area   */
    int         length;     /* the size of this string                    */
} XmCommandCallbackStruct;
```

XmContainerOutlineCallbackStruct

The callback structure passed to Container Outline callback routines. It is defined as follows in *<Xm/Xm.h>:*

```
typedef struct {
    int             reason;             /* reason that callback was called */
    XEvent          *event;             /* event that triggered callback   */
    Widget          item;               /* container item associated       */
                                        /* with event                      */
    unsigned char   new_outline_state;  /* the requested state             */
} XmContainerOutlineCallbackStruct;
```

XmContainerSelectCallbackStruct

The callback structure passed to Container Select callback routines. It is defined as follows in *<Xm/Xm.h>:*

```
typedef struct {
    int             reason;              /* reason that callback was called */
    XEvent          *event;              /* event that triggered callback   */
    WidgetList      selected_items;      /* the list of selected items      */
    int             selected_item_count; /* the number of selected items    */
    unsigned char   auto_selection_type; /* type of selection event         */
} XmContainerSelectCallbackStruct;
```

XmConvertCallbackStruct

The callback structure passed to the XmNconvertCallback routines of widgets when they are asked to convert a selection. It is defined as follows in *<Xm/Transfer.h>*:

```
typedef struct {
    int             reason;          /* reason that callback is invoked          */
    XEvent          *event;          /* event that triggered callback            */
    Atom            selection;       /* requested conversion selection           */
    Atom            target;          /* the conversion target                    */
    XtPointer       source_data;     /* selection source information             */
    XtPointer       location_data;   /* information on data to be transferred    */
    int             flags;           /* input status of the conversion           */
    XtPointer       parm;            /* parameter data for the target            */
    int             parm_format;     /* format of parameter data                 */
    unsigned long   parm_length;     /* number of elements in parameter data     */
    Atom            parm_type;       /* the type of the parameter data           */
    int             status;          /* output status of the conversion          */
    XtPointer       value;           /* returned conversion data                 */
    Atom            type;            /* type of conversion data returned         */
    int             format;          /* format of the conversion data            */
    unsigned long   length;          /* number of elements in conversion data    */
} XmConvertCallbackStruct;
```

XmCutPasteProc

The prototype for the procedure that copies data passed by name to the clipboard. `XmClipboardStartCopy()` specifies a procedure of this type. It is defined as follows in <*Xm/CutPaste.h*>:

typedef void (*XmCutPasteProc) (Widget widget, long *data_id, long *private_id, int *reason)

An XmCutPasteProc takes four arguments. The first argument, *widget*, is the widget passed to the callback routine, which is the same widget as passed to `XmClipboardBeginCopy`(). *The data_id* argument is the ID of the data item that is returned by `XmClipboardCopy`() and *private_id* is the private data passed to `XmClipboardCopy`(). *The reason* argument takes the value XmCR_CLIPBOARD_DATA_REQUEST, which indicates that the data must be copied to the clipboard, or XmCR_CLIPBOARD_DATA_DELETE, which indicates that the client can delete the data from the clipboard. Although the last three parameters are pointers, the values are read-only and changing them has no effect.

XmDestinationCallbackStruct

The callback structure passed to the XmNdestinationCallback routines of widgets when they are the destination of a data transfer. It is defined as follows in <*Xm/Transfer.h*>:

```
typedef struct {
    int         reason;           /* reason that callback is invoked          */
    XEvent      *event;           /* event that triggered callback            */
    Atom        selection;        /* requested selection type, as an Atom     */
    XtEnum      operation;        /* the type of transfer requested           */
    int         flags;            /* whether destination and source are same*/
    XtPointer   transfer_id;      /* unique identifier for the request        */
    XtPointer   destination_data;  /* information about the destination       */
    XtPointer   location_data;    /* information about the data               */
    Time        time;             /* time when transfer operation started     */
} XmDestinationCallbackStruct;
```

XmDirection

An enumerated type that specifies a direction, used either in laying out components of a widget, or in rendering compound strings. The valid values for the type are:

XmRIGHT_TO_LEFT_TOP_TO_BOTTOM
XmLEFT_TO_RIGHT_TOP_TO_BOTTOM
XmRIGHT_TO_LEFT_BOTTOM_TO_TOP
XmLEFT_TO_RIGHT_BOTTOM_TO_TOP
XmRIGHT_TO_LEFT
XmLEFT_TO_RIGHT
XmTOP_TO_BOTTOM_RIGHT_TO_LEFT
XmBOTTOM_TO_TOP_RIGHT_TO_LEFT
XmTOP_TO_BOTTOM_LEFT_TO_RIGHT
XmBOTTOM_TO_TOP_LEFT_TO_RIGHT
XmTOP_TO_BOTTOM
XmBOTTOM_TO_TOP
XmDEFAULT_DIRECTION

XmDisplayCallbackStruct

The callback structure passed to Display XmNnoFontCallback and XmNnoRenditionCallback callback routines. It is defined as follows in <*Xm/Xm.h*>:

```
typedef struct {
    int             reason;         /* reason that the callback was called
    */
    XEvent          *event;         /* event that triggered callback      */
    XmRendition     rendition;      /* rendition with a missing font      */
    char            *font_name;     /* font which is not loadable         */
    XmRenderTable   render_table;   /* render table with missing rendition
    */
    XmString        tag;            /* tag of the missing rendition       */
} XmDisplayCallbackStruct;
```

XmDragDropFininshCallbackStruct

The callback structure passed to the XmNdragDropFinishCallback of a DragContext object. It is defined as follows in <*Xm/DragC.h*>:

```
typedef struct {
    int         reason;         /* the reason the callback was called     */
    XEvent      *event;         /* event that triggered callback          */
    Time        timeStamp;      /* time at which operation completed      */
} XmDragDropFinishCallbackStruct, *XmDragDropFinishCallback;
```

XmDragMotionCallbackStruct

The callback structure passed to the XmNdragMotionCallback of a DragContext object. It is defined as follows in *<Xm/DragC.h>*:

```
typedef struct {
    int             reason;         /* reason the callback was called     */
    XEvent          *event;         /* event that triggered callback      */
    Time            timeStamp;      /* timestamp of logical event         */
    unsigned char   operation;      /* current operation                  */
    unsigned char   operations;     /* supported operations               */
    unsigned char   dropSiteStatus; /* valid, invalid, or none            */
    Position        x;              /* x-coordinate of pointer            */
    Position        y;              /* y-coordinate of pointer            */
} XmDragMotionCallbackStruct, *XmDragMotionCallback;
```

XmDragProcCallbackStruct

The callback structure passed to the XmNdragProc of a drop site. It is defined as follows in *<Xm/DropSMgr.h>*:

```
typedef struct {
    int             reason;         /* reason the callback was called     */
    XEvent          *event;         /* event that triggered callback      */
    Time            timeStamp;      /* timestamp of logical event         */
    Widget          dragContext;    /* DragContext widget associated      */
                                    /* with operation                     */
    Position        x;              /* x-coordinate of pointer            */
    Position        y;              /* y-coordinate of pointer            */
    unsigned char   dropSiteStatus; /* valid or invalid                   */
    unsigned char   operation;      /* current operation                  */
    unsigned char   operations;     /* supported operations               */
    Boolean         animate;        /* toolkit or receiver does animation */
} XmDragProcCallbackStruct, *XmDragProcCallback;
```

XmDrawingAreaCallbackStruct

The callback structure passed to DrawingArea callback routines. It is defined as follows in *<Xm/Xm.h>:*

```
typedef struct {
    int         reason;     /* reason that the callback was called        */
    XEvent      *event;     /* event that triggered callback;             */
                            /* for XmNresizeCallback, this is NULL        */
    Window      window;     /* the widget's window                        */
} XmDrawingAreaCallbackStruct;
```

XmDrawnButtonCallbackStruct

The callback structure passed to DrawnButton callback routines. It is defined as follows in *<Xm/Xm.h>:*

```
typedef struct {
    int         reason;       /* reason that the callback was called      */
    XEvent      *event;       /* event that triggered callback            */
    Window      window;       /* ID of window in which event occurred     */
    int         click_count;  /* number of multi-clicks                   */
} XmDrawnButtonCallbackStruct;
```

XmDropFinishCallbackStruct

The callback structure passed to the XmNdropFinishCallback of a DragContext object. It is defined as follows in *<Xm/DragC.h>*:

```
typedef struct {
    int             reason;           /* reason the callback was called   */
    XEvent          *event;           /* event that triggered callback    */
    Time            timeStamp;        /* time at which drop completed     */
    unsigned char   operation;        /* current operation                */
    unsigned char   operations;       /* supported operations             */
    unsigned char   dropSiteStatus;   /* valid, invalid, or none          */
    unsigned char   dropAction;       /* drop, cancel, help, or interrupt */
    unsigned char   completionStatus; /* success or failure               */
} XmDropFinishCallbackStruct, *XmDropFinishCallback;
```

XmDropProcCallbackStruct

The callback structure passed to the XmNdropProc of a drop site. It is defined as follows in *<Xm/DropSMgr.h>*:

```
typedef struct {
    int             reason;           /* reason callback was called       */
    XEvent          *event;           /* event that triggered callback    */
    Time            timeStamp;        /* timestamp of logical event       */
    Widget          dragContext;      /* DragContext widget associated    */
                                      /* with operation                   */
    Position        x;                /* x-coordinate of pointer          */
    Position        y;                /* y-coordinate of pointer          */
    unsigned char   dropSiteStatus;   /* valid or invalid                 */
    unsigned char   operation;        /* current operation                */
    unsigned char   operations;       /* supported operations             */
    unsigned char   dropAction;       /* drop or help                     */
} XmDropProcCallbackStruct, *XmDropProcCallback;
```

XmDropSiteEnterCallbackStruct

The callback structure passed to the XmNdropSiteEnterCallback of a DragContext object. It is defined as follows in *<Xm/DragC.h>*:

```
typedef struct {
    int             reason;         /* reason the callback was called   */
    XEvent          *event;         /* event that triggered callback    */
    Time            timeStamp;      /* time of crossing event           */
    unsigned char   operation;      /* current operation                */
    unsigned char   operations;     /* supported operations             */
    unsigned char   dropSiteStatus; /* valid, invalid, or none          */
    Position        x;              /* x-coordinate of pointer          */
    Position        y;              /* y-coordinate of pointer          */
} XmDropSiteEnterCallbackStruct, *XmDropSiteEnterCallback;
```

XmDropSiteLeaveCallbackStruct

The callback structure passed to the XmNdropSiteLeaveCallback of a DragContext object. It is defined as follows in *<Xm/DragC.h>*:

```
typedef struct {
    int             reason;         /* reason callback was called       */
    XEvent          *event;         /* event that triggered callback    */
    Time            timeStamp;      /* time of crossing event           */
} XmDropSiteLeaveCallbackStruct, *XmDropSiteLeaveCallback;
```

XmDropStartCallbackStruct

The callback structure passed to the XmNdropStartCallback of a DragContext object. It is defined as follows in *<Xm/DragC.h>*:

```
typedef struct {
    int             reason;         /* reason callback was called       */
    XEvent          *event;         /* event that triggered callback    */
    Time            timeStamp;      /* time at which drag completed     */
    unsigned char   operation;      /* current operation                */
    unsigned char   operations;     /* supported operations             */
    unsigned char   dropSiteStatus; /* valid, invalid, or none          */
    unsigned char   dropAction;     /* drop, cancel, help, or interrupt */
    Position        x;              /* x-coordinate of pointer          */
    Position        y;              /* y-coordinate of pointer          */
    Window          window;         /* internal: not documented         */
    Atom            icchandle;      /* internal: not documented         */
} XmDropStartCallbackStruct, *XmDropStartCallback;
```

XmDropTransferEntryRec
A structure that specifies the targets of a drop operation for a Drop Transfer object. It is defined as follows in *<Xm/DropTrans.h>:*

```
typedef struct {
    XtPointer    client_data;    /* data passed to the transfer proc    */
    Atom         target;         /* target format of the transfer       */
} XmDropTransferEntryRec, *XmDropTransferEntry;
```

XmDropTransferEntry
See XmDropTransferEntryRec.

XmFileSelectionBoxCallbackStruct
The callback structure passed to FileSelectionBox callback routines. It is defined as follows in *<Xm/Xm.h>:*

```
typedef struct {
    int          reason;         /* reason that callback was called     */
    XEvent       *event;         /* event that triggered callback       */
    XmString     value;          /* current value of XmNdirSpec resource */
    int          length;         /* number of bytes in value member     */
    XmString     mask;           /* current value of XmNdirMask resource */
    int          mask_length;    /* number of bytes in mask member      */
    XmString     dir;            /* current base directory              */
    int          dir_length;     /* number of bytes in dir member       */
    XmString     pattern;        /* current search pattern              */
    int          pattern_length; /* number of bytes in pattern member   */
} XmFileSelectionBoxCallbackStruct;
```

XmFontContext
A typedef for a font list context that lets an application access the font list entries and font list tags in a font list. This data type is an opaque structure returned by a call to `XmFontListInitFontContext`(), and is used in subsequent calls to `XmFontListGetNextEntry`(), `XmFontListGetNextFont`() and `XmFontListFreeFontContext`().

XmFontList
A font list contains entries that describe the fonts that are in use. In Motif 1.1, each entry associates a font and a character set. In Motif 1.2, each entry consists of a XmFontListEntry and an associated tag, where the XmFontListEntry specifies a font or a font set. XmFontList is an opaque data type used in calls to font list routines and string manipulation routines. When a Motif compound string is displayed, the font list tag is used to match the string with a font or font set, so that the compound string is displayed appropriately. The font list tag XmFONTLIST_DEFAULT_TAG causes compound strings to be displayed using the font for the current locale.

To specify a font list in a resource file, use the following syntax:

resource_spec: font_entry [, font_entry] ...

The value specification consists of at least one font list entry, with multiple entries separated by commas. Each font_entry specifies a font or a font set and an optional font list entry tag. Use the following syntax to specify a single font:

font_name [= font_list_entry_tag]

To specify the optional tag for a single font, separate the font_name and the font_list_entry_tag by an equal sign (=). Use the following syntax to specify a font set:

font_name [; font_name] ... : [font_list_entry_tag]

Separate multiple font_names with semicolons and end the specification with a colon, followed by the optional tag. A font_name is an X Logical Font Description (XLFD) string. If a font_list_entry_tag is not specified for an entry, XmFONTLIST_DEFAULT_TAG is used.

In Motif 2.0 and later, the XmFontList is considered obsolete, and is replaced by the XmRenderTable. The XmFontList type is maintained for backwards compatibility, and is implemented through a render table.

XmFontListEntry

In Motif 1.2, a font list entry is an element of an XmFontList that specifies a font or a font set. Each XmFontListEntry is associated with a font list entry tag. XmFontListEntry is an opaque type.

In Motif 2.0 and later, the XmFontList and XmFontListEntry are considered obsolete, and are replaced by the XmRenderTable and XmRendition object respectively. The XmFontList and XmFontListEntry types are maintained for backwards compatibility, and are implemented directly through the render table and rendition object.

XmFontType

An enumerated type that specifies the type of entry in a XmFontListEntry. It is defined as follows in *<Xm/Xm.h>*:

```
typedef enum {
    XmFONT_IS_FONT,         /* specifies a font         */
    XmFONT_IS_FONTSET       /* specifies a font set     */
    XmFONT_IS_XFT           /* specifies an XFT font */
} XmFontType;
```

XmHighlightMode

An enumerated type that defines the kind of text highlighting that results from calls to `XmTextSetHighlight`() and `XmTextFieldSetHighlight`(). It is defined as follows in *<Xm/Xm.h>:*

```
typedef enum {
    XmHIGHLIGHT_NORMAL,          /* no highlighting                */
    XmHIGHLIGHT_SELECTED,        /* highlight in reverse video     */
    XmHIGHLIGHT_SECONDARY_SELECTED
                                 /* highlight by underlining       */
    XmSEE_DETAIL                 /* unused except by abortive      */
                                 /* Motif 2.0 CSText widget        */
} XmHighlightMode;
```

XmICCEncodingStyle

An enumerated type which specifies the way in which compound string tables are converted to and from a text property. It is defined as follows in *<Xm/Xm.h>:*

```
typedef enum {
    XmSTYLE_STRING               = XStringStyle,
    XmSTYLE_COMPOUND_TEXT        = XCompoundTextStyle,
    XmSTYLE_TEXT                 = XTextStyle,
    XmSTYLE_STANDARD_ICC_TEXT    = XStdICCTextStyle,
    XmSTYLE_LOCALE               = 32,
    XmSTYLE_COMPOUND_STRING
} XmICCEncodingStyle;
```

XmIncludeStatus

A typedef for unsigned char that is used to define the way in which compound strings are parsed when a ParseMapping object is applied to an input stream. Variables of this type can have the following values:

```
XmINSERT        /* concatenate XmNsubstitute value to output */
                /* parsing is continued */
XmINVOKE        /* result determined by XmNinvokeParseProc   */
XmTERMINATE     /* concatenate XmNsubstitute value to output */
                /* parsing is termi nated                    */
```

XmKeySymTable

A pointer to a list of KeySyms.

XmListCallbackStruct

The callback structure passed to List widget callback routines. It is defined as follows in *<Xm/Xm.h>*:

```
typedef struct {
    int         reason;                 /* reason that callback was called */
    XEvent      *event;                 /* event that triggered callback   */
    XmString    item;                   /* item most recently selected at  */
                                        /* the time event occurred         */
    int         item_length;            /* number of bytes in item member*/
    int         item_position;          /* item position in XmNitems array*/
    XmString    *selected_items;        /* list of items selected at the   */
                                        /* time event occurred             */
    int         selected_item_count;    /* number of items in              */
                                        /* selected_items                  */
    int         *selected_item_positions; /* array of integers marking     */
                                        /* selected items                  */
    char        selection_type;         /* type of the most recent         */
                                        /* selection                       */
    char        auto_selection_type;    /* 2.0 or later: automatic         */
                                        /* selection type                  */
} XmListCallbackStruct;
```

The structure members event, item, item_length, and item_position are valid for any value of reason. The structure members selected_items, selected_item_count, and selected_item_ positions are valid when the reason field has a value of XmCR_MULTIPLE_SELECT or XmCR_EXTENDED_SELECT. The structure member selection_type is valid only when the reason field is XmCR_EXTENDED_SELECT.

For the strings pointed to by item and selected_items, as well as for the integers pointed to by selected_item_positions, storage is overwritten each time the callback is invoked. Applications that need to save this data should make their own copies of it.

selected_item_positions is an integer array. The elements of the array indicate the positions of each selected item within the List widget's XmNitems array.

selection_type specifies what kind of extended selection was most recently made. One of three values is possible, defined in *<Xm/List.h>*:

```
XmINITIAL       /* selection was the initial selection              */
XmMODIFICATION  /* selection changed an existing selection          */
XmADDITION      /* selection added non-adjacent items to an        */
                /* existing selection                               */
```

auto_selection_type specifies at what point within the selection the user is. Possible values, defined in *<Xm/Xm.h>:*

XmAUTO_UNSET	XmAUTO_BEGIN
XmAUTO_MOTION	XmAUTO_CANCEL
XmAUTO_NO_CHANGE	XmAUTO_CHANGE

XmMergeMode

An enumerated type that specifies the way in which renditions are merged into a render table. The valid values for the type are:

XmSKIP	XmMERGE_REPLACE
XmMERGE_OLD	XmMERGE_NEW
XmDUPLICATE	

XmDUPLICATE is an internal value used in mapping XmFontList and XmFontListEntry types to the render table types of Motif 2.0 and later.

XmNavigationType

An enumerated type that specifies the type of keyboard navigation associated with a widget. The valid values for the type are:

XmNONE	XmTAB_GROUP
XmSTICKY_TAB_GROUP	XmEXCLUSIVE_TAB_GROUP

XmNotebookCallbackStruct

The callback structure passed to Notebook selection callback routines. It is defined as follows in *<Xm/Xm.h>*:

```
typedef struct {
    int         reason;             /* reason that callback was called  */
    XEvent      *event;             /* points to event structure        */
                                    /* that triggered callback          */
    int         page_number;        /* current logical page number      */
    Widget      page_widget;        /* widget associated with           */
                                    /* current logical page number      */
    int         prev_page_number;   /* previous logical page number     */
    Widget      prev_page_widget;   /* widget associated with           */
                                    /* previous logical page number     */
} XmNotebookCallbackStruct;
```

XmNotebookPageInfo
Specifies a structure passed to the function `XmNotebookGetPageInfo`() in order to retrieve information about a Notebook page. It is defined as follows in <*Xm/Notebook.h*>:

```
typedef struct {
    int         page_number;        /* logical page number          */
    Widget      page_widget;        /* widget ID of a page child    */
    Widget      status_area_widget; /* widget ID of a status area child */
    Widget      major_tab_widget;   /* widget ID of a major tab child */
    Widget      minor_tab_widget;   /* widget ID of a minor tab child */
} XmNotebookPageInfo;
```

XmOffset
A long integer that represents the units used in calculating the offsets into a widget's instance data. The type is used internally to Motif. See also XmOffsetPtr.

XmOffsetModel
An enumerated type that specifies whether tabs are calculated at absolute offsets, or relative to the previous tab. The valid values for the type are:

```
XmABSOLUTE    XmRELATIVE
```

XmOffsetPtr
A pointer to an XmOffset value, which is returned by a calls to `XmResolveAllPartOffsets`() and `XmResolvePartOffsets`().

XmOperationChangedCallbackStruct
The callback structure passed to the XmNoperationChangedCallback of a DragContext object. It is defined as follows in <*Xm/DragC.h*>:

```
typedef struct {
    int             reason;             /* reason callback was called   */
    XEvent          *event;             /* event that triggered callback */
    Time            timeStamp;          /* timestamp of logical event   */
    unsigned char   operation;          /* current operation            */
    unsigned char   operations;         /* supported operations         */
    unsigned char   dropSiteStatus;     /* valid, invalid, or none      */
} XmOperationChangedCallbackStruct, *XmOperationChangedCallback;
```

XmParseMapping
A typedef for a parse mapping object that lets an application control the way in which an input stream of bytes is converted into the components or segments within a compound string. This data type is an opaque structure returned by a call to `XmParseMappingCreate`(), and is placed into an XmParseTable and used in subsequent calls to the string manipulation routines: `XmStringParse-`

`Text`(), `XmStringTableParseStringArray`(), and `XmStringTable-Unparse`(), and `XmStringUnparse`().

XmParseModel
An enumerated type which specifies how non-text components of a compound string are unparsed. It is defined as follows in <*Xm/Xm.h*>:

```
typedef enum {
    XmOUTPUT_ALL,
    XmOUTPUT_BETWEEN,
    XmOUTPUT_BEGINNING,
    XmOUTPUT_END,
    XmOUTPUT_BOTH
} XmParseModel;
```

This data type is used in calls to the following compound string routines: `XmStringTableUnparse`(), and `XmStringUnparse`().

XmParseProc
A procedure within an XmParseMapping object for controlling the way in which an input stream is parsed into a compound string. It is defined as follows in <*Xm/Xm.h*>:

```
typedef XmIncludeStatus (*XmParseProc) (    XtPointer           *in_out,
                                            XtPointer           text_end,
                                            XmTextType          type,
                                            XmStringTag         locale_tag,
                                            XmParseMapping      entry,
                                            int             pattern_length,
                                            XmString          *str_include,
                                            XtPointer           call_data);
```

XmParseTable
A typedef for an array of parse mapping objects, used for parsing an input stream into a compound strings.

```
typedef XmParseMapping *XmParseTable;
```

XmPushButtonCallbackStruct
The callback structure passed to PushButton callback routines. It is defined as follows in <*Xm/Xm.h*>:

```
typedef struct {
    int         reason;         /* reason that callback was called      */
    XEvent      *event;         /* event that triggered callback        */
    int         click_count;    /* number of multi-clicks               */
} XmPushButtonCallbackStruct;
```

XmPopupHandlerCallbackStruct

The callback structure passed to Popup Handler callback routines. It is defined as follows in *<Xm/Xm.h>:*

```
typedef struct {
    int         reason;         /* the reason the callback is invoked      */
    XEvent      *event;         /* event that triggered callback           */
    Widget       menuToPost;    /* the menu to post                        */
    Boolean     postIt;         /* whether to continue posting             */
    Widget      target;         /* manager descendant issuing request      */
} XmPopupHandlerCallbackStruct;
```

XmPrintShellCallbackStruct

The callback structure passed to PrintShell callback routines. It is defined as follows in <Xm/Xm.h>:

```
typedef struct {
    int         reason;         /* reason that the callback is invoked     */
    XEvent      *event;         /* event that triggered the callback       */
    XPContext   context;        /* X Print Context                         */
    Boolean     last_page;      /* whether this is the last page           */
    XtPointer   detail;         /* PDM selection                           */
} XmPrintShellCallbackStruct;
```

XmQualifyProc

The prototype for the qualification procedure that produces a qualified directory mask, base directory, and search pattern for the directory and file search procedures in a FileSelectionBox. The XmNqualifySearchDataProc resource specifies a procedure of this type, which is defined as follows in *<Xm/FileSB.h>*:

typedef void (*XmQualifyProc) (Widget widget, XtPointer input_data, XtPointer output_data)

An XmQualifyProc takes three arguments. The first argument, widget, is the FileSelectionBox widget. The input_data argument is a pointer to an XmFileSelectionBoxCallbackStruct that contains input data to be qualified. The output_data argument is a pointer to an XmFileSelectionBoxCallbackStruct that is to be filled in by the qualification procedure.

XmRendition

An opaque data structure, implemented as a pseudo-widget, which encapsulates the resources required to render a compound string. This data type is returned by a call to `XmRenditionCreate`(), and is used in subsequent calls to the following routines: `XmRenderTableAddRenditions`(), `XmRenditionFree`(), `XmRenditionRetrieve`(), `XmRenditionUpdate`().

XmRenderTable
An opaque data structure, representing a list of XmRendition objects, used to render compound strings. Typically used as the XmNrenderTable resource of a widget, the type us used in calls to the following routines: `XmRenderTableCopy`(), `XmRenderTableFree`(), `XmRenderTableGetRendition`(), `XmRenderTableGetRenditions`(), `XmRenderTableGetTags`(), `XmRenderTableRemoveRenditions`().

XmRepTypeEntry
A pointer to a representation type entry structure which contains information about the value names and values for an enumerated type. The Motif representation type manager routines use values of this type, which is defined as follows in <*Xm/RepType.h*>:

```
typedef struct {
    String          rep_type_name;      /* name of representation type      */
    String          *value_names;       /* array of value names             */
    unsigned char   *values;            /* array of numeric values          */
    unsigned char   num_values;         /* number of values                 */
    Boolean         reverse_installed;  /* reverse converter installed flag */
    XmRepTypeId     rep_type_id;        /* representation type ID           */
} XmRepTypeEntryRec, *XmRepTypeEntry, XmRepTypeListRec,
                                              *XmRepTypeList;
```

XmRepTypeId
An unsigned short that specifies the identification number of a representation type registered with the representation type manager. The representation type manager routines use values of this type.

XmRepTypeList
See XmRepTypeEntry.

XmRowColumnCallbackStruct
The callback structure passed to RowColumn callback routines. It is only used by map and unmap callbacks, and is defined as follows in <*Xm/Xm.h*>:

```
typedef struct {
    int         reason;         /* reason that callback was called      */
    XEvent      *event;         /* event that triggered callback        */
    Widget      widget;         /* ID of activated RowColumn item       */
    char        *data;          /* value of application's client data   */
    char        *callbackstruct; /* created when item is activated      */
} XmRowColumnCallbackStruct;
```

widget, *data*, and callbackstruct *are set to NULL*.

XmScaleCallbackStruct

The callback structure passed to Scale widget callback routines. It is defined as follows in *<Xm/Xm.h>:*

```
typedef struct {
    int         reason;     /* reason that callback was called     */
    XEvent      *event;     /* event that triggered callback       */
    int         value;      /* new value of the slider             */
} XmScaleCallbackStruct;
```

XmScreenColorProc

The prototype for the per-screen color calculation procedure which is specified through the XmScreen resource XmNcolorCalculationProc. It is defined as follows in *<Xm/Screen.h>*:

```
typedef void (*XmScreenColorProc)(
                Screen      *screen,    /* screen of top-level window          */
                XColor      *bg_color,  /* specifies the background color      */
                XColor      *fg_color,  /* returns the foreground color        */
                XColor      *sel_color, /* returns the select color            */
                XColor      *ts_color,  /* returns the top shadow color        */
                XColor      *bs_color)  /* returns the bottom shadow color     */
```

An XmScreenColorProc takes six arguments. The first argument is a pointer to a screen object. The second argument, bg_color, is a pointer to an XColor structure that specifies the background color. The red, green, blue, and pixel fields in the structure contain valid values. The rest of the arguments are pointers to XColor structures for the colors that are to be calculated. The procedure fills in the red, green, and blue fields in these structures.

XmScrollBarCallbackStruct

The callback structure passed to ScrollBar callback routines. It is defined as follows in *<Xm/Xm.h>*:

```
typedef struct {
    int         reason;     /* reason that callback was called     */
    XEvent      *event;     /* event that triggered callback       */
    int         value;      /* value of the slider's new location  */
    int         pixel;      /* coordinate where selection occurred */
} XmScrollBarCallbackStruct;
```

XmSearchProc

The prototype for a search procedure that searches the directories or files in a FileSelectionBox. The XmNdirSearchProc and XmNfileSearchProc resources specify procedures of this type, which is defined as follows in *<Xm/FileSB.h>:*

```
typedef void (*XmSearchProc) (Widget widget, XtPointer search_data)
```

An XmSearchProc takes two arguments. The first argument, widget, is the FileSelectionBox widget. The search_data argument is a pointer to an XmFileSelectionBoxCallbackStruct that contains the information for performing a search.

XmSecondaryResourceData

A structure that specifies information about secondary resources associated with a widget class. `XmGetSecondaryResourceData`() returns an array of these values. The type is defined as follows in *<Xm/Xm.h>*:

```
typedef struct {
    XmResourceBaseProc  base_proc;
    XtPointer           client_data;
    String              name;
    String              res_class;
    XtResourceList      resources;
    Cardinal            num_resources;
} XmSecondaryResourceDataRec, *XmSecondaryResourceData;
```

XmSelectionBoxCallbackStruct

The callback structure passed to SelectionBox callback routines. It is defined as follows in *<Xm/Xm.h>:*

```
typedef struct {
    int         reason;     /* reason that callback was called              */
    XEvent      *event;     /* event that triggered callback                */
    XmString    value;      /* selection string that was either chosen      */
                            /* from the SelectionBox list or typed in       */
    int         length;     /* number of bytes of value                     */
} XmSelectionBoxCallbackStruct;
```

XmSelectionCallbackStruct

The callback structure passed to routines which are responsible for data transfer from the primary selection. The function `XmTransferValue`() takes as its third parameter a procedure which is responsible for inserting data into the desti-

nation. The procedure receives a pointer to an XmSelectionCallbackStruct as callback data when invoked. It is defined as follows in *<Xm/Transfer.h>:*

```
typedef struct {
    int             reason;      /* reason the callback was invoked          */
    XEvent          *event;      /* event which triggered callback           */
    Atom            selection;   /* selection that has been converted        *
    Atom            target;      /* target for which conversion requested    */
    Atom            type;        /* type of the selection value              */
    XtPointer       transfer_id; /* unique identifier for transfer operation */
    int             flags;       /* unused: pass constant                    */
                                 /* XmSELECTION_DEFAULT                      */
    int             remaining;   /* number of transfers remaining in         */
                                 /* operation                                */
    XtPointer       value;       /* the data transferred in this request     */
    unsigned long   length;      /* the number of elements in the value      */
    int             format;      /* size of each element in the value        */
} XmSelectionCallbackStruct;
```

XmSpinBoxCallbackStruct

The callback structure passed to SpinBox callback routines. It is defined as follows in *<Xm/Xm.h>:*

```
typedef struct {
    int         reason;           /* the reason that the callback was called */
    XEvent      *event;           /* points to event that triggered callback */
    Widget      widget;           /* the textual child affected by callback  */
    Boolean     doit;             /* whether to perform the changes          */
    int         position;         /* specifies the index of the next value   */
    XmString    value;            /* specifies the next value                */
    Boolean     crossed_boundary; /* whether the SpinBox has wrapped
    */
} XmSpinBoxCallbackStruct;
```

XmString

The data type for Motif compound strings. In Motif 1.2, a compound string is composed of one or more segments, where each segment can contain a font list element tag, a string direction, and a text component. The font list element tag XmFONTLIST_DEFAULT_TAG specifies a text segment encoded in the current locale. In Motif 1.1, compound strings use character set identifiers rather than font list element tags. The character set identifier for a compound string can have the value XmSTRING_DEFAULT_CHARSET, which takes the character set from the current language environment, but this value may be removed from future versions of Motif.

XmStringCharSet
A typedef for char * that is used to define the character set of a compound string in Motif 1.1. Variables of this type can have the following values, among others:

XmSTRING_ISO8859_1
XmSTRING_OS_CHARSET
XmSTRING_DEFAULT_CHARSET

XmSTRING_DEFAULT_CHARSET specifies the character set from the current language environment, but this value may be removed from future versions of Motif.

XmStringCharSetTable
A pointer to a list of XmStringCharSets.

XmStringComponentType
An unsigned char value that specifies the type of component in a compound string segment. Values of this type are returned by calls to `XmStringGetNextComponent`() and `XmStringPeekNextComponent`(). The valid values for the type are:

```
XmSTRING_COMPONENT_FONTLIST_ELEMENT_TAG
                                /* font list element tag component     */
                                /* obsolete in Motif 2.0               */
XmSTRING_COMPONENT_CHARSET
                                /* character set identifier component; */
                                /* obsolete in Motif 1.2               */
XmSTRING_COMPONENT_TEXT         /* text component                      */
XmSTRING_COMPONENT_LOCALE_TEXT
                                /* locale-encoded text component       */
XmSTRING_COMPONENT_DIRECTION
                                /* direction component                 */
XmSTRING_COMPONENT_SEPARATOR
                                /* separator component                 */
XmSTRING_COMPONENT_END          /* last component in string            */
XmSTRING_COMPONENT_UNKNOWN
                                /* unknown component                   */
XmSTRING_COMPONENT_LOCALE
                                /* the locale specifier                */
XmSTRING_COMPONENT_WIDECHAR_TEXT
                                /* widechar text component             */
XmSTRING_COMPONENT_LAYOUT_PUSH
                                /* stacked layout direction            */
XmSTRING_COMPONENT_LAYOUT_POP
                                /* unstacked layout component          */
```

```
    XmSTRING_COMPONENT_RENDITION_BEGIN
                                    /* beginning of rendition          */
    XmSTRING_COMPONENT_RENDITION_END
                                    /* end of rendition                */
    XmSTRING_COMPONENT_TAG          /* charset/font list tag component */
    XmSTRING_COMPONENT_TAB          /* tab component                   */
```

XmStringContext
: A typedef for a string context that lets an application access the components or segments within a compound string. This data type is an opaque structure returned by a call to `XmStringInitContext`(), and is used in subsequent calls to the four other string context routines: `XmStringFreeContext`(), `XmStringGetNextSegment`(), `XmStringGetNextComponent`(), and `XmStringPeekNextComponent`().

`XmStringDirection`
: An unsigned char used for determining the direction in which a compound string is displayed. The type is used in calls to `XmStringDirectionCreate`() and `XmStringSegmentCreate`(). The valid values for the type are:

    XmSTRING_DIRECTION_L_TO_R
    XmSTRING_DIRECTION_R_TO_L
    XmSTRING_DIRECTION_DEFAULT

XmStringTable
: An opaque typedef for XmString * that is used for arrays of compound strings.

XmStringTag
: A typedef for char * that is used to specify the tag which identifies components or segments within a compound string. This data type is used in calls to the following compound string routines: `XmRenderTableCopy`(), `XmRenderTableGetRendition`(), `XmRenderTableGetRenditions`(), `XmRenderTableGetTags`(), `XmRenderTableRemoveRenditions`(), `XmRenditionCreate`(), `XmRenditionRetrieve`(), `XmStringGenerate`(), `XmStringParseText`(), `XmStringPutRendition`(), `XmStringTableParseStringArray`(), `XmStringTableUnparse`(), and `XmStringUnparse`().

XmTab
: Specifies a tab stop, which is used to lay out compound strings within a columns. This data type is an opaque structure returned by a call to `XmTabCreate`(), and is used in calls to the following tab routines: `XmTabGetValues`(), `XmTabFree`(), `XmTabListInsertTabs`()

XmTabList
Specifies a list of tab stops, which are used to lay out compound strings within a columns. This data type is an opaque structure returned by a call to XmTabListInsertTabs(), and is used in calls to the following tab routines: XmTabListReplacePositions(), XmTabListRemoveTabs(), XmTabListGetTab(), XmTabListTabCount(), XmTabListCopy(), XmTabListFree(), and XmTabListInsertTabs().

XmTextBlockRec
A structure that specifies information about a block of text in a Text or TextField widget. The text field in an XmTextVerifyCallbackStruct points to a structure of this type, which is defined as follows in *<Xm/Xm.h>*:

```
typedef struct {
    char          *ptr;     /* pointer to the text to insert      */
    int           length;   /* length of this text                */
    XmTextFormat  format;   /* text format (e.g., FMT8BIT, FMT16BIT) */
} XmTextBlockRec, *XmTextBlock;
```

XmTextBlockRecWcs
A structure that specifies information about a block of text in wide-character format in a Text or TextField widget. The text field in an XmTextVerifyCallbackStructWcs points to a structure of this type, which is defined as follows in *<Xm/Xm.h>:*

```
typedef struct {
    wchar_t    *wcsptr;   /* pointer to text to insert      */
    int        length;    /* length of this text            */
} XmTextBlockRecWcs, *XmTextBlockWcs;
```

XmTextDirection
An enumerated type that specifies the search direction in calls to XmTextFindString() and XmTextFindStringWcs(). It is defined as follows in *<Xm/Xm.h>:*

```
typedef enum {
    XmTEXT_FORWARD,         /* search forward     */
    XmTEXT_BACKWARD         /* search backward    */
} XmTextDirection;
```

XmTextPosition
A long integer, used by Text and TextField routines for determining the position of a character inside the text string.

XmTextSource
A pointer to an opaque structure that specifies a text source. The type is used in calls to XmTextGetSource() and XmTextSetSource().

XmTextType

An enumerated type which specifies the type of data contained within an input stream. It is defined as follows in <*Xm/Xm.h*>:

```
typedef enum {
    XmCHARSET_TEXT,
    XmMULTIBYTE_TEXT,
    XmWIDECHAR_TEXT,
    XmNO_TEXT
} XmTextType;
```

This data type is used in calls to the following compound string routines: `XmParseMappingGetValues`(), `XmParseMappingSetValues`(), `XmStringGenerate`(), `XmStringParseText`(), `XmStringTableParseStringArray`(), `XmStringTableUnparse`(), and `XmStringUnparse`().

XmTextVerifyCallbackStruct

The callback structure passed to the XmNlosingFocusCallback, XmNmodifyVerifyCallback, and XmNmotionVerifyCallback callback routines of Text and TextField widgets. It is defined as follows in <*Xm/Xm.h*>:

```
typedef struct {
    int         reason;     /* reason that callback was called       */
    XEvent      *event;     /* event that triggered callback         */
    Boolean     doit;       /* do the action (True) or undo it (False) */
    long        currInsert; /* the insert cursor's current position  */
    long        newInsert;  /* desired new position of insert cursor */
    long        startPos;   /* start of text to change               */
    long        endPos;     /* end of text to change                 */
    XmTextBlock text;       /* describes the text to insert          */
} XmTextVerifyCallbackStruct, *XmTextVerifyPtr;
```

start_pos *specifies the location at which to start modifying text. start_pos* is unused if the callback resource is XmNmotionVerifyCallback, *and is the same as the current_insert* member if the callback resource is XmNlosingFocusCallback.

end_pos specifies the location at which to stop modifying text (however, if no text was modified, end_pos has the same value as start_pos). end_pos is unused if the callback resource is XmNmotionVerifyCallback, and is the same as the current_insert member if the callback resource is XmNlosingFocusCallback.

XmTextVerifyCallbackStructWcs

The callback structure passed to the XmNmodifyVerifyCallbackWcs of Text and TextField widgets. It is defined as follows in <*Xm/Xm.h*>:

```
typedef struct {
    int             reason;     /* reason that callback was called          */
    XEvent          *event;     /* event that triggered callback            */
    Boolean         doit;       /* do the action (True) or undo it (False)*/
    long            currInsert; /* the insert cursor's current position     */
    long            newInsert;  /* desired new position of insert cursor    */
    long            startPos;   /* start of text to change                  */
    long            endPos;     /* end of text to change                    */
    XmTextBlockWcs  text;       /* describes the text to insert             */
} XmTextVerifyCallbackStructWcs, *XmTextVerifyPtrWcs;
```

All of the fields in this structure are the same as the fields in the XmTextVerify-CallbackStruct *except text*, which points to a XmTextBlockRecWcs *structure*.

XmToggleButtonCallbackStruct

The callback structure passed to ToggleButton callback routines. It is defined as follows in <*Xm/Xm.h*>:

```
typedef struct {
    int         reason;     /* reason that callback was called      */
    XEvent      *event;     /* event that triggered callback        */
    int         set;        /* selection state of the toggle        */
} XmToggleButtonCallbackStruct;
```

XmToggleButtonState

An enumerated type that specifies the state of a ToggleButton. The valid values for the type are:

XmUNSET XmSET XmINDETERMINATE

XmTopLevelEnterCallbackStruct

The callback structure passed to the XmNtopLevelEnterCallback of a DragContext object. It is defined as follows in *<Xm/DragC.h>*:

```
typedef struct {
    int             reason;             /* reason callback was called       */
    XEvent          *event;             /* event that triggered callback    */
    Time            timestamp;          /* timestamp of logical event       */
    Screen          screen;             /* screen of top-level window       */
    Window          window;             /* window being entered             */
    Position        x;                  /* x-coordinate of pointer          */
    Position        y;                  /* y-coordinate of pointer          */
    unsigned char   dragProtocolStyle;  /* drag protocol of initiator       */
    Atom            iccHandle;          /* internal: not documented         */
} XmTopLevelEnterCallbackStruct, *XmTopLevelEnterCallback;
```

XmTopLevelLeaveCallbackStruct

The callback structure passed to the XmNtopLevelLeaveCallback of a DragContext object. It is defined as follows in *<Xm/DragC.h>*:

```
typedef struct {
    int         reason;         /* reason callback was called       */
    XEvent      *event;         /* event that triggered callback    */
    Time        timestamp;      /* timestamp of logical event       */
    Screen      screen;         /* screen of top-level window       */
    Window      window;         /* window being left                */
} XmTopLevelLeaveCallbackStruct, *XmTopLevelLeaveCallback;
```

XmTransferStatus

An enumerated type that specifies the status of a data transfer operation. The value is passed as a parameter to `XmTransferDone`() in order to terminate current data transfer. The valid values for the type are:

XmTRANSFER_DONE_SUCCEED
XmTRANSFER_DONE_CONTINUE
XmTRANSFER_DONE_FAIL
XmTRANSFER_DONE_DEFAULT

XmTraversalDirection

An enumerated type that specifies direction of traversal in a XmTraverseObscuredCallbackStruct. It is defined as follows in <*Xm/Xm.h*>:

```
typedef enum {
    XmTRAVERSE_CURRENT,
    XmTRAVERSE_NEXT,
    XmTRAVERSE_PREV,
    XmTRAVERSE_HOME,
    XmTRAVERSE_NEXT_TAB_GROUP,
    XmTRAVERSE_PREV_TAB_GROUP,
    XmTRAVERSE_UP,
    XmTRAVERSE_DOWN,
    XmTRAVERSE_LEFT,
    XmTRAVERSE_RIGHT
    XmTRAVERSE_GLOBALLY_FORWARD      /* 2.0 */,
    XmTRAVERSE_GLOBALLY_BACKWARD     /* 2.0 */
} XmTraversalDirection;
```

XmTraverseObscureCallbackStruct

The callback structure passed to the XmNtraverseObscuredCallback of a ScrolledWindow widget. It is defined as follows in <*Xm/Xm.h*>:

```
typedef struct {
    int                 reason;         /* reason the callback was called */
    XEvent              *event;         /* event that triggered callback  */
    Widget              traversal_destination;
                                        /* widget or gadget to traverse to */
    XmTraversalDirection direction;     /* direction of traversal         */
} XmTraverseObscuredCallbackStruct;
```

XmVisibility
An enumerated type that specifies the visibility state of a widget. A value of type XmVisibility is returned by `XmGetVisibility`(). It is defined as follows in *<Xm/Xm.h>:*

```
typedef enum {
    XmVISIBILITY_UNOBSCURED,          /* completely visible  */
    XmVISIBILITY_PARTIALLY_OBSCURED,  /* partially visible   */
    XmVISIBILITY_FULLY_OBSCURED       /* not visible         */
} XmVisibility;
```

XrmValue
A structure defined in *<X11/Xresource.h>*, used in `XtConvert`() and other resource conversion routines:

```
typedef struct {
    unsigned int    size;
    XPointer        addr;
} XrmValue, *XrmValuePtr;
```

XrmValuePtr
See XrmValue.

XtAccelerators
A pointer to an opaque internal type, a compiled accelerator table. A pointer to an XtAccelerators structure is returned by a call to `XtParseAcceleratorTable`(). Usually, the compiled accelerator table is produced automatically by resource conversion of a string accelerator table stored in a resource file.

XtCallbackList
A structure defined as follows in *<X11/Intrinsic.h>*:

```
typedef struct _XtCallbackRec {
    XtCallbackProc      callback;
    XtPointer           closure;
} XtCallbackRec, *XtCallbackList;
```

Applications which use `XtAddCallback`() or `XtRemoveCallback`() do not need to use the XtCallbackList type. It can, however, be used to set a callback resource by passing the structure to `XtCreateWidget`() or `XtSetValues`(). Any structure so defined should be declared static. In most documentation, the closure member is referred to as client_data.

XtCallbackProc

The prototype for callback functions. It is defined as follows in *<X11/Intrinsic.h>*:

```
typedef void (*XtCallbackProc) (Widget widget, XtPointer client_data,
XtPointer call_data)
```

XtConvertSelectionIncrProc

The prototype for an incremental selection conversion procedure. The XmNconvertProc for a DragContext object is of this type, which is defined as follows in *<X11/Intrinsic.h>*:

```
typedef Boolean (*XtConvertSelectionIncrProc)(
                                    Widget          widget,
                                    Atom            *selection,
                                    Atom            *target,
                                    Atom            *type_return,
                                    XtPointer       *value_return,
                                    unsigned long   *length_return,
                                    int             *format_return,
                                    unsigned long   *max_length,
                                    XtPointer       client_data,
                                    XtRequestId     *request_id)
```

XtCreatePopupChildProc

The prototype for a procedure that pops up the child of a shell when the shell is popped up. The XmNcreatePopupChildProc resource of Shell specifies a procedure of this type, which is defined as follows in *<X11/Intrinsic.h>*:

```
typedef void (*XtCreatePopupChildProc) (Widget shell)
```

XtKeyProc

The prototype for a keycode-to-keysym translation procedure. `XmTranslateKey`() is the default XtKeyProc for Motif applications. The prototype is defined as follows in *<X11/Intrinsic.h>*:

```
typedef void (*XtKeyProc)(  Display     *display,
                            KeyCode     keycode,
                            Modifiers   modifiers,
                            Modifiers   *modifiers_return,
                            KeySym      *keysym_return)
```

XtOrderProc
: The prototype for a procedure that allows composite widgets to order their children. The XmNinsertPosition resource of Composite specifies a procedure of this type, which is defined as follows in *<X11/Composite.h>*:

```
typedef Cardinal (*XtOrderProc) (Widget child)
```

XtPointer
: A datum large enough to contain the largest of a char*, int*, function pointer, structure pointer, or long value. A pointer to any type or function, or a long, may be converted to an XtPointer and back again and the result will compare equally to the original value. In ANSI-C environments, it is expected that XtPointer will be defined as void *.

XtSelectionCallbackProc
: The prototype for a selection callback procedure. The XmNtransferProc for a DropTransfer object is of this type, and is defined as follows in *<X11/Intrinsic.h>*:

```
typedef void (*XtSelectionCallbackProc)(    Widget          widget,
                                            XtPointer       client_data,
                                            Atom            *selection,
                                            Atom            *type,
                                            XtPointer       value,
                                            unsigned long   *length,
                                            int             *format0
```

XtTranslations
: A pointer to an opaque internal type, a compiled translation table. A pointer to an XtTranslations structure is returned by a call to `XtParseTranslationTable`(). Usually, the compiled translation table is produced automatically by resource conversion of a string translation table stored in a resource file.