
Subject: IDLVM and retail

Posted by [JD Smith](#) on Mon, 04 Sep 2006 21:07:44 GMT

[View Forum Message](#) <> [Reply to Message](#)

I have a large widget program which runs (optionally) under the IDLVM. Around IDL version 6.0, when the VM was introduced, there was an inconsistency in how non-blocking widgets were dealt with in the VM vs. a normal IDL session, which was fixed in IDL 6.1:

Beginning with IDL 6.1, the XMANAGER procedure honors the value of the NO_BLOCK keyword in Runtime and Virtual Machine modes.

Because XMANAGER did not honor the NO_BLOCK keyword in previous releases, widget applications that worked properly (that is, not blocking) when run in a licensed full version of IDL behaved differently when run in Runtime or Virtual Machine mode. This difference in behavior has been removed; widget applications should behave identically (with regard to blocking behavior) in all IDL licensing modes.

Other differences between IDL's Virtual Machine mode and full licensed mode, such as the fact that programs that call the IDL EXECUTE function will not run in the IDL Virtual Machine, are still in effect. If you have modified your widget application to work around the old blocking behavior by removing the NO_BLOCK keyword from calls to XMANAGER or by substituting the JUST_REG keyword, you may need to reinsert the NO_BLOCK keyword to achieve the desired behavior.

My program has an error catching system that differentiates between two modes: running with a GUI, or running "lights-out" as a script. In the former case, the error is caught, displayed for the user with DIALOG_MESSAGE, and the RETALL command is then issued to return all the way back to the active command line, letting the program continue to run. When running as a script, instead MESSAGE is called, and the error halts the program.

What I have just discovered, to my dismay, is that RETALL completely closes down an IDLVM session, rather than returning to its "non-blocking" widget event processing shell, as the above 6.1 update note would imply.

Here's an example

```
;;; file test_vm.pro
pro test_vm_event, ev
  a=dialog_message('Testing RETALL',/ERROR)
  retail
```

```
    return
end

pro test_vm
    b=widget_base(/ROW)
    but=widget_button(b,VALUE='Do It!')
    widget_control, b,/realize
    XManager,'test_vm',b,/NO_BLOCK
    return
end
```

Run TEST_VM in a fresh IDL session, and you'll see a button. Click it, get an error message 'Testing RETALL', but the tiny app will continue to run (the desired behavior).

Now compile it to a .sav file:

```
IDL> save,FILE='test_vm.sav',/ROUTINES
```

quit IDL, and then:

```
% idl -vm=test_vm.sav
```

You'll see the same button. Click it, get the 'Testing RETALL' message, but notice that now when the message is dismissed and RETALL is called, the entire IDLVM session is shut down! I'm not sure why this is the case, but I need a way to work around it. Since there are many simultaneous event components all injecting events into the program suite, there is no convenient place to put a top level CATCH. I was relying on the background event processing inside of IDL to continue to run, i.e. to serve as an absolute top-level catch, but it doesn't do so in the VM.

Suggestions appreciated.

JD
