;CreatingAWindowWithMultipleDisplays.rtfd;;¬   Next Section ;LoadingNibFilesDynamically.rtfd;;¬   Previous Section

DisplayingAnAttentionPanel;¬Displaying an attention panel

arrow.eps ¬   Call NSRunAlertPanel().

Or

arrow.eps ¬   Create an attention panel in Interface Builder and load it dynamically.

When you can accomplish an end programmatically or in Interface Builder, the recommended course is almost always Interface Builder. A notable exception is displaying attention panels. You display attention panels to tell the user something about the current context (such as an error that occurred), to clarify or complete an action the user is taking, or to give the user a chance to take corrective steps.

Displaying Attention PanelsProgrammatically

For most situations requiring attention panels, the easiest and most appropriate thing to do is call a function: **NSRunAlertPanel()**. In the following example, the application informs users that, because of hardware incompatibility, it cannot proceed:

```
if(![LiveVideoView doesWindowSupportVideo:bufWindow
                   standard:&type size:&vidSize])
  {
    NSRunAlertPanel(@"No Video Present", @"This machine is not
     €€capable of running video applications. Since this program
     €€is exclusively for Video,it will now exit.", @"OK", nil, nil);
    [self terminate:self];
}
```

Notes on the code: The arguments of **NSRunAlertPanel()** determine what appears on the panel. The first argument is the heading (above the dividing line), and the second is the text (below the line). The next three arguments are the titles of the buttons that appear across the bottom of the panel. The first of these titles goes to the default button, which has a carriage return associated with it. You can remove a button by giving **nil** as its title, but you must specify something for all three arguments. The declaration of **NSRunAlertPanel()** permits a variable number of arguments, so you can have **printf()**-style format specifiers in the panel heading and text and variables following the third button argument.

The call to **NSRunAlertPanel()** in the example above creates the following panel:

DisplayAttn1.tiff ¬

The Application Kit defines other functions related to **NSRunAlertPanel()**. For more information on these functions, see the ªFunctionsº section of the *Application Kit Reference*.

Loading Attention PanelsCreated in Interface Builder

The panel created by **NSRunAlertPanel()** might not be adequate for certain situations. For example, you might want to display an attention panelthat has a special view object, say one that shows the progress of some lengthy process (such as a progress bar for loading or copying files). And you want to give the user the options of aborting or pausing that process. You'd want something like this:

_Displaying.eps ¬

To implement a custom attention panel, you perform almost identical steps as you do to create an Infopanel:

1. Pick a custom class, typically the application's controller, to manage the panel.
2. Specify an action and outlet in the controller class.
3. Connect the action in the main nib file.
4. Create a nib file for the attention panelby choosing Document **arrow.eps ¬** New Modules **arrow.eps ¬** New Attention Panel.
5. Compose the text, graphics, and other UI elements of the panel.
6. Drag the controller's header file to the attention panel's nib file window.
7. Assign the controller class to File's Owner.
8. Assign the attention panelto the File's Owner attention panel outlet.
9. In the action method, load the panel's nib file with **loadNibNamed:owner:**.

There are some important differences between attention panels and Info panels. With attention panels, you typically load the nib file not as the result of a user action (for instance, clicking an Info Panel command), but because of internal conditions in your code. Also, you dismiss an Info Panel by clicking its close box; you usually dismiss an attention panel by clicking a button on the panel. This means that, for custom attention panels, you will have to define and implement action methods for the buttons on the panels. (This is something **NSRunAlertPanel()** simulates by returning a code indicating the button clicked.)