Using Notifications (Events) in SolidWorks Macros

Notifications or Events give you a powerful method for triggering code to run when something happens in SolidWorks. This is the ideal method, for example, if you would like to run a procedure that checks to see if certain file properties have been filled out when a user saves a part. Unlike typical macros which run after the user clicks a button or manually runs the macro, macros that use notifications run continuously. They are prompted to run procedures when specific actions occur in SolidWorks.

This example will process notifications from SolidWorks to return when a part model is active and a notification from the part when it is saved.

Initial Code

1. Start a new macro by selecting **Tools, Macro, New** or clicking on the Macro toolbar. Save the macro to *"C:\Macros"* (create a new folder if necessary) as *notifications.swp*. This begins our macro with the following code.

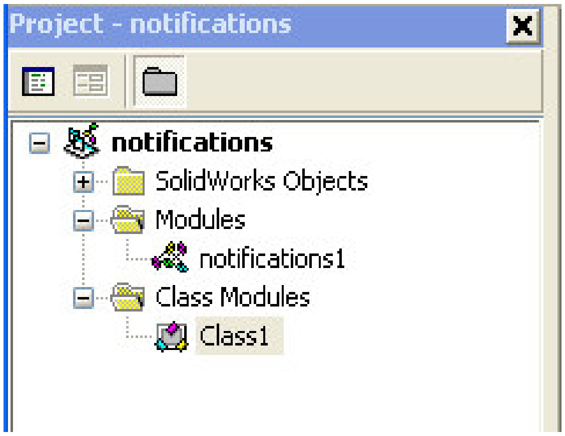
```
Dim swApp As Object
Sub main()

Set swApp = Application.SldWorks
End Sub
```

Classes

To enable notifications you must insert a class module into the macro. Notifications cannot be used in a standard code module.

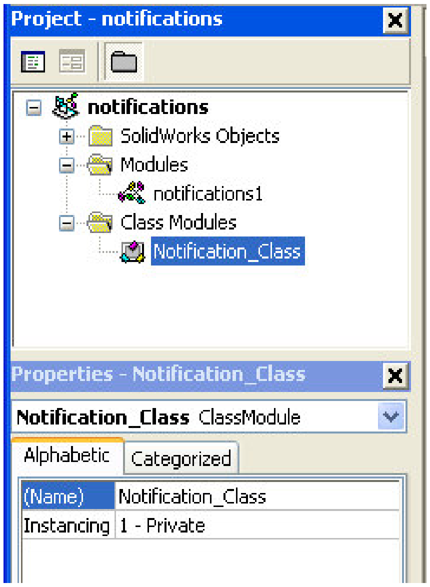
2. Add a new class module to your macro by selecting **Insert, Class Module** from the VBA interface (macro editor). You should see the following in the VBA Project Explorer with a new class module.



WithEvents Declaration

Declaring a variable using the **`WithEvents`** keyword tells the code that this is an object that will be triggered by SolidWorks. Several objects in SolidWorks support notifications. Each of which has its own list of notifications that are specific to the object type. This example uses notifications from the SolidWorks application and from parts.

3. Rename the class module *Notification_Class* and add the following declarations to enable notifications for the SolidWorks application and SolidWorks parts.



Notification_Class ClassModule

(Name)	Notification_Class
Instancing	1 - Private

```
Option Explicit
Dim WithEvents swApp As SldWorks.SldWorks
Dim WithEvents MyPart As SldWorks.PartDoc
```

4. Add the following procedure for attaching to the SolidWorks object through the class.

```
Public Sub MonitorSolidWorks()
    Set swApp = Application.SldWorks
End Sub
```

5. Switch back to the original code module named *notifications1*. Remove the `swApp` declaration and modify the code in the main procedure as follows.

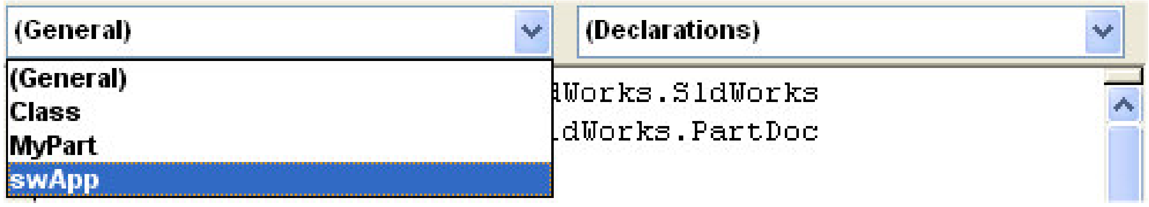
```
Sub main()
    Dim MyClass As New Notification_Class
    MyClass.MonitorSolidWorks
End Sub
```

You must create an object of the **`Class`** type (the name of the class you created) to use one of its methods. Using the **`New`** keyword creates an instance of the object the first time one of its procedures is called. This eliminates the need to use the **`Set`** command after declaring the object.

6. Switch back to the *Notification_Class* class module.

Creating the Notification Functions

Since the `swApp` and `MyPart` objects were declared using the `WithEvents` keyword, their notifications or events will be available through the object pull down list in the VBA interface.



```
lWorks.SldWorks
dWorks.PartDoc
```

7. Select **swApp** from the object list. From the procedures list, select **ActiveDocChangeNotify**. A new private fuction will be created that will run every time the active document in SolidWorks changes. This includes when a new file is opened or when switching between open documents.

8. Add the following code to the `swApp_ActiveDocChangeNotify` function.

```
Private Function swApp_ActiveDocChangeNotify() As Long
    If swApp.ActiveDoc.GetType = swDocPART Then
        Set MyPart = swApp.ActiveDoc
    Else
        Set MyPart = Nothing
    End If
End Function
```

The new code will check if the active document is a part. If it is, it will set the `MyPart` object to the active part document. Otherwise, it will set it to nothing, or no object.

9. Create a function that runs when a part is first saved by selecting MyPart from the object list and FileSaveAsNotify2 from the procedure list. Add the following code to the newly created MyPart_FileSaveAsNotify2 function.

```
Private Function MyPart_FileSaveAsNotify2(ByVal FileName _
  As String) As Long
    Dim result As Long
    result = MsgBox("Click OK to save " & FileName _
        & ".", vbOKCancel + vbQuestion)
    If result = vbCancel Then
        MyPart_FileSaveAsNotify2 = 1
    End If
End Function
```

FileSaveAsNotify2

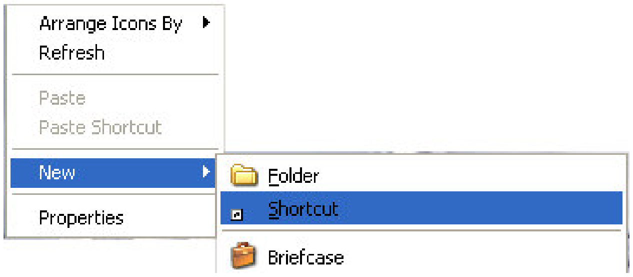
`MyPart_FileSaveAsNotify2` will run after the user has filled out the Save As dialog and clicked OK, but prior to the file actually being saved. This is a pre-notification. This allows you to catch the Save As action to make any necessary checks. It also passes the variable `FileName`. This corresponds to the file name the user typed into the Save As dialog. In the example, if the user clicks Cancel on the message box, the Function is set to a non-zero value, causing the Save As operation to be canceled.

10. Test your new macro by selecting **Run, Run Sub/UserForm** or by clicking ▶. Switch back to SolidWorks, create a new part and click Save. You should see your message box display.

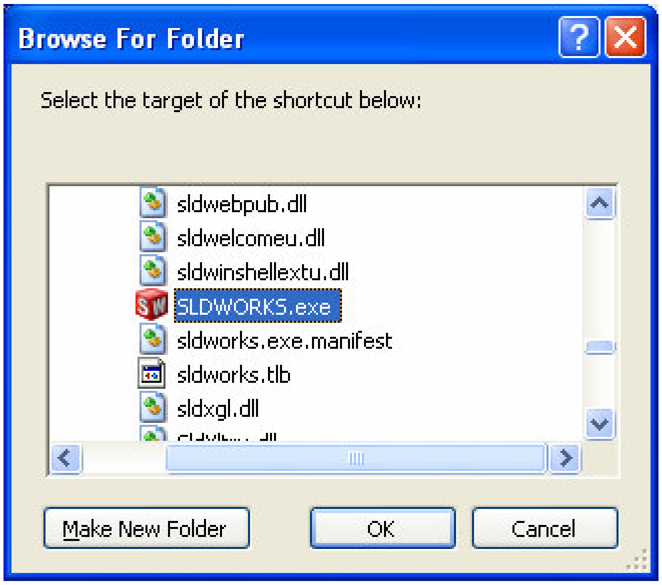
Starting Macros with SolidWorks

For notifications to work properly, they usually need to run during your entire SolidWorks session. In fact, if you manually start a macro that uses notifications, it will continue to run until you close your SolidWorks session. To start a macro with a SolidWorks session you must create a shortcut to the SolidWorks executable, adding the "/m" command line argument. Follow the steps below to launch a macro with your SolidWorks session.

11. Create a new shortcut on your Desktop by right-clicking and selecting **New, Shortcut**.



12. Browse to the SolidWorks executable, typically found in "C:\Program Files\SolidWorks\SLDWORKS.exe", and click OK.



Select the target of the shortcut below:

13. Click **Next** and name the shortcut “SolidWorks with Notifications”. Then click **Finish**.

14. Right-click on the newly created shortcut and select **Properties**. In the **Target** box, add the following text. This example assumes the standard SolidWorks executable location and that you have saved the macro to the directory mentioned in step 1.

```
"C:\Program Files\SolidWorks\SLDWORKS.exe" /m "C:\Macros\notifications.swp"
```

Conclusion

Start SolidWorks by using the new shortcut. Create a new part and save it. You should see your message box display.

Notifications are the building blocks to automatically running any procedures without requiring the user to remember to. They can be a great way to build in automated check or output routines. Here are some common uses and ideas for your own utilities: prompt the user to add properties or materials when they create or save models, automated part numbering, standards checking, notify a user that additional information must be updated, check for under or over defined sketches, etc.

Author: Mike Spens