SunSoft, Inc.
2550 Garcia Avenue
Mountain View, CA 94043

For more information, call 1 800 227-9227.

Printed in USA 8/92 92205-001 1.5K

# *Table of Contents*

# Tool Inter-Operability: A Hands On Demonstration

## A Simple Demonstration of How the ToolTalk Service Works

*A White Paper*

SunSoft
A Sun Microsystems Company

*Code Example 4* Modified Xfontsel.c File

```
    /*
     * This is not a message we recognize.
     * If it’s a request, or a notice that caused us to start, fail it.
     */

    if (tt_message_class(incoming) == TT_REQUEST ||
     tt_message_status(incoming) == TT_WRN_START_MESSAGE) {
        tt_message_fail(incoming);
    }
    tt_message_destroy(incoming);
    tt_release(ttmark);
}

/*
 * Called when the Apply button is pressed. Replies to the outstanding
 * message and turn off the Apply button.
 */

/* ARGSUSED */
void ReplyToMessage(w, msg, wdata)
 Widget w;
 caddr_t msg;
 caddr_t wdata;
{

    tt_message_arg_val_set(replymsg, 0, currentFontNameString);
    tt_message_reply(replymsg);
    tt_message_destroy(replymsg);
    replymsg = 0;
    XtVaSetValues(applyButton, XtNsensitive, FALSE, 0);
}
```

*Code Example 4* Modified Xfontsel.c File

```
        fprintf(stderr,"%s returned ToolTalk error: %s\n",
            procname, tt_status_message(errid));
        exit(1);
    }
}

void processToolTalkMessage()                            /* Process message */
{
    int ttmark;                                          /* ToolTalk mark */
    Tt_message incoming;                                 /* Incoming message */

    ttmark = tt_mark();

    incoming = tt_message_receive();                     /* Receive message */
    /*
     * It's possible that the file descriptor may become active but
     * there's not actually a ToolTalk message for us.
     */
    if (incoming == 0) return;

    if (tt_is_err(tt_ptr_error(incoming))) {
        dieFromToolTalkError("tt_message_receive",
                 tt_ptr_error(incoming));
    }

    if (0==strcmp(tt_message_op(incoming),"GetFontName")) {
        /*
         * This is the message we expected. If we're already
         * busy, reject it. Otherwise activate the "apply" button.
         */
        if (replymsg) {
            tt_message_reject(incoming);
            tt_message_destroy(incoming);
            tt_release(ttmark);
            return;
        }
        XtVaSetValues(applyButton, XtNsensitive, TRUE, 0);
        replymsg = incoming;
        tt_release(ttmark);
        return;
    }
```

*Code Example 4* Modified Xfontsel.c File

```
 XtAppMainLoop(appCtx);

 tt_close();                                          /* End ToolTalk session */
 tt_release(ttmark);                                  /* Free space */
}

...
       Boolean field_bits[FIELD_COUNT];
       int max_field;
       if (*fontName == DELIM) field++;
/*
 *
 * For the ToolTalk demonstration, use the standard routines
 * instead of BSD; change the line
 * bzero( field_bits, sizeof(field_bits) );
 * to read as follows:
 *
 */
       memset( field_bits, 0, sizeof(field_bits) );
       if (Matches(pattern, fontName++, field_bits, &max_field)) {

...

     XtDisownSelection(w, XA_PRIMARY, time);
    XtSetSensitive(currentFontName, False);
 }

/*
 * For the ToolTalk Demonstration, add the following lines:
 */
}

void dieFromToolTalkError(procname, errid)            /* Fail if error occurs */
 char *procname;                                      /* Process name */
 Tt_status errid;                                     /* Error identifier */
{
    /*
     * Don't die on warnings or TT_OK.
     */

    if (tt_is_err(errid)) {
```

*Code Example 4* Modified Xfontsel.c File

```
   int f;
   for (f = 0; f < FIELD_COUNT; f++) currentFont.value_index[f] = -1;
}

/*
 * For the ToolTalk demonstration, add the following lines
 * to make the top of stack the ToolTalk session and set
 * it to be the default session.
 */
ttmark = tt_mark();
ttrc = tt_default_session_set(                    /* set the default session .. */
tt_X_session(                                     /* .. to the X session for .. */
DisplayString(                                    /* .. the X server displaying ..*/
XtDisplay(top))));                                /* .. our top window... */
   /*
    * Fail if no default session
    */
dieFromToolTalkError("tt_default_session_set",ttrc);
procid = tt_open();
   /*
    * Fail if no proces identifier
    */
dieFromToolTalkError("tt_open",tt_ptr_error(procid));
ttfd = tt_fd();
   /*
    * Fail if no ToolTalk file descriptor
    */
dieFromToolTalkError("tt_fd",tt_int_error(ttfd));
ttrc = tt_ptype_declare("xfontsel");
   /*
    * Fail if ptype not declared
    */
dieFromToolTalkError("tt_ptype_declare",tt_int_error(ttfd));
ttrc = tt_session_join(tt_default_session());
   /*
    * Fail if unable to join session
    */
dieFromToolTalkError("tt_session_join",ttrc);
   /*
    * Add input
    */
XtAddInput(ttfd, (XtPointer)XtInputReadMask, processToolTalkMessage, 0);
```

*Code Example 4* Modified Xfontsel.c File

```
 * as follows:
 */
   Widget /* commandBox, fieldBox, currentFontName,*/ viewPort;

   commandBox = XtCreateManagedWidget("commandBox",formWidgetClass,pane,NZ);
   {

...

    ownButton =
       XtCreateManagedWidget("ownButton",toggleWidgetClass,commandBox,NZ);
/*
 * For the ToolTalk demonstration, add the following lines to create
 * an apply button for the font change:
 */
    applyButton =
    XtVaCreateManagedWidget("applyButton",
                  commandWidgetClass,
                  commandBox,
                  XtNlabel, "apply",
                  XtNfromHoriz, ownButton,
                  XtNleft, XtChainLeft,
                  XtNright, XtChainLeft,
                  0);

    countLabel =
       XtCreateManagedWidget("countLabel",labelWidgetClass,commandBox,NZ);

    XtAddCallback(quitButton, XtNcallback, Quit, NULL);
    XtAddCallback(ownButton,XtNcallback,OwnSelection,(XtPointer)True);
/*
 * For the ToolTalk demonstration, add the following line to notify
 * Xedit when the apply button has been pressed:
 */
    XtAddCallback(applyButton, XtNcallback,ReplyToMessage, NULL);
   }

   fieldBox = XtCreateManagedWidget("fieldBox", boxWidgetClass, pane, NZ);

...

 {
```

*Code Example 4* Modified Xfontsel.c File

```
int matchingFontCount;
static Boolean anyDisabled = False;
Widget ownButton;
/*
 * For the ToolTalk demonstration, add the next line to add
 * the apply button to change the font:
 */
Widget applyButton;                                  /* Add apply button */
Widget fieldBox;
/*
 * For the ToolTalk demonstration, add the next line to add
 * a command box to make the font change:
 */
Widget commandBox;                                   /* Make commandBox global */
Widget countLabel;

...

void main(argc, argv)
 unsigned int argc;
 char **argv;
{
 Widget topLevel, pane;

/*
 * For the ToolTalk demonstration, add the following lines:
 */
 int ttmark, ttfd;                          /* ToolTalk mark, ToolTalk file descriptor */
 char *procid;                              /* Process identifier */
 Tt_status ttrc;                            /* ToolTalk status */

 topLevel = XtInitialize( NULL, "XFontSel", options, XtNumber(options),
            &argc, argv );

...

 pane = XtCreateManagedWidget("pane",panedWidgetClass,topLevel,NZ);
 {
/*
 * For the ToolTalk demonstration, make the command box widget
 * global; change the line
 * Widget commandBox, /* fieldBox, currentFontName,*/ viewPort;
```

## *Adding ToolTalk Code to the Xfontsel Files*

The changes made to the Xfontsel files are described in the "Modifying the Xfontsel Application" section.

*Code Example 4* Modified Xfontsel.c File

```
#ifndef lint
static char Xrcsid[] = "$XConsortium: xfontsel.c,v 1.16 89/12/12 14:10:48 rws
 Exp $";
#endif

...

#include <X11/Xaw/Viewport.h>
#include <X11/Xmu/Atoms.h>

/*
 * For the ToolTalk demonstration, add the following line to include
 * the ToolTalk header file:
 */
#include <desktop/tt_c.h>                              /* ToolTalk header file */

#define MIN_APP_DEFAULTS_VERSION 1

...

void SetCurrentFont();
Boolean IsXLFDFontName();

/*
 * For the ToolTalk demonstration, add the following lines to tell
 * Xfontsel how to handle a ToolTalk message:
 */
void dieFromToolTalkError();                           /* Fail if error occurs */
void processToolTalkMessage();                         /* Process ToolTalk message */
void ReplyToMessage();                                 /* Reply to ToolTalk message */
Tt_message replymsg;

typedef void (*XtProc)();

...
```

*Code Example 3* Modified commands.c File

```
}

void
DoChangeFont()                                      /* Change font */
{
    Tt_message m;                                   /* ToolTalk message */
    Tt_status ttrc;                                 /* ToolTalk status */

        /*
         * Create request
         */
    m = tt_prequest_create(TT_SESSION, “GetFontName”);
        /*
         * Add arguments to message
         */
    tt_message_arg_add(m,TT_OUT,”string”,(char *)NULL);
        /*
         * Add callback to notify when change complete
         */
    tt_message_callback_add(m,FinishChangeFont);
        /*
         * Send message
         */
    ttrc = tt_message_send(m);
        /*
         * Fail if error occurs
         */
    dieFromToolTalkError(“tt_message_send”,ttrc);
}



void
DoSave()
{
```

*Code Example 3* Modified commands.c File

```
}
#endif

/*
 * For the ToolTalk demonstration, add the following lines to have Xfontsel
 * send a callback that the operation has either completed or failed:
 */
static Tt_callback_action FinishChangeFont(m,p)         /* ToolTalk message callback */
 Tt_message m;                                          /* ToolTalk message */
 Tt_pattern p;                                          /* ToolTalk pattern */
{
    static XFontStruct *fs;                             /* Font structure */
    int ttmark;                                         /* ToolTalk mark */

    ttmark = tt_mark();                                 /* ToolTalk mark */
        /*
         * If operation fails, notify user
         */
    if (TT_FAILED==tt_message_state(m)) {
        XeditPrintf("Font change failed\n");
        tt_message_destroy(m);                          /* Destroy message */
    } else if (TT_HANDLED==tt_message_state(m)) {
        XFontStruct *newfs;
        /* Try to load the new font */
        newfs = XLoadQueryFont(CurDpy,tt_message_arg_val(m,0));
        /* If the new font is OK, and there is an old font,
         * unload the old font. Then use the new font
         */
        if (newfs) {
            if (fs) {
                XUnloadFont(CurDpy, fs->fid);
            }
            XtVaSetValues(textwindow, XtNfont, newfs, 0);
            fs = newfs;
        }
        tt_message_destroy(m);                          /* Destroy message */
    }
    tt_release(ttmark);                                 /* Release mark */
        /*
         * Process callback to notify sender operation completed
         */
    return TT_CALLBACK_PROCESSED;
```

*Code Example 2* Modified Xedit.c File

```
   tt_release(ttmark);                                    /* Free space */
 }

 void dieFromToolTalkError(procname, errid)
 char *procname;
 Tt_status errid;
                                                          /* Fail if error occurs */
 {
   /*
    * Don’t die on warnings or TT_OK.
    */

   if (tt_is_err(errid)) {
       fprintf(stderr,”%s returned ToolTalk error: %s\n”,
           procname, tt_status_message(errid));
       exit(1);
   }
 }
```

*Code Example 3* Modified commands.c File

```
#if (!defined(lint) && !defined(SABER))
static char Xrcsid[] = “$XConsortium: commands.c,v 1.27 89/12/10
                  17:08:26 rws Exp $”;
#endif /* lint && SABER */

...

#ifdef USG
int rename (from, to)
 char *from, *to;
{
 (void) unlink (to);
 if (link (from, to) == 0) {
 unlink (from);
 return 0;
 } else {
 return -1;
 }
```

*Code Example 2* Modified Xedit.c File

```
...

void
Feep()
{
XBell(CurDpy, 0);
/*
 *For the ToolTalk demonstration, add the following lines to receive and
 * process an incoming message:
 */
}


void processToolTalkMessage()                    /* Process ToolTalk message */
{
   int ttmark;                                   /* ToolTalk mark */
   Tt_message incoming;                          /* Incoming message */

   ttmark = tt_mark();                           /* ToolTalk mark */

   incoming = tt_message_receive();              /* Receive incoming message */
   /*
    * The callback should process the message, so we should never
    * get a message returned.
    */
   if (incoming == 0) return;                    /* Return incoming message */

   if (tt_is_err(tt_ptr_error(incoming))) {
       dieFromToolTalkError("tt_message_receive",
               tt_ptr_error(incoming));
   }

   /*
    * This is not a message we recognize.
    * If it is a request, or a notice that caused us to start, fail it.
    */

   if (tt_message_class(incoming) == TT_REQUEST ||
    tt_message_status(incoming) == TT_WRN_START_MESSAGE) {
       tt_message_fail(incoming);
   }
   tt_message_destroy(incoming);                 /* Destroy message */
```

*Code Example 2* Modified Xedit.c File

```
 * to make the top of stack the ToolTalk session and set
 * it to be the default session.
 */
ttmark = tt_mark();
ttrc = tt_default_session_set(                /* set the default session .. */
tt_X_session(                                 /* .. to the X session for .. */
DisplayString(                                /* .. the X server displaying ..*/
XtDisplay(top))));                            /* .. our top window... */
      /*
       * Fail if no default session
       */
dieFromToolTalkError("tt_default_session_set",ttrc);
procid = tt_open();                           /* Initialize ToolTalk */
      /*
       * Fail if no process identifier
       */
dieFromToolTalkError("tt_open",tt_ptr_error(procid));
ttfd = tt_fd();                               /* ToolTalk file descriptor */
      /*
       * Fail if no file descriptor
       */
dieFromToolTalkError("tt_fd",tt_int_error(ttfd));
      /*
       * Activate file descriptor
       */
XtAddInput(ttfd, (XtPointer)XtInputReadMask, processToolTalkMessage, 0);

XtMainLoop();
}

...

MakeCommandButton(b_row, "load", DoLoad);
/*
 * For the ToolTalk demonstration, add the next line to make
 * a button for the font change command:
 */
MakeCommandButton(b_row, "changefont", DoChangeFont);
filenamewindow = MakeStringBox(b_row, "filename", filename);
}
XtCreateManagedWidget("bc_label", labelWidgetClass, outer, NULL, ZERO);
```

*Code Example 1* Modify the Xedit.h File

```
...

 extern void DoChangeFont();                               /* Change font */
```

*Code Example 2* Modified Xedit.c File

```
#if (!defined(lint) && !defined(SABER))
 static char Xrcsid[] = "$XConsortium: xedit.c,v 1.23 89/12/07
 19:19:17 kit Exp $";
 #endif    /* lint && SABER */

...

 void
 main(argc, argv)
 int argc;
 char **argv;
 {
 Widget top;
 String filename = NULL;
 static void makeButtonsAndBoxes();

 /*
  * For the ToolTalk demonstration, add the following lines:
  */
 int ttmark;                                            /* ToolTalk mark */
 int ttfd;                                              /* ToolTalk file descriptor */
 char *procid;                                          /* Process identifier */
 Tt_status ttrc;                                        /* ToolTalk status */

 top = XtInitialize( "xedit", "Xedit", NULL, 0, &argc, argv);

...

 XtRealizeWidget(top);
 XDefineCursor(XtDisplay(top),XtWindow(top),
        XCreateFontCursor( XtDisplay(top), XC_left_ptr));
 /*
  * For the ToolTalk demonstration, add the following lines
```

## *Adding ToolTalk Code to the Xedit Files*

The changes made to the Xedit files are described in the “Modifying the Xedit Application” section.

*Code Example 1* Modify the Xedit.h File

```
 /*
 *  rcs_id[] = "$XConsortium: xedit.h,v 1.18 89/07/21 19:52:58 kit Exp $";
 */

...

 #include <X11/Xaw/Viewport.h>
 #include <X11/Xaw/Cardinals.h>

 /*
 * For the ToolTalk demonstration, add the following include line.
 */
 #include <desktop/tt_c.h>                                 /* ToolTalk header */

 extern struct _app_resources {
 Boolean enableBackups;
 char *backupNamePrefix;
 char *backupNameSuffix;

...

 /* externals in xedit.c */

 extern void Feep();
 /*
 * For the ToolTalk demonstration, add the following externals.
 */
 extern void processToolTalkMessage();                     /* Process ToolTalk message */
 extern void dieFromToolTalkError();                       /* Fail if error occurs */
 extern Display *CurDpy;                                   /* Display */

...

 /* externs in commands.c */
```

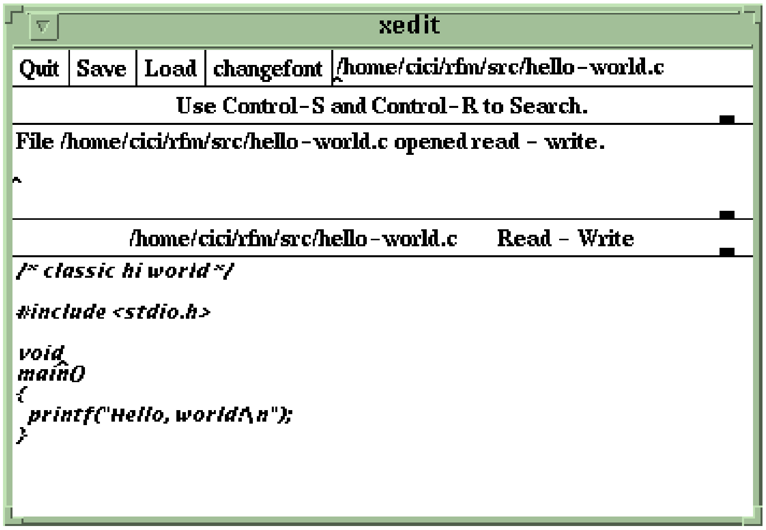


*Figure 5* The Xedit Screen Showing the Font Change

## *Adding ToolTalk Code to the Demonstration Applications*

This section illustrates how ToolTalk code was added to the Xedit and Xfontsel applications.

- The ellipses (. . .) indicate code has been skipped. The code examples only show a few lines of code proceeding and preceding the inserted ToolTalk lines of code.
- The actual ToolTalk code that you need to add to the files is shown in bold type; for example:

```
#include <desktop/tt_c.h>    /* ToolTalk header */
```

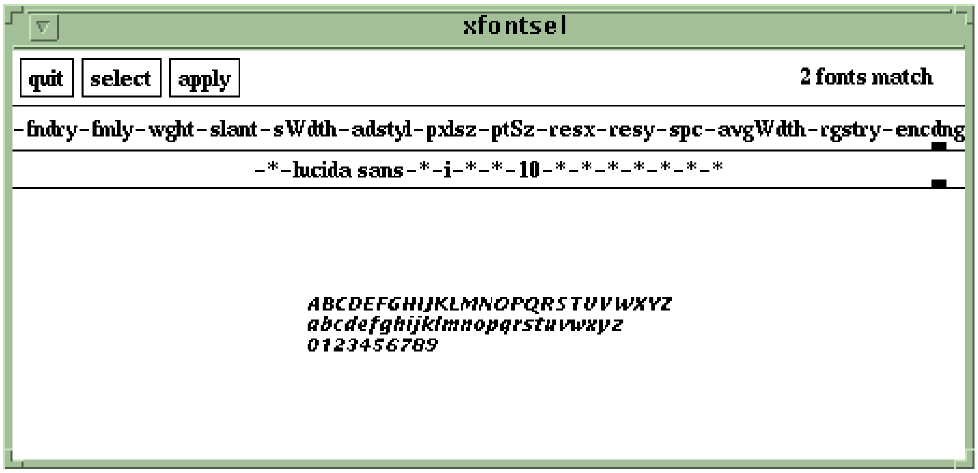


*Figure 4* The Xfontsel Screen

**b. Select the new font on the Xfontsel screen.**

**c. Click in the 'apply' button on the Xfontsel screen.**
A screen similar to the one shown in Figure 9 is displayed.

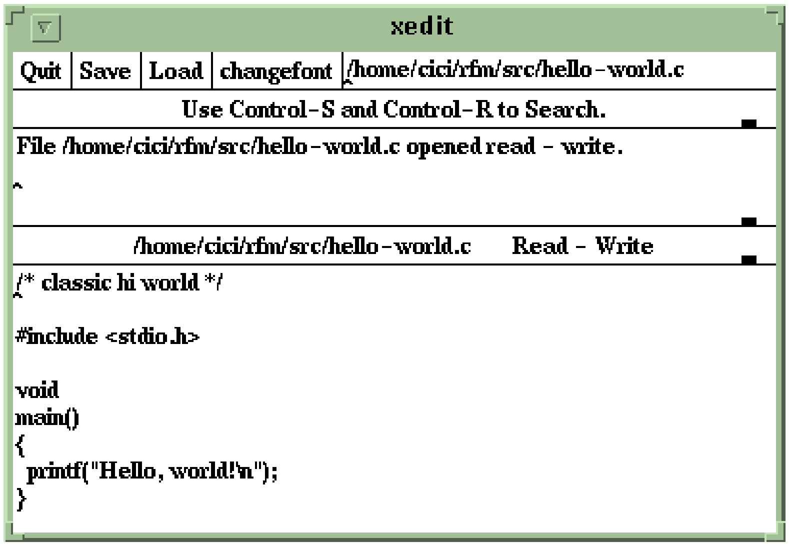


*Figure 3* The Xedit Screen with a File Loaded

**3. Change the displayed font.**

**a. Click in the 'changefont' button on the Xedit screen.**

A screen similar to the one shown in Figure 4 is displayed.

## *We Have Tool Communication!*

You are now ready to see how the ToolTalk technology works.

1. **Start the Xedit application.**
   To start Xedit, enter the command as follows:

```
machine_name% xedit
```

   A screen similar to the one shown in Figure 2 is displayed.

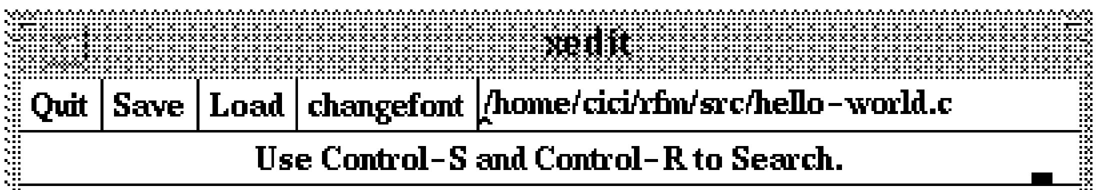


*Figure 2* The Xedit Screen

2. **Load a file.**

   A screen similar to the one shown in Figure 3 is displayed.

```
ptype xfontsel {/* Process type identifier */
    start "/directory_name/xfontsel";/* Start string */


 handle:   /* Receiving process */
        /* A signature is divided
         * into two parts by the => as follows:
         * Part 1 specifies how the message is to be matched;
         * Part 2 specifies the what special is to be taken when
         * a match occurs.
         */
 session GetFontName(out string fontname) => start;


}
```

*Figure 1* The Xfontsel.ptype File

When your tool declares a ptype, the message patterns listed in it are automatically registered; the ToolTalk service then matches messages it receives to these registered patterns. These static message patterns remain in effect until your tool closes communication with the ToolTalk service.

After you have created the ptype file, you need to install the ptype. To do this, run the ToolTalk type compiler as follows:

```
machine_name% tt_type_comp xfontsel.ptype
```

where *xfontsel.ptype* is the name of your ptype file.

After you have made the indicated changes to the Xfontsel files, created a ptype file, and installed the pytpe, compile the Xfontsel program.

## *Modifying the Xfontsel Application*

To modify the Xfontsel application so that it will be able to communicate with the Xedit application, you need to modify the following files:

- the Xfontsel.c file
- the Makefile

You also need to create a new file to declare the ToolTalk process type (ptype) for the Xfontsel application.

For the ToolTalk demonstration, Xfontsel needs to know:

- where to find the ToolTalk header file
- how to handle a ToolTalk message when it receives one
- how to process an error caused by a ToolTalk message
- how to behave when the apply button is activated for the new change fonts command

Xfontsel also needs to display an apply button and a command box to make the font change. In addition, you need to add code to tell Xfontsel when to send a ToolTalk callback message, and how to join the ToolTalk session. These modifications are made in the Xfontsel.c file, as shown in Code Example 4 on page 17 with commented explanations.

Next, modify the the Makefile for the Xfontsel program so that it uses the ToolTalk libraries. To do this, add the `-ltt` option to the as follows:

```
LOCAL_LIBRARIES = -ltt $(XAWLIB) $(XMULIB) $(XTOOLLIB) $(XLIB)
```

The ToolTalk types mechanism is designed to help the ToolTalk service route messages. You first define a process type (ptype), and then compile the ptype with the ToolTalk type compiler, tt_type_comp. For the ToolTalk demonstration, you need to create a ptype file for the Xfontsel application, as shown in Figure 1.

---

**Note –** *directory_name* is the pathname to the directory in which the modified Xfontsel files reside.

---

## *Modifying the Xedit Application*

To modify the Xedit application so that it will be able to communicate with the Xfontsel application, you need to modify the following files:

- the Xedit.h file
- the Xedit.c file
- the commands.c file
- the Makefile

For the ToolTalk demonstration, Xedit needs to know about the ToolTalk header file. Xedit also needs to know about the new ToolTalk commands in the `Xedit.c` file. These changes are made to the xedit.h file, as shown with commented explanations in Code Example 1 on page 10.

Next, you need to add code to the `Xedit.c` file to set the ToolTalk session, make a button for the font change function, and to allow Xedit to receive and process ToolTalk messages. These changes to the file are shown with commented explanations in Code Example 2 on page 11.

Now add code to the `commands.c` file so that Xedit can tell the Xfontsel application to send a reply when the font change has been completed, or to notify it if the operation failed. You also need to add code that tells Xfontsel what operation Xedit wants performed. These changes to the file are shown with commented explanations in Code Example 3 on page 14.

The final modification you need to make to the Xedit program is to change the the Makefile so that it uses the ToolTalk libraries. To do this, add the `-ltt` option as follows:

```
LOCAL_LIBRARIES = -ltt $(XAWLIB) $(XMULIB) $(XTOOLLIB) $(XLIB)
```

After you have made the indicated changes to the Xedit files, compile the Xedit program.

## Inter-Application Communication Made Easy

The ToolTalk service provides you with a complete set of functions for application integration. Using the functionality provided with the ToolTalk service, existing applications can be made to "speak" to each other. This paper shows you how to try out this technology for yourself.

The demonstration of the ToolTalk service is simple: while using a simple text editor, you can ask an interface for selecting font names to change the font displayed in the loaded file. The ToolTalk demo consists of two applications from X11R4:

- Xedit – a simple text editor for X
  found in `/usr/lib/X11/app-defaults/Xedit`
- Xfontsel – a point-and-click interface for selecting X11 font names
  found in `$XFILESEARCHPATH/XFontSel`

This paper outlines the simple steps to modify these two applications so that they can inter-operate; the "Adding ToolTalk Code to the Demonstration Applications" section shows how the ToolTalk code is incorporated into the source code files.

## Adding Inter-Operability Functionality

Before the tools are able to inter-operate, you need to make modifications to the .c and Makefiles for each of the applications; and to the header file for the Xedit application. You also need to create a new file to declare the ToolTalk process type (ptype) for the Xfontsel application.

Use any standard editor, such as *vi*, to make these modifications and to create the ptype file.

# *A Simple Demonstration of How the ToolTalk Service Works*

*Astrid Julienne and Richard McAllister*

## *What is Tool Inter-Operability?*

Tool inter-operability is the control integration of multiple applications in a distributed environment. As computer users increasingly demand that indendently developed appplications work together, inter-operability is becoming an important theme for software developers. By cooperatively using each other's facilities, inter-operating applications offer users capabilities that would be difficult to provide in a single application. This article presents a simple demonstration of this exciting technology of the future to show you how the ToolTalk™ service can enable your application to communicate with other applications *today*.

## *Introducing the ToolTalk Service*

The ToolTalk service is designed to facilitate the development of inter-operating applications that serve individuals and work groups. The ToolTalk service enables independent applications to communicate with other applications without having direct knowledge of each other. Applications create and send ToolTalk messages to communicate. Applications pass these messages to the ToolTalk service, which then determines the recipients and delivers the messages to the appropriate application.

The ToolTalk service, from SunSoft™ a Sun Microsystems Company, is also available on a number of platforms.