## The XsMDICanvas Class

This section describes how to build and manipulate an MDI using the *XsMDICanvas* class. Minimally, you must perform the following actions to build and display an MDI canvas:

1. Create the *XsMDICanvas* object.
2. Create the documents as instances of subclasses of *XsMDIWindow*.
3. Add the documents to the canvas
4. Show the canvas

**Constructor and Destructor:**

The *XsMDICanvas* accepts two arguments:

XsMDICanvas (const char \*name, Widget parent)

The *name* parameter specifies the name of the canvas and is used as the widget name for the underlying implementation. The *parent* parameter specifies the widget that is to be used as the parent of the canvas.

The *XsMDICanvas* destructor destroys the canvas, but it **does not** destroy any of the underlying documents. It is up to the application to destroy these.

**Adding and removing documents:**

After the documents are created, they must be added to the canvas. The *XsMDICanvas::add()* member-function adds documents to the canvas:

virtual void add (XsMDIWindow \*window)

The behaviour of adding the same document to the canvas more than once is undefined. Documents can be removed from the canvas by using:

virtual void remove (XsMDIWindow \*window)

Additionally, all documents can be removed from the canvas with:

void removeAll ( )

The number of documents currently installed in the canvas can be retrieved with:

int numWindows ( ) const

**Showing the canvas:**

In order to show (manage) the canvas call the *show* member function:

virtual void show ( ) This member-function is responsible for cycling all of the installed documents and calling their respective *XsMDIWindow::\_buildClientArea()* member-functions. After each document has been created, *show* will then manage each document and, finally, itself.

**Window Placement:**

The current implementation of *XsMDICanvas* uses a very simple algorithm to place the documents on the canvas. In order to implement a more specific placement algorithm, derive a class from *XsMDICanvas* and override the member-function:

virtual void \_placeWindow (XsMDIWindow \*win)

This member-function is called for each document in the canvas to compute the location of the document. Please refer to the code (*XsMDICanvas.C*) for more details.

**Canvas Behaviour:**

The *XsMDICanvas* is implemented as an *XmScrolledWindow* with an *XmDrawingArea* work-window. The instance name for the work-window is *canvas*. Internal callbacks in the canvas class force the work-window to be at-least the size of the clip-window. This prevents documents from being clipped as they are moved around.

By default, the *XmDrawingArea* work-window has its *XmNresizePolicy* set to *XmRESIZE\_GROW*. This will allow the work-area to grow to whatever size necessary, but it will not automatically shrink as windows are manipulated. If different behaviour is desired, the *XmNresizePolicy* resource on the work-area can be set to *XmRESIZE\_ANY*. This will force the work-window to recompute its size as windows are manipulated, and it will grow and shrink as necessary. However, the *XsMDICanvas* will still force the work-area to be at-least the size of the clip-window.

To change the default behaviour, add the following resource:

<XsMDICanvas name>\*canvas.resizePolicy: XmRESIZE\_ANY