Participating in ToolTalk Sessions 7

This chapter provides instructions on how to participate in a ToolTalk session. It also shows you how to manage storage of values passed in from the ToolTalk service and how to handle errors that the ToolTalk service returns.

To use the ToolTalk service, your application calls ToolTalk functions from the ToolTalk API library. To modify your application to use the ToolTalk service, you must first include the ToolTalk API header file in your program. After you have initialized the ToolTalk service and joined a session, you can join files and additional user sessions. When your process is ready to quit, you unregister your message patterns and leave your ToolTalk session.

7.1 Including the ToolTalk API Header File

To modify your application to use the ToolTalk service, first you must include the ToolTalk API header file `tt_c.h` in your program. This file resides in the `$TTHOME/include/desktop/` directory.

The following code sample shows how a program includes this file.

```
#include <stdio.h>
#include <sys/param.h>
#include <sys/types.h>

#include <desktop/tt_c.h>
```

7.2 Registering with the ToolTalk Service

Before you can participate in ToolTalk sessions, you must register your process with the ToolTalk service. You can either register in the ToolTalk session in which the application was started (the *initial session*), or locate another session and register there.

The ToolTalk functions you need to register with the ToolTalk service are shown in Table 7-1.

Table 7-1 Registering with the ToolTalk Service

Return Type	ToolTalk Function	Description
char *	tt_open(void)	Process identifier
int	tt_fd(void)	File descriptor
char *	tt_X_session(const char *xdisplay)	Return the session identifier of the specified X display server.
Tt_status	tt_default_session_set(const char *sessid)	Sets the session to which tt_open will connect.

Registering in the Initial Session

To initialize and register your process with the initial ToolTalk session, your application needs to obtain a process identifier (*procid*). You can then obtain the file descriptor (*fd*) that corresponds to the newly initialized ToolTalk process.

The following code sample first initializes and registers the sample program with the ToolTalk service, and then obtains the corresponding file descriptor.

```
int ttfd;
char   *my_procid;

/*
 * Initialize ToolTalk, using the initial default session
 */

my_procid = tt_open();

/*
 * obtain the file descriptor that will become active whenever
 * ToolTalk has a message for this process.
 */

ttfd = tt_fd();
```

`tt_open` returns the procid for your process and sets it as the default procid; `tt_fd` returns a file descriptor for your current procid that will become active when a message arrives for your application.

Caution – Your application must call `tt_open` before other `tt_` calls are made; otherwise, errors may occur. However, there are two exceptions: `tt_default_session_set` and `tt_X_session` can be called before `tt_open`.

When `tt_open` is the first call made to the ToolTalk service, it sets the initial session as the default session. The default session identifier (*sessid*) is important to the delivery of ToolTalk messages. The ToolTalk service automatically fills in the default sessid if an application does not explicitly set the session message attribute. If the message is scoped to `TT_SESSION`, the message will be delivered to all applications in the default session that have registered interest in this type of message.

Registering in a Specified Session

To register in a session other than the initial session, your program must find the name of the other session, set the new session as the default, and register with the ToolTalk service.

The following code sample shows how to join an *X* session named `somehost:0` that is not your initial session.

```
char   *my_session;
char   *my_procid;

my_session = tt_X_session("somehost:0");
tt_default_session_set(my_session);
my_procid = tt_open();
ttfd = tt_fd();
```

Note – The required calls must be in the specified order.

1. `tt_X_session();`
   This call retrieves the name of the session associated with an X display server. `tt_X_session()` takes the argument

   ```
   char *xdisplay_name
   ```

   where `xdisplay_name` is the name of an X display server (in this example, `somehost:0`, `:0`).

2. `tt_default_session_set();`
   This call sets the new session as the default session.

3. `tt_open();`
   This call returns the procid for your process and sets it as the default procid.

4. `tt_fd();`
   This call returns a file descriptor for your current procid.

Registering in Multiple Sessions

There may be cases when you want to send and receive your messages in different sessions. To register in multiple sessions, your program must find the identifiers of the sessions to which it wants to connect, set the new sessions, and register with the ToolTalk service.

The following code sample shows how to connect *procid* to *sessid1*, and *procid2* to *sessid2*.

```
tt_default_session_set(sessid1);
my_procid1 = tt_open();
tt_default_session_set(sessid2);
my_procid2 = tt_open();
tt_fd2 = tt_fd();
```

You can then use tt_default_procid_set to switch between the sessions.

7.3 Setting Up to Receive Messages

Before your application can receive messages from other applications, you must set up your process to watch for arriving messages. When a message arrives for your application, the file descriptor becomes active. The code you use to alert your application that the file descriptor is active depends on how your application is structured.

For example, a program that uses the XView notifier, through the `xv_main_loop` or `notify_start` calls, can have a callback function invoked when the file descriptor becomes active. The following code sample invokes `notify_set_input_func` with the handle for the message object as a parameter.

```
/*
 * Arrange for XView to call receive_tt_message when the
 * ToolTalk file descriptor becomes active.
 */
notify_set_input_func(base_frame,
       (Notify_func)receive_tt_message,
       ttfd);
```

Table 7-2 describes various window toolkits and the call used to watch for arriving messages.

Table 7-2 Code Used to Watch for Arriving Messages

Window Toolkits	Code Used
XView	notify_set_input_func()
X Window System Xt (Intrinsics)	XtAddInput() or XtAddAppInput()
Other toolkits including Xlib structured around `select(2)` or `poll(2)` system calls	The file descriptor returned by `tt_fd()` Note: Once the file descriptor is active and the select call exits, use tt_message_receive() to obtain a handle for the incoming message.

7.4 Sending and Receiving Messages in the Same Process

Normally, the receiver deletes the message when it has completed the requested operation. However, the ToolTalk service uses the same message ID for both the receiver and the requestor. When sending and receiving messages in the same process, these features cause the message underneath the requestor to be deleted as well.

One workaround is to put a refcount on the message. To do this, use the `tt_message_user[_set]()` function.

Another workaround is to destroy the message in the receiver only if the sender is not the current procid; for example:

```
Tt_callback_action
my_pattern_callback(Tt_message m, Tt_pattern p)
{
    /* normal message processing goes here */

    if (0!=strcmp(tt_message_sender(m),tt_default_procid()) {
    tt_message_destroy(m);
    }
    return TT_CALLBACK_PROCESSED;
}
```

7.5 Sending and Receiving Messages in a Networked Environment

You can use the ToolTalk service in a networked envirnoment; for example, you can start a tool on a different machine or join a session that is running on a different machine. To do so, invoke a ttsession with either the `-c` or `-p` option.

- The `-c` option will invoke the named program and place the right session id in its `_SUN_TT_SESSION` environment variable. For example, the command

```
ttsession -c cmdtool
```

  defines `_SUN_TT_SESSION` in that cmdtool and any ToolTalk client you run with the environment variable `$_SUN_TT_SESSION` set to its value will join the session owned by this ttsession.

- The `-p` option prints the session id to standard output. ttsession then forks into the background to run that session.

To join the session, an application must either pass the session id to `tt_default_session_set` or place the session id in the environment variable `_SUN_TT_SESSION` before it calls the `tt_open` function. `tt_open` will check the environment variable `_SUN_TT_SESSION` and join the indicated session (if it has a value).

7.6 Unregistering from the ToolTalk Service

When you want to stop interacting with the ToolTalk service and other ToolTalk session participants, you must *unregister* your process before your application exits.

```
        /*
         * Before leaving, allow ToolTalk to clean up.
         */
        tt_close();

        exit(0);
}
```

`tt_close` returns `Tt_status` and closes the current default procid.

7