Creating and registering new dialogs

Objective: *This tutorial shows you how to create a new sample dialog and register it for use in your projects.*

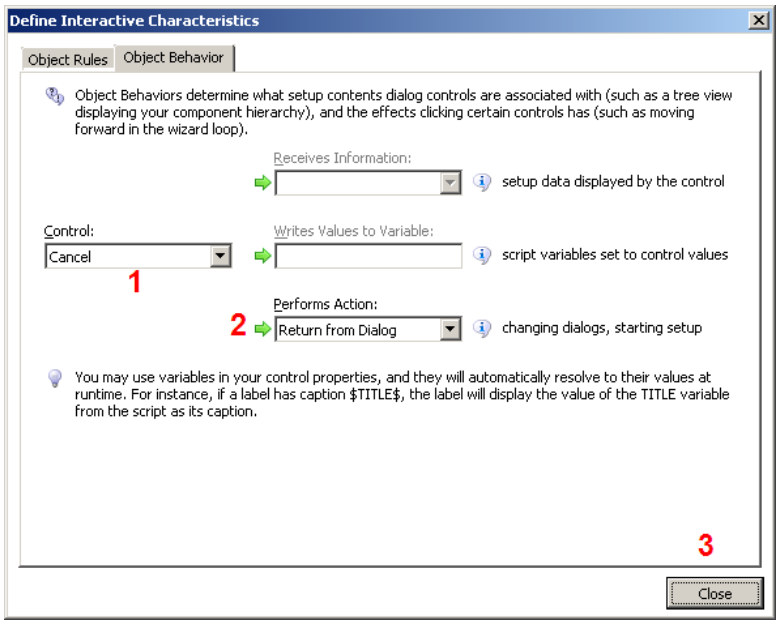
Step 1:

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

Step 2:

Select the **Design** tab and click the **Edit Forms** item on the **Tools** ribbon.

Add a button to the dialog and name it **Cancel**. Edit other properties if you want to. Double-click the button. This displays the **Define Interactive Characteristics** dialog. Select the **Object Behavior** tab do the three steps illustrated below:



There are many things you can do with controls, but that's another story. The important thing I am demonstrating here is how you create a skeleton dialog that does something minimal but useful all the same. IA dialogs do not expose code-behind. This means you cannot write code for the events raised by the controls. Rather, you interact with the control using a subset of defined behaviours and optional rules.

Behaviours and rules provide a robust granularity of interaction. What makes the interaction relatively easy to work with is that you can only work using what IA exposes. You can select things from dropdowns for example and rules are easily constructed using designers. You do have some editing flexibility but the benefit of this controlled editing is that you can create the environment for your dialog

Creating and registering new dialogs

with the minimum of fuss. Because there is no code directly associated with the controls, you don't have to worry about using a potentially difficult to learn scripting language.

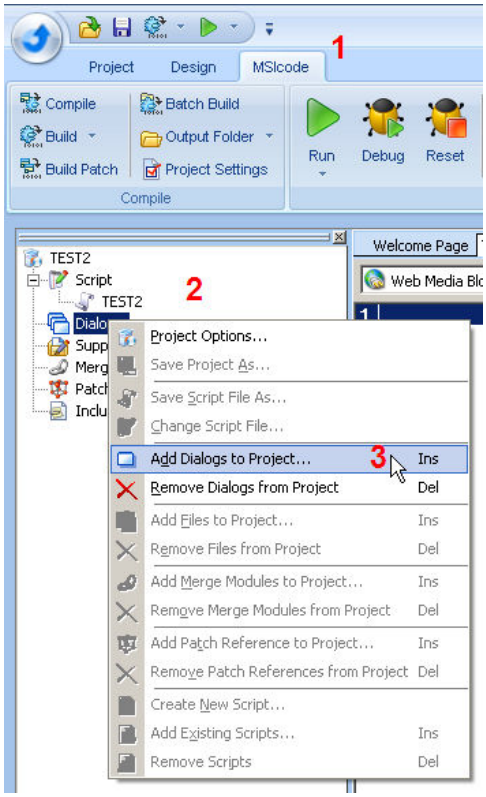
Of course, you can write code to respond to actions signalled by the dialog and this is performed in the MSIcode window. For now, this tutorial will do nothing more complex than close the dialog.

Step 3:

Save the dialog using the **File** menu **Save** item and select an appropriate name for your dialog, for example **TestDialog**. When you save the dialog, make sure you save it in the same folder where you created the project. We are not interested here in using the dialog across multiple projects. Saving the dialog in the project folder keeps the dialog local to the project. If you now look in the project folder for your project you will see two new files called **TestDialog.dfm** and **TestDialog.dfm.miaf**.

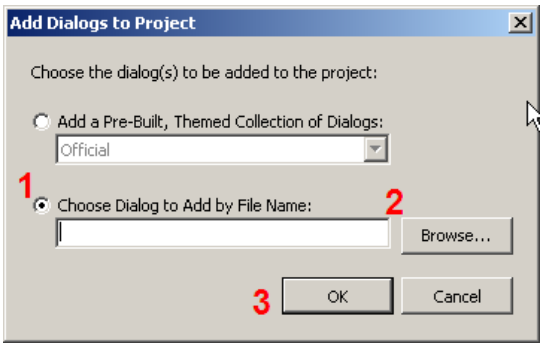
Step 4:

We have created the dialog but that is all. When you create a dialog it exists on the file system but the dialog does not yet exist in IA. We now need to register the dialog in IA. Select the **MSIcode** tab and select the **Dialogs** node. The following picture shows the steps you need to do:



Creating and registering new dialogs

When you add a dialog to the project you will see the following:

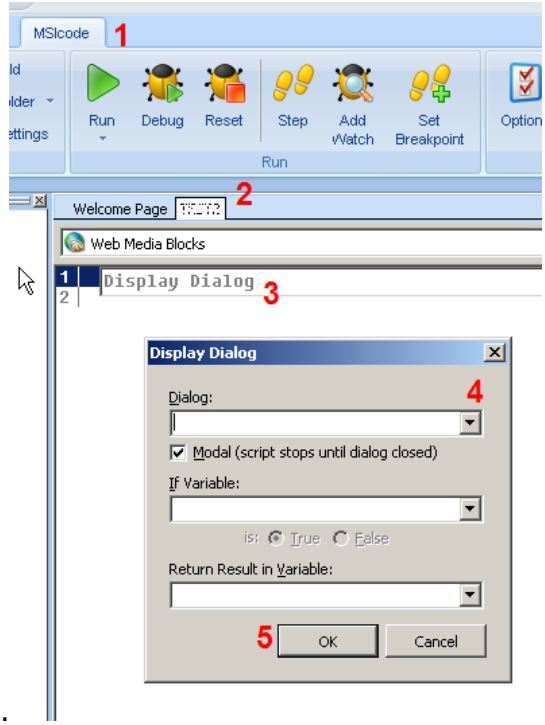


Because we are adding a dialog by filename the concept of themes does not apply in this tutorial. Themes are very useful as they provide a set of dialogs with similar look and feel. At this time however we are only using a simple dialog. Do the three steps as shown above.

Enter the pathname or use the browse button. When you have selected the dialog, click the **OK** button. If you now look under the **Dialogs** node you will now see your dialog registered in IA. We can now use it.

Step 5:

Let's now use the dialog in the MSIcode window. To use the dialog in your code position the cursor at line 1 and start typing **Display Dialog**. As you can see, IA completes the statement as you type. When you see **Display Dialog** on the line press the Enter key and that will cause the selection dialog to be displayed:



The first thing you will notice is that the MSIcode is mostly generated for you using code designers. This is a good thing. The code window is influenced by what

you do with the designers and the designers influence the code that is emitted. When you interact with the code you use a number of built-in functions and optional third-party plug-ins. You are restricted in what you can use but you are *not* limited by what you can do. The scripting environment is built-in in IA and is not an afterthought or separate product. You can certainly add coding errors to your project but the syntax will at least be clean and much easier to correct.

The dropdown containing the list of dialogs will contain one entry only and that is the name of your dialog. Select your dialog name. Notice the **Modal** checkbox. Ensure it is selected. Click **OK**. When you close the dialog you will notice that IA generates the statement for you. The syntax of the statement reflects the settings you used in the designer. If you position the cursor on line 1 and press the enter key, notice that IA will decode the statement and it will display the designer using the values originally in the statement. This is one of the most important and useful features of the IA code window: two-way association between the statement and the designer and it is something worth getting used to.

Step 6:

Save your project.

Build it.

Run it.

As the dialog was created modally, the script will do nothing more until you dismiss the dialog by clicking the **Cancel** button. Study the association between the behaviour described for the button in **Step 2** earlier.

Summary:

Our cancel button is configured to return from the dialog doing nothing more than closing the dialog and giving control to the next statement (if any) in the MSIcode. Our dialog does not do very much but it demonstrates the minimum steps you need to do to create a dialog and register it for use in IA.

Dialogs are usually created as modal as this is important for your users during the interview process and lets the user continue the installation when ready.

Dialogs should always have buttons to cancel an installation and other buttons to navigate forward and backward along the user interview chain.

You have seen how the designers and the code co-relate. I readily admit that this tutorial will not shake any trees but as an introduction to creating a dialog it will help you to get started.