Progress bar

Objective: *This tutorial shows you how to use a progress bar in a dialog.*

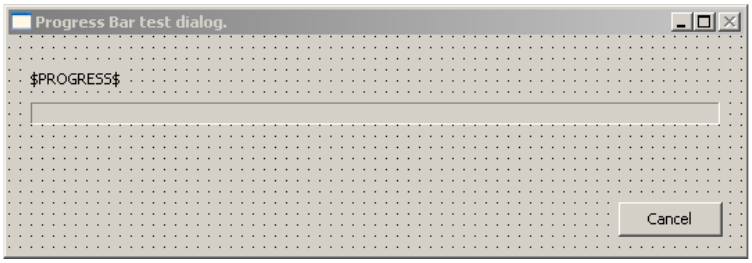
Step 1:

Create a new **Blank Setup** project in IA.

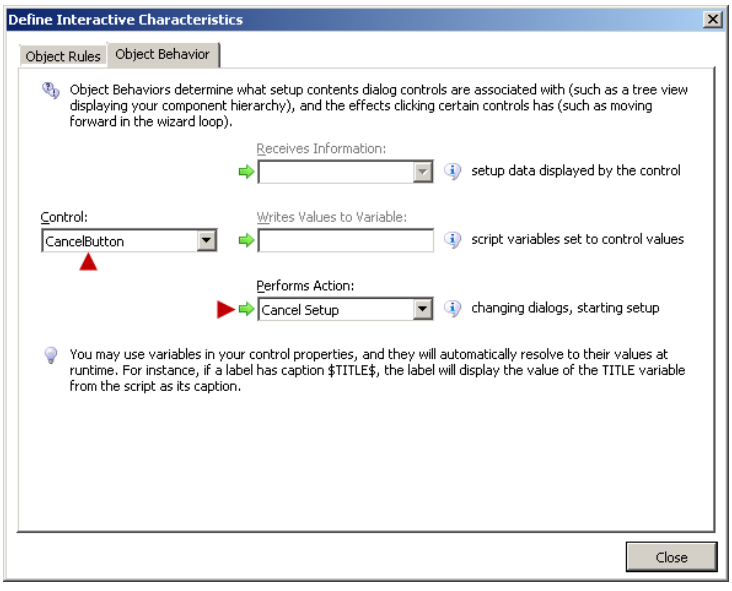
Step 2:

The dialog we are going to create uses a label, a progress bar, and a button. The label is used to display the text value of the progress bar value. The progress bar will display the current value of the pre-defined variable; I'll describe this later. Finally, the button will be used to cancel the dialog at runtime.

Select the **Design** tab and click the **Edit Forms** item on the **Tools** ribbon. Add the label, progress bar, and button so that the dialog looks something like this (we will set the properties in a moment):



Edit the properties of the button and name it **Cancel**. Double-click the button. This displays the **Define Interactive Characteristics** dialog. Select the **Object Behavior** and change the **Cancel** button as highlighted:



This document is compatible with Install**Aware** 2.x and above.

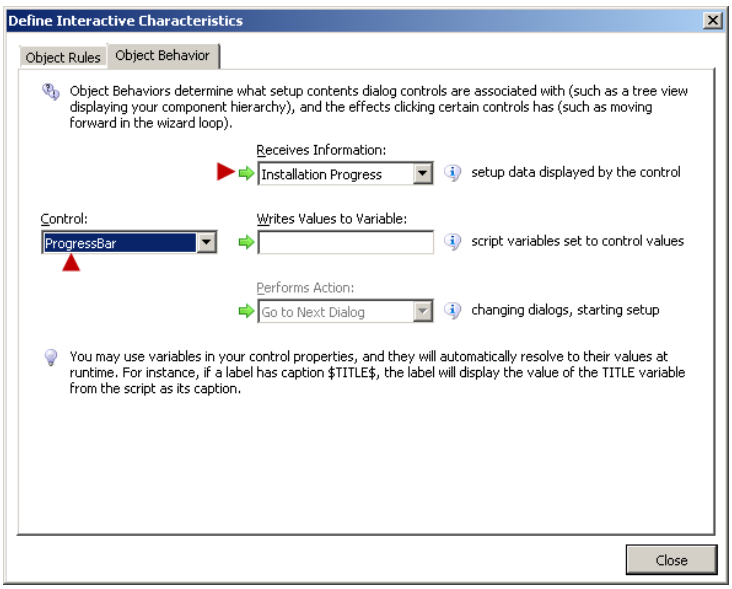
Written by Peter Hamilton-Scott, March 2009.

Progress bar

We have wired-up the button to the **Cancel Setup** action. If the button is pressed the **Cancel Setup** action will set the **ABORT** pre-defined variable. IA has a number of pre-defined variables which you can inspect with the help system (F1).

Double-click the label and change its caption to **$PROGRESS$.** Again, **PROGRESS** is a pre-defined variable associated with progress bars. The label will display the text value of the progress bar.

Select the progress bar in the **Control** dropdown. Don't waste time trying to double-click the progress bar. It does not work! You have to select another control first and *then* select the progress bar. Change the progress bar behaviour to look like this:



Notice the **Installation Progress** item in the **Receives Information** dropdown. The image displayed in a progress bar is taken from the **PROGRESS** pre-defined variable.

The **Installation Progress** is internally wired-up to the **PROGRESS** variable. You don't need to do anything else except change its value to something in the range of the progress bar (see the minimum and maximum property values) and the image of the progress bar will be updated automagically [sic].

Step 3:

Save the dialog using **File -> Save** and select a name for your dialog, say, **ProgressBarTest**.

You now need to register the dialog in IA.

Select the **MSIcode** tab and select the **Dialogs** node. Right-click and select the **Add Dialogs to Project...** menu item.

Select the **Choose Dialog to Add by File Name** radio button and using the **Browse…** button select the dialog you have just created.

Step 4:

Copy and paste the following code into the **MSIcode** window:

```
Set Variable PROGRESS to 0
Display Dialog: ProgressBarTest, use as progress dialog (non-modal)
label: Main loop
if Variable ABORT Equals TRUE
 Terminate Installation
end
PROGRESS = $PROGRESS$ + 1
if Variable PROGRESS not Equals 100
 Comment: Do something here...
 Run Program dir , startup in folder C:\Temp (WAIT)
 GoTo Label: Main loop
end
Call DLL Function Kernel32.dll->Sleep
Hide Dialog
```

We want to display the progress bar so we must ensure the value of the **PROGRESS** pre-defined value is set to 0. You can, of course, initialise it to any value you want. The dialog can now be displayed and notice it *must* be displayed as non-modal if you want the code below it to be executed.

The first thing the main loop will do is check the status of the **ABORT** pre-defined variable and this is used to terminate the installation. Remember, this pre-defined variable is set if the **Cancel** button is clicked as it performs the **Cancel Setup** action.

We then increment the pre-defined variable by any value you want to use, in our example by 1. ***Tip***: use the **Mathematics** built-in function to do this. We then check the pre-defined variable value against the upper value we want to run the loop for. It makes sense to compare the pre-defined value to the maximum value you've assigned to the progress bar. Once the loop has started to run you can then do anything that's appropriate for your loop. The dialog - in common with all Windows applications - is sensitive to events being raised at the right time so that the progress bar can be refreshed. You should make sure that if you run a periodic application in the loop that it must allow events to be serviced. Otherwise your installation may appear to freeze, giving your users the impression it is doing nothing. In the example I've used my application redirects a directory listing to the *nul* Windows device. The loop then resumes and keeps repeating until we want it to stop.

The last two lines are worthy of mention. The penultimate line demonstrates how to call a Windows kernel function. Although it's not readily apparent the code calls the **Sleep** function to suspend activity for two seconds. This allows the progress bar to refresh fully and gives your user a final confirmation that execution has completed. It's sometimes useful to do this and a short sleep interval allows that to be mentally registered but it's purely arbitrary if you want to do so.

The final line, **Hide Dialog** will close the dialog. You should do this when you've finished with your custom dialogs. Otherwise, the dialog will remain on the screen

until another dialog is displayed. Again, this is arbitrary but closing dialogs at the right time can enhance the professional perception of your installations packages.

Step 5:

Save your project and build it and run it. Depending on what your application does inside the loop, you might not see smooth scrolling of progress bar values in the label. You can't really control this all of the time so it's quite acceptable if you see the label jump from say, 9 to 30 in an instant. The label is intended only to show you how to display the progress bar value.

Summary:

The progress bar is linked to the **PROGRESS** pre-defined variable because we have wired it to the **Installation Progress** behaviour.

The **Cancel** button is associated with the **Cancel Setup** action and that in turn notifies your script using the **ABORT** pre-defined variable.

It may be useful to save the value of the progress bar pre-defined variable before you use it and then restore it once you've finished with it. This will ensure that if the pre-defined variable was already in use it can be reinstated for the other activity in your installation.

Do not expect smooth scrolling of either the progress bar or any related label value. These controls are subject to timely event notifications for best effect.

Suspending your installation with a timed pause can be helpful to give your users final confirmation that you have completed a lengthy process or set of processes.

Instead of waiting for another dialog to close your custom dialog you should explicitly hide it when you've finished with it.