Printing

This chapter describes how application developers can use the X Print Service and printing−related APIs to achieve printing from their applications.

The X printing architecture is fundamentally concerned with reusing most of the X Protocol and X library APIs with an X Server that generates such printer languages as PostScript and PCL (refer to the specification, *X Printing Architecture*). Correspondingly, the model chosen for widget printing is to re−use the code that renders the widgets on the video side (through their *expose* method) for the X Print Server side.

An application therefore needs only to create new instances of widgets using a **Shell** widget created on the X Print Server and to set resources according to what it is to be printed on paper. In addition, depending on the programming model chosen for rendering, the application may instruct the widgets to render themselves manually or use callbacks to change the state of the widget content between pages.

As a result of that design choice, there is no clear boundary between printing only the content of the widgets and printing high−resolution screen dumps of the widget's visual representation. The separation between current content and current visual is left to the programmer, and so widget printing becomes mostly a matter of the programmer deciding which widget resources need to be copied from the video widget instance to the print widget instance. A section in this chapter is intended to help the programmer answer this question for the Motif **XmText** and **XmLabel** widget classes.

As far as rendering the widget on the print side is concerned, two models of programming are supported:

- Synchronous, where the application directs the widget to print itself (using the **XmRedisplayWidget** API)
- Asynchronous, where all drawing/printing happens as a result of *Exposure* and other *Print* events being generated by the X Print Server and automatically dispatched through the **XmPrintShell** callback mechanism (**XmNpageSetUpCallback**).

In addition, a convenience function to set up the X Print Server connection and a top−level **Print** shell widget are provided, as well as a convenience function supporting a print−to−file service. Both APIs are also usable from Motif/CDE applications that do not print using widgets but print by means of direct rendering via *Xlib*.

The Printing APIs

The printing APIs are as follows:

- **XmPrintSetup**
- **XmPrintShell**
- **XmPrintPopupPDM**
- **XmGetScaledPixMap**
- **XmRedisplayWidget**
- **XmPrintToFile**

This section summarizes the APIs and gives examples of their use.

XmPrintSetup registers an X Print Server connection with Xt, sets resources appropriately, and creates an **XmPrintShell** that it returns to the caller. In this example, the print shell is created in the **OKcallback** of a **DtPrintSetupBox**:

```
void OKCallback(Widget w, XtPointer client_data, XtPointer call_data)
{
    DtPrintSetupCallbackStruct *pbs = call_data;
    static Widget shell;
    static int num;          /* app data to print */

    if (!shell) {
      shell =
          XmPrintSetup(widget,
                       XpGetScreenOfContext(pbs->print_display,
                                            pbs->print_context)
                       "Print", NULL, 0);
      XtAddCallback(p->print_shell, XmNpageSetupCallback,
                    PageSetupCB, &num);
      XtAddCallback(p->print_shell, XmNpdmNotificationCallback,
                    PdmNotifyCB, &num);
    }

    num = 0;

    /* check print-to-file & print */
}
```

XmPrintShell encapsulates some of the X Print Service functionalities and provides the framework for the asynchronous printing programming model. Here is an example of a **XmNpageSetupCallback**:

```
void PageSetupCB(Widget w, XtPointer client_data, XtPointer
                 call_data)
{
   Widget print_shell = widget;
   XmPrintShellCallbackStruct* pr_cbs = call_data;
   int *num = client_data;
   static Widget form, text;
   char buf[10];

   /* don't do anything after last page is printed *
   if (pr_cbs->last_page) return;

   /* create the widgets once */
   if (!form) {
      form = XmCreateForm("form", print_shell, NULL, 0);
      text = XmCreateTextField("text", form, NULL, 0);
   }

   /* setup app's data in print text widget */
   sprintf(buf, "%d", *num);
   XmTextSetString(text, buf);
```

```
   num++
   if (num > 9) pr_cbs->last_page = True;
}
```

XmPrintPopupPDM sends a notification to start a print dialog manager on behalf of the application. An example:

```
if (XmPrintPopupPDM(print_shell, top_level) !=
     XmPDM_NOTIFY_SUCCESS)
 {
     /* let user know of problem */
 }
```

XmGetScaledPixMap retrieves a *Pixmap* from an **XBM** or **XPM** file and potentially scales it to match the printer resolution. Again, this example is from a **XmNpageSetupCallback**:

```
int *month = client_data;
static Widget form, label;
Pixmap pixmap;
unsigned long fg, bg;
int depth;

/* create the widgets once */
if (!form) {
   form = XmCreateForm("form", print_shell, NULL, 0);
   label1 = XmCreateLabel("label", form, NULL, 0);
}

XtVaGetValues(label, XmNforeground, &fg,
              XmNbackground, &bg, XmNdepth, &depth);

/* since label is a child of a XmPrintShell, and we have
   specified 0 as the scaling factor, it will be scaled
   appropriately.  Also, the cache will take care of calling
   this more than once */
pixmap = XmGetScaledPixmap(label,
                           ((*month) ? "month.xpm": "week.xpm"),
                           fg, bg, depth, 0);
```

XmRedisplayWidget calls the *expose* method of a widget in order to draw its content; for example:

```
XmTextScroll(text, lines_per_page);
XmRedisplayWidget(text);
```

XmPrintToFile retrieves the data being sent by a Print Server and prints it to a file on the client side. What follows is the remaining part of the **OKCallback** above:

```
int save_data = XPSpool;

if (pbs->destination == DtPRINT_TO_FILE)
save_data = XPGetData;

/* start job must precede XpGetDocumentData in XmPrintToFile */
XpStartJob(XtDisplay(print_shell), save_data);

/* setup print to file */
if (pbs->destination == DtPRINT_TO_FILE)
{
   if (!XmPrintToFile(XtDisplay(print_shell),
                      pbs->dest_info, FinishPrintToFile, NULL))
      {
        /* output some kind of error message */
       XpCancelJob(XtDisplay(print_shell), True);

       /* go back to the event loop as if we had never printed */
          return;
      }
}
```

Motif Widget Printing

In this section, we examine the Motif widgets (mainly, **XmText** and **XmLabel**) and give hints about which widget resources need to be turned on or off in the printing case. This section also gives examples of sample code that copies data from a video widget to a corresponding print widget.

The assumption we make in this discussion is that we want to print the content of the widgets, rather than the visual representation of the widgets; in other words, our goal is not to print high−resolution screen dumps. Although it's impossible to define what constitutes the content of an arbitrary widget class (that is a sufficiently difficult task for the widgets we know), we can start by pointing out the resources that are important visually in Xt/Motif, in order to help the programmer set or unset them appropriately for printing.

Purely Visual Resources

At the **Core** class level, there are **XmNbackground**, which the application will probably want to set to white for paper output, and **XmNborderWidth**, which should be *0* (its default value).

For **XmPrimitive**, the **XmNshadowThickness** and **XmNhighlightThickness** resources are probably good candidates for *0* value as well, unless some three−dimensional effect is intended on the paper. **XmNforeground** should probably be black.

For the **XmText** and **XmTextField** widgets, one obvious setting is to turn off the blinking of the I−beam cursor. This happens automatically (refer to default dynamic and **XmNcursorPositionVisible** == False) if the widget is rooted to a **XmPrintShell**.

The print case almost certainly shouldn't print the scrollbars in a **ScrolledText** instance. The **XmNscrollHorizontal** and **XmNscrollVertical** resources can be set to *False* to suppress the printing of scrollbars. Note that, as a rule, it is better to avoid **ScrolledText** altogether in the case of printing.

For the **Text** widget, **XmMwordWrap** is a resource that deserves some attention; it's up to the programmer to decide what the application should do with it. If a wordwrap−on video text buffer is transferred to a print text buffer using (as is likely to be the case) a different character size, font, resolution, and so on (with wordwrap on as well), the number of pages will be different in the print and video cases. That's probably be fine since the number of pages on the video side should not be treated as content to be printed out: they are virtual pages, after all. If the video text has wordwrap off, the sequence of lines on the print side will be the same, and it's up to the application or its user to guarantee that this layout is appropriate for the print font, paper size, and so on, because no new formatting is done. The application can decide to take a wordwrap −on video text buffer and generate explicit newlines before passing it to the print text, so that the same layout is preserved. However, the application does so at the risk of allowing the content to run off the edge of the paper).

Finally, the **Text** widget also carries internal margins (**XmNmarginWidth** and **XmNmarginHeight**) that the print code might want to reset before printing.

For the **XmLabel** widget, the important visual resources are margins and alignment, since shadows and highlights are covered by **XmPrimitive**. **XmLabel** carries six margins resources (**XmNmarginWidth**, **XmNmarginHeight**, **XmNmarginTop**, **XmNmarginBottom**, **XmNmarginRight**, **XmNmarginLeft**) and the **XmNaligment** resource. These resources are mostly useful when the widget holds a border, highlight, or shadow of some kind. If borders are not to be transferred, there is no real need to transfer a margin either.

Content Resources

In connection with defining the content for **XmText** and **XmLabel**, the resource settings that the programmer would probably want to transfer to the print widget are the following: **XmNvalue** and **XmNvalueWcs** for the text widgets (or by using **XmTextGetString/XmTextSetString**). The **XmText** resource **XmNtopCharacter** may be relevant if the current page only is to be printed. **XmTextSetHighlight** can be used to highlight some text in a text widget.

For **XmLabel**, **XmNlabelType** (**PIXMAP** or **STRING**), and **XmNlabelString** or **XmNlabelPixmap** are important resources.

In both the label and text cases, the setting of **XmNstringDirection** is probably inherited from the environment, so it's probably best to leave it alone.

Examples

In the following examples, a simple widget initialize method wants to know whether it is creating a widget for printing or video.

```
Widget CopyText(print_parent, video_text...)
/*-------------*/
{
    /* get the content out of the video text */
    buffer = XmTextGetString (video_text);
     /* now create the print instance */
    ptext = XtVaCreateWidget(print_parent, ...,
                             XmNvalue, buffer, NULL);
    XtFree(buffer);
    return ptext;

}

Widget CopyLabel(print_parent, video_label...)
/*-------------*/
{
    /* get the string content out of the video label */
    XtVaGetValues(video_label, XmNlabelString, &buffer, NULL);
     /* now create the print instance */
    plabel = XtVaCreateWidget(print_parent, ...,
                              XmNlabelString, buffer, NULL);
    XtFree(buffer);
    return plabel;

}
```