```
Function DialogDoneButtonProc(ba) : ButtonControl
   STRUCT WMButtonAction &ba
   switch( ba.eventCode )
      case 2:            // mouse up
         // turn off the named window hook
         SetWindow $ba.win hook(dlog)=$""
         // kill the window AFTER this routine returns
         Execute/P/Q/Z "KillWindow " + ba.win
         break
   endswitch
   return 0
End
```

Window Hook Deactivate and Kill Events

The actions caused by these events (eventCode 2, 14, 15, 16 and 17) potentially affect multiple subwindows.

If you kill a subwindow, the root window's hook functions receives a subwindowKill event for that subwindow and any child subwindows.

If you kill a root window, the root window's hook function(s) receives a subwindowKill event for each child subwindow, and then the root window's hook function(s) receive a kill event.

Likewise, hiding and showing windows can result in subwindows being hidden or shown. In each case, the window hook function receives a hide or show event for each affected window or subwindow.

The winName member of WMWinHookStruct will be set to the full subwindow path of the subwindow that is affected.

Events for an exterior subwindow are a special case. See **Hook Functions for Exterior Subwindows** on page IV-305.

The hook functions attached to an exterior subwindow will receive a subwindowKill event if the exterior subwindow is killed as a result of killing the parent window. But it will receive a regular kill event if it is killed directly. Normal subwindows always receive only subwindowKill events.

The kill-related events are sent in this order when a window or subwindow is killed:

1. A killVote event is sent to the root window's hook function(s). If any hook function returns 2, no further events are generated and the window is not killed.
2. If the window is not a subwindow and wasn't created with /K=1, /K=2 or /K=3, the standard window close dialog appears. If the close is cancelled, the window is not killed, the window will receive an activate event when the dialog is dismissed, and no further events are generated. Otherwise, proceed to step 3.
3. If the window being killed has subwindows, starting from the bottom-most subwindow and working back toward the window being killed:

3a. If the subwindow is a panel, action procedures for controls contained in the subwindow are called with event -1, "control being killed".

3b. The root window's hook function(s) receive a subwindowKill event for the subwindow. If any hook function returns 1, no further subwindow hook events or control being killed events are sent, but the window killing process continues.

Steps 3a and 3b are repeated for each subwindow until the window or subwindow being killed is reached.

4. If the killed window is a root window, a kill event is sent to the root window's hook function(s). If any hook function returns 2, no further events are generated and the window is not killed. This method of preventing a window from closing is to be avoided: use the killVote event or the window-equivalent of `NewPanel/K=2`.

   Prior to Igor 7, you could return 2 when the window hook received a kill event to prevent the killing of the window. This is no longer supported. Use the killVote event instead.