
Subject: Error Handling, Was: Passing optional parameters
Posted by [davidf](#) on Thu, 10 Feb 2000 08:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Ed Meinel (edward.s.meinel@aero.org) writes:

> Ugh, I **hate** MESSAGE. Why cause a crash when it is easy to exit nicely?

While we are on the subject of error handling, let me give you my 10 second lecture on how to keep a widget (or object) program running forever. :-)

First of all, the problem I have with Message and Dialog_Message in widget programs is that they hardly ever tell you exactly where the error occurred. In other words, if you do a traceback on the error they point to themselves rather than to the line of code that actually **caused** the error. This is especially the case when I am CATCHing the error. (Which is more or less the thing to do in almost any program, let alone widget programs.)

So, I am really interested in the **next-to-last** error, not the current one.

Then, I like to write programs that work in a device-independent way. I don't like to write programs that can work on the display, but can't work in the PostScript or Z-buffer device. Heck, I might want to display graphics **everywhere**, so why penalize myself? Up until IDL 5.3, when IDL has suddenly gotten a lot smarter (although slower, if you believe recent reports), you couldn't display a widget dialog like Dialog_Message if you were in the PostScript or Z-buffer device. So you had to know where you were and then either call Message or Dialog_Message depending upon what was going to work.

Anyway, whenever I find myself futzing around writing 10 lines of code to do what should be done in one line of code, I think about writing an IDL program. :-)

So I wrote the program ERROR_MESSAGE, which you can find on my web page:

http://www.dfanning.com/programs/error_message.pro

The purpose of the program is to handle all these decisions for me so I don't have to think anymore. :-)

Called in its very simplest form:

```
ok = Error_Message()
```

it uses either Message or Dialog_Message to report the error message found in !Error_State.Msg. Or, you can supply your own message:

```
ok = Error_Message("You idiot. Contact Coyote!")
```

If you want a traceback report showing the exact line of code that caused the error, printed in the Command Log window, you can set the Traceback keyword:

```
ok = Error_Message(/Traceback)
```

In practice I don't usually want a traceback (it scares hell out of users) unless I am debugging the code, so I usually set the Traceback keyword from a "debug" variable.

```
ok = Error_Message(Traceback=debug)
```

You can also (as of today) set the usual Message and Dialog_Message keywords Continue and Informational if you don't really want to stop everything by actually causing or (as we computer geeks call it) throwing an error.

Alright, alright. I'm getting around to the 10 second lecture!

So, to keep a widget program running forever, you are going to CATCH any error that occurs, alert the user, and back out of the event handler gracefully. For me, this usually means sticking that info structure back in the user value of the top-level base so it can be found the next time I need it.

But--and here is the whole point of this long-winded tirade--you really ought to be sure you **cancel** the CATCH when you get into your error handling code. If you don't, then errors in your error handling code will cause you to be in one VERY big loop! (A student in a recent class had his error in the CATCH cancel code and even fervent prayer wasn't able to help. :-)

So here is what the top of almost any event handler (or object method) code that I write looks like:

```
PRO JUNK_EVENT, event
```

```
On_Error, 2 ; Return to caller if there is no CATCH.
```

```
Widget_Control, event.top, Get_UValue=info, /No_Copy
```

```
Catch, theError
```

```
IF theError NE 0 THEN BEGIN
```

```
Catch, /Cancel
```

```
    ok = Error_Message(Traceback=info.debug)
```

```
    Widget_Control, event.top, Set_UValue=info, /No_Copy
```

```
    RETURN
```

```
ENDIF
```

Of course, if your event handler (or object method) is a function then the RETURN statement has to return some value. I usually use -1 or 0.

Doing this, or something very much like it, in most programs will really keep things running smoothly for a very long time.

Cheers,

David

--

David Fanning, Ph.D.

Fanning Software Consulting

Phone: 970-221-0438 E-Mail: davidf@dfanning.com

Coyote's Guide to IDL Programming: <http://www.dfanning.com/>

Toll-Free IDL Book Orders: 1-888-461-0155
