

---

Subject: GROUP\_LEADER (before I start my holidays ...)  
Posted by [Martin Schultz](#) on Mon, 22 Dec 1997 08:00:00 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Dear all,

first of all, I want to thank all of you who helped me out several times during the last 1/2 year since I joined this newsgroup. I guess, a special thanks must go to David F. (unfortunately I never succeeded to reach you via e-mail, David: there is still a 2% issue to be solved ?).

This time, I am back to widgeting a little, trying to improve my EXPLORE data analysis tool. In the online help, the WIDGET\_BASE function allows a parameter named GROUP\_LEADER which will destroy the new widget whenever the GROUP\_LEADER widget is destroyed. Sounds good, I thought, and tried it. However, when I kill my GROUP\_LEADER widget, only the contents of that new child is destroyed, and an empty window remains on the screen. I checked the widget ID of my GROUP\_LEADER to make sure it is the correct one.

Below is an example, simply call w\_test and close the MAIN WINDOW with either button ... BTW: I am using IDL 5.02, Unix version on AIX 4

Happy holidays to all of you,  
Martin.

--

-----  
Dr. Martin Schultz  
Department for Earth&Planetary Sciences, Harvard University  
186 Pierce Hall, 29 Oxford St., Cambridge, MA-02138, USA

phone: (617)-496-8318  
fax : (617)-495-4551

e-mail: [mgs@io.harvard.edu](mailto:mgs@io.harvard.edu)  
IDL-homepage: <http://www-as.harvard.edu/people/staff/mgs/idl/>  
-----

pro w\_test

```
dlg = w_edit(title='MAIN WIDGET',text=['Line 1','Line 2'])
```

```
widget_control,dlg,/realize  
print,'widget_ID(DLG) = ',dlg
```

```
dlg2 = w_edit(title='DEPENDENT WIDGET', $
             group_leader=dlg, $
             text=['2nd window:', 'Line 1', 'Line 2'])
```

```
widget_control,dlg2,/realize
print,'widget_ID(DLG2) = ',dlg2
```

```
event = widget_event(dlg)
```

```
widget_control,event.top,/destroy
```

```
end
```

```
;-----
;+
; NAME:
;   W_EDIT
;
; PURPOSE:
;   creates a simple text editor with an OK and Cancel button
;   and handles the events of this widget.
;
; CATEGORY:
;   general purpose widgets - modal widgets
;
; CALLING SEQUENCE:
;   dlg = W_EDIT(parent, [keywords])
;
; INPUTS:
;   PARENT --> widget ID of the parent widget
;
; KEYWORD PARAMETERS:
;   TITLE --> window title for the editor window (default blank)
;
;   TEXT --> initial text in the editor window (string array)
;
;   GROUP_LEADER --> if this widget shall be used as a simple display
;   window, you can specify a GROUP_LEADER (= widget-ID of a
;   dialog box). The window will then disappear as soon as the
;   dialog box is closed. NOTE: the OK and Cancel buttons will
;   not be shown in this case.
;
; OUTPUTS:
;   w_edit returns a widget ID of the editor. For implementation
;   see example below.
```

```

;
; SUBROUTINES:
;   EDIT_EVENT --> handles editor events. Reacts only to OK or Cancel.
;-

```

```

FUNCTION edit_event, event

```

```

    parent=event.handler

```

```

; Retrieve the structure from the child that contains the sub ids.
stash = WIDGET_INFO(parent, /CHILD)
WIDGET_CONTROL, stash, GET_UVALUE=state, /NO_COPY

```

```

    passevent = 0

```

```

; -----
; button pressed ("OK" or "Cancel")
; -----
if(event.id eq state.bID) then begin
    widget_control,state.textID,get_value=text

    info = text
    value = 1-event.value ; OK=1, Cancel=0
    passevent = 1 ; this terminates the dialog
endif

```

```

; Restore the state structure
WIDGET_CONTROL, stash, SET_UVALUE=state, /NO_COPY

```

```

if (passevent) then $
    return, { ID:parent, TOP:event.top, HANDLER:0L, VALUE:value , $
            INFO:info } $
else $
    return,0
END

```

```

;-----

```

```

FUNCTION w_edit, TITLE=title, TEXT=text, UVALUE=uval, $
    GROUP_LEADER=group_leader

```

```

ON_ERROR,2 ; return to caller

```

```

; Defaults for keywords
IF NOT (KEYWORD_SET(uval)) THEN uval = 0
if (not keyword_set(title)) then title = ''
if (not keyword_set(text)) then text = ''
if (not keyword_set(group_leader)) then group_leader = 0

print,'GROUP_LEADER=',group_leader

base = WIDGET_BASE(TITLE=title, UVALUE = uval, $
    frame = 3, /column, $
EVENT_FUNC = "edit_event" )

ysize = 8 > n_elements(text) < 40

textf = widget_text(base,/editable,xsize=80,ysize=ysize,frame=3, $
    /scroll,value=text,group_leader=group_leader)

if (group_leader eq 0) then $
    buttons = cw_bgroup(base,/row,['OK','Cancel']) $
else $
    buttons = -1

state = { bID:buttons, textID:textf }
; Save out the initial state structure into the first child's UVALUE.
WIDGET_CONTROL, WIDGET_INFO(base, /CHILD), SET_UVALUE=state, /NO_COPY

RETURN, base

END

```

## File Attachments

---

1) [w\\_test.pro](#), downloaded 120 times

---