Handling Errors 14

The ToolTalk service returns error status in the function's return value rather than in a global variable. ToolTalk functions return one of these error values:

- Tt_status
- int
- `char*` or opaque handle

Each return type is handled differently to determine if an error occurred. For example, the return value for `tt_default_session_set` is a `Tt_status` code. If the ToolTalk service sets the default session to the specified sessid:

- Without a problem — the `Tt_status` code returned is `TT_OK`.
- With a problem — the `Tt_status` code returned is `TT_ERR_SESSION`. This status code informs you that the sessid you passed was not valid.

14.1 Retrieving ToolTalk Error Status

You can use the ToolTalk error handling functions shown in Table 14-1 to retrieve error values.

Table 14-1 Retrieving ToolTalk Error Status

ToolTalk Function	Description
tt_pointer_error(char * return_val)	Returns an error encoded in a pointer.
tt_pointer_error((void *) (p))	Returns an error encoded in a pointer cast to `VOID *`.
tt_int_error(int return_val)	Returns an error encoded in an integer.

The return type for these functions is Tt_status.

14.2 Checking ToolTalk Error Status

You can use the ToolTalk error macro shown in Table 14-2 to check error values.

Table 14-2 ToolTalk Error Macros

Return Type	ToolTalk Macro	Expands to
Tt_status	tt_is_err(status_code)	(TT_WRN_LAST < (status_code))

14.3 Returned Value Status

Functions with Natural Return Values

If a ToolTalk function has a natural return value such as a pointer or an integer, a special *error value* is returned instead of the real value.

Functions with No Natural Return Values

If a ToolTalk function does not have a natural return value, the return value is an element of `Tt_status` *enum*.

To see if there is an error, use the ToolTalk macro `tt_is_err`, which returns an integer.

- If the return value is 0, the `Tt_status` *enum* is either `TT_OK` or a warning.
- If the return value is 1, the `Tt_status` *enum* is an error.

If there is an error, you can use the `tt_status_message` function to obtain the character string that explains the `Tt_status` code, as shown Figure 14-1.

```
char *spec_id, my_application_name;
Tt_status tterr;

tterr = tt_spec_write(spec_id);
if (tt_is_err(tterr)) {
     fprintf(stderr, "%s: %s\n", my_application_name,
         tt_status_message(tterr));
}
```

Figure 14-1 Obtaining an Error Explanation

14.4 Returned Pointer Status

If an error occurs during a ToolTalk function that returns a pointer, the ToolTalk service provides an address within the ToolTalk API library that indicates the appropriate `Tt_status` code. To check whether the pointer is valid, you can use the ToolTalk macro `tt_ptr_error`. If the pointer is an error value, you can use `tt_status_message` to get the `Tt_status` character string.

Figure 14-2 checks the pointer and retrieves and prints the `Tt_status` character string if an error value is found.

```
char *old_spec_id, new_file, new_spec_id, my_application_name;
Tt_status tterr;

new_spec_id = tt_spec_move(old_spec_id, new_file);
tterr = tt_ptr_error(new_spec_id);
switch (tterr) {
    case TT_OK:
     /*
      * Replace old_spec_id with new_spec_id in my internal
      * data structures.
      */
     update_my_spec_ids(old_spec_id, new_spec_id);
     break;
    case TT_WRN_SAME_OBJID:
     /*
      * The spec must have stayed in the same filesystem,
      * since ToolTalk is reusing the spec id. Do nothing.
      */
     break;
    case TT_ERR_FILE:
    case TT_ERR_ACCESS:
    default:
     fprintf(stderr, "%s: %s\n", my_application_name,
         tt_status_message(tterr));
     break;
}
```

Figure 14-2 Retrieving a Returned Pointer Status

14.5 Returned Integer Status

If an error occurs during a ToolTalk function that returns an integer, the return value is out-of-bounds. The `tt_int_error` function returns a status of `TT_OK` if the value is not out-of-bounds.

To check if a value is out-of-bounds, you can use the `tt_is_err` macro to determine if an error or a warning occurred.

To retrieve the character string for a `Tt_status` code, you can use `tt_status_message`.

Figure 14-3 checks a returned integer.

```
Tt_message msg;
int num_args;
Tt_status tterr;
char *my_application_name;

num_args = tt_message_args_count(msg);
tterr = tt_int_error(num_args);
if (tt_is_err(tterr)) {
     fprintf(stderr, "%s: %s\n", my_application_name,
        tt_status_message(tterr));
}
```

Figure 14-3 Checking a Returned Integer

14.6 Broken Connections

The ToolTalk service provides a function to notify processes if your tool exits unexpectedly. When you include the `tt_message_send_on_exit` call, the ToolTalk service queues the message internally until one of two events happen:

1. Your process calls `tt_close`.
   In this case, the ToolTalk service deletes the message from its queue.

2. The connection between the ttsession server and your process is broken; for example, the application crashed.
   In this case, the ToolTalk service matches the queued message to a pattern and delivers it in the same manner as if your had sent the message normally before exiting.

Your process can also send a normal message on a normal termination by calling `tt_message_send` before it calls `tt_close`. In this case, if your process sends its normal termination message but crashes before it calls `tt_close`, the ToolTalk service will deliver both the normal termination message and the `tt_message_send_on_exit` message to interested processes.

14.7 Error Propagation

ToolTalk functions that accept pointers always check the pointer passed in and return `TT_ERR_POINTER` if the pointer is an error value. This check allows you to combine calls in reasonable ways without checking the value of the pointer for every single call.

In the following code sample, a message is created, filled in, and sent. If `tt_message_create` fails, an error object is assigned to *m*, and all the `tt_message_`*xxx*`_set` and `tt_message_send` calls fail. To detect the error without checking between each call, you only need to check the return code from `tt_message_send`.

```
Tt_message m;

m=tt_message_create();
tt_message_op_set(m,"OP");
tt_message_address_set(m,TT_PROCEDURE);
tt_message_scope_set(m,TT_SESSION);
tt_message_class_set(m,TT_NOTICE);
tt_rc=tt_message_send(m);
if (tt_rc!=TT_OK)...
```