Check out the New Clarion Magazine!

Check out our new Clarion content at www.ClarionMag.com!

This site will remain online as an article archive.

No! You Can't Cancel

By Steven Parker

Posted December 13 2006

Clarion developers are a nasty bunch. They do not want to let their end users cancel forms. At least it seems so from the frequency with which questions are asked about trapping the Escape key or Windows' red X.

In many cases, the desire to stop a user from exiting a data entry form without saving is due to ill-considered design. For example, I have a form on Purchase Order header records that has a tab on it, and on that tab a user may add, change or delete records in the Purchase Order detail file (I plead "the boss made me do it").

What's the big deal? Glad you asked. Suppose a user starts a new Purchase Order (creating a parent record) and adds items to the PO (creating child records). If there were a way for the user to escape from the form, the child records would have been saved but the parent record would not be. Can you say "hosed files?"

In this case, I should have done all my work in memory. If the user accepts the form (presses the Ok button), write the queue or In-Memory Database Driver records to file. If the user cancels, discard the queue or IMDD file. To handle inserts or deletes of existing parent-child (header-detail) combinations, there is quite a bit of detail to attend to. In any case, I could:

- dot my i's and cross my t's
- just use Mike Hanson's SuperInvoice

or,

- try to prevent the user canceling when any change has been made to the child file.

Management chose the last alternative. Truly, I am innocent!

There are three ways that I can think of to stop a user from canceling a form. Of course, I presuppose that there is a variable or condition that is set and tested to determine whether a cancel is allowable before implementing any of them. In the sample app, downloadable at the end of this article, I do not set up any testable condition; be warned.

Disable the X

Microsoft, once again, in its infinite wisdom provides a red X in the upper right corner of each window. Clicking on this closes the window and by-passes all the validation built into the templates. Clarion template-generated procedures do not trap this.

The solution to disabling the X comes from (who else?) Jim Kane. In the sample app, from the Frame, click Browse | Browse Employees file - no X. Then press Insert and you will see the form in Figure 1.

Figure 1. Red X disabled on Form

This is how Mr. Kane did it:

In the global data area, add the following equates:

```
SC_CLOSE                equate(0f060h)
MF_GRAYED           equate(1)
```

In the global map, prototype the following APIs:

```
Module('win32')
  EnableMenuItem(signed hMenu, signed uIDEnableItem, signed uEnable),|
      bool,pascal,PROC
  DrawMenuBar(signed hWnd),bool,pascal,PROC
  GetSystemMenu(signed hWnd, bool act),signed,pascal
End
```

In the form procedure, declare `hMenu` as a local long. Then after the window is opened:

```
hMenu = GetSystemMenu(ThisWindow{PROP:Handle}, False)
EnableMenuItem(hMenu, sc_close, MF_GRAYED)
DrawMenuBar(ThisWindow{PROP:Handle})
```

As Mr. Kane notes, this will disable the Windows close option but the Esc key and ALT-F4 will still close the window.

Trapping ESC

In the sample app, from the Frame, click Browse | Browse the Reviews file (Wizard) - no ESC Then press Insert. Press the escape key:

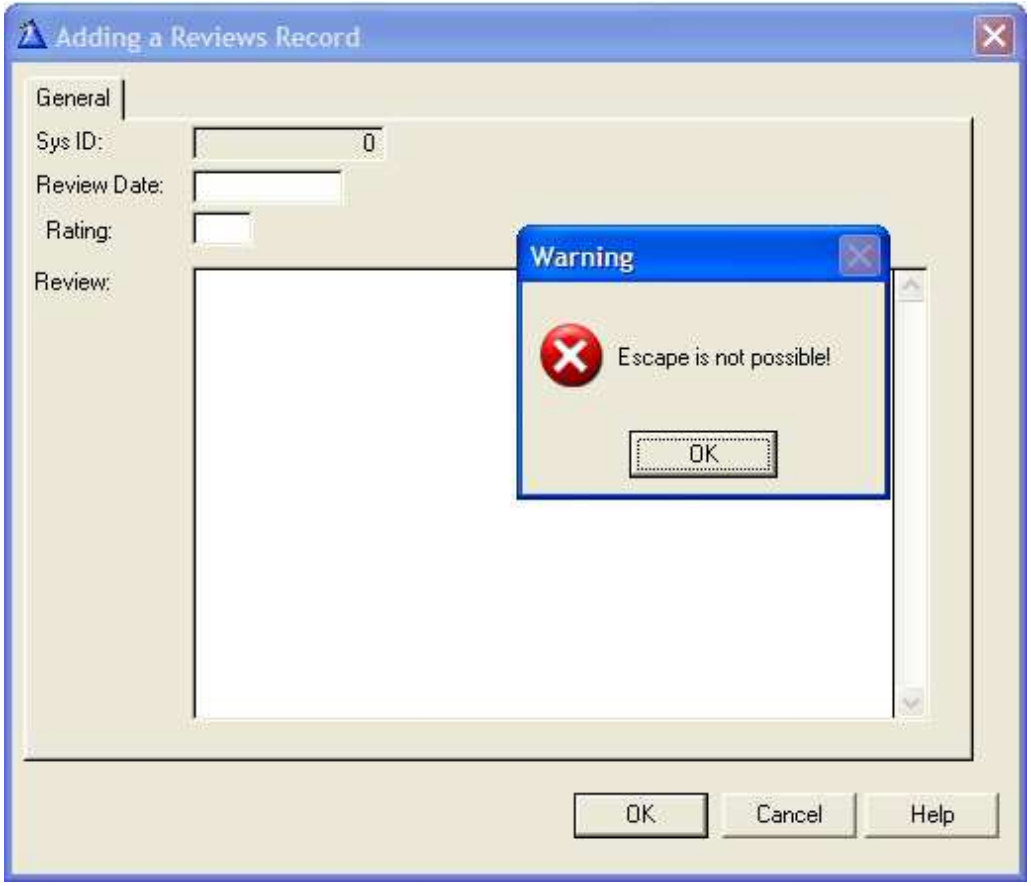

Figure 2. Trapping the ESC key

Here is how this is done.

On the window, right click and select Alert... and click Add. When the Input Key window comes up, press the ellipsis and type `EscKey` in the Enter Key prompt.

Then, in Window Event handling for the Alert key:

```
If KeyCode() = EscKey
   Message('Escape is not possible!','Warning',Icon:Hand)
   Select(1)
```

```
    Cycle
    Return Level:Notify
  End
```

will prevent the user from closing the window on the Esc key.

But the Cancel button still works.... So, not even combining trapping the X and trapping the Esc key will fully cover the need.

Besides, using both techniques begins to involve a lot of code.

TakeCloseEvent

Any time a window is closed, its `TakeCloseEvent` method is called. Therefore, this is an *ideal* place to test a "do not close" condition. For example:

```
If MyCondition = True
  BEEP(BEEP:SystemExclamation)  ;  YIELD()
  MESSAGE('Cancellation is futile.', |
       'Cancel Not Permitted', ICON:Exclamation)
  Return Level:Notify
End
```

In the sample app, Browse | Browse Reviews (Adjusted) - Late Trap. Before going into this update form, I suggest you make sure you have Windows' TaskManager ready. I initialized `MyCondition` to `True`, so you will never get out of the form without killing it from TaskManager.

Change the code (the app is C5.5) and change this embed so that it reads:

```
If MyCondition = True and Self.Response = RequestCancelled
```

and you may have better luck closing the form window. With this modification, only canceling is barred and this is what is wanted: the Esc key, the Cancel button and the red X will not close the form.

The success of this approach depends on correctly setting the condition(s) to check....

Summary

I knew I should have designed my purchase order parent-child update so that it was done using queues or, later, the In-Memory Database Driver. Inserts would have been easy; if the user pressed Ok, dump the memory buffers to file. Changes would have been a bit more difficult since I would have to re-write the header record but remove all child records before dumping the new ones to file.

Of course, there is also the issue of autonumbered keys. In my case, the Purchase Order header file has one.

But, this is what management wants. Yes, yes, I should have fought it.

Yet, given that I didn't, the templates still forgive me. I have several ways to stop users from hurting my files.

Download the source

Steve Parker started his professional life as a Philosopher but now tries to imitate a Clarion developer. He has been attempting to subdue Clarion since version 2007 (DOS, that is). He reports that, so far, Clarion is winning. Steve has been writing about Clarion since 1993.

Article comments

by Jim Breisch on December 20 2006 (comment link)

With this modification, only canceling is barred and this is what is wanted: the Esc key, the Cancel button and the red X will not close the form.

Either I'm suffering from a brain cramp or the above is contadictory. I think maybe he editor got carried away with his
red pen <g>

---

by Steven Parker on December 20 2006 (comment link)

Okay, perhaps it might have been better stated as something like: if the user is cancelling _and_ the condition you set also fails, the user cannot close the form; IOW, the user may cancel only when the developer decides to allow it.

BACK TO TOP