

# How To Create & Use Custom Event Handlers in Inventor's iLogic Rules

You may or may not be familiar with Inventor's ability to automatically run your Inventor iLogic rules when certain events happen, but that is what I will be writing about in this document. There are two main ways to set this type of behavior up. You can use the Event Triggers dialog, or you can create your own custom event handler code, using Inventor API (Application Programming Interface) code. Both of those will be described in further detail below, but the primary focus of this document is the second option. This is an advanced topic, so a basic understanding of Inventor's iLogic rules is recommended before reading ahead.

## About The Event Triggers Dialog and How To Use It

This is a user interface tool, which you can access that the following location:

Manage tab > iLogic panel > left click on Event Triggers

When you click on that tool, it will open a dialog by the same name. Within this dialog, are four tabs along the top, a list of events on the right side, a list of iLogic rules in this document in the upper left area (may be empty), and a list of external iLogic rules in the lower left area (again, may be empty).

VB.NET (and iLogic) can use the "AddHandler" & "RemoveHandler" statements - OR - can use the "WithEvents" & "Handles" statements. But in VBA you can only use the "WithEvents" & "Handles" statements. The term '**handler**' means the Sub routine that will 'handle' the task of reacting to the event. The two ways mentioned above help you set up event handlers by specifying which event you want to handle and specifying what Sub you want to handle the event. (It cannot be a Function, because it cannot return a value.)

'Events' can be declared within Classes, Structures, Modules, and Interfaces. I will use a Class called LightSwitch and a variable for it called oLightSwitch as an example. It has an Event called 'OnToggle' defined within it. To set up a handler for this using the add & remove statements, you would create a line of code like the following, where the 'AddHandler' part specifies what event you want to handle, and the 'AddressOf' part specifies the name of the Sub routine you want to run when that event happens. It is common practice to name the handler like Variable.Event, but replace the dot (.) with an underscore (\_). Make sure the Sub called oLightSwitch\_OnToggle exists.

AddHandler oLightSwitch.OnToggle, AddressOf oLightSwitch\_OnToggle

To use 'WithEvents' statement and 'Handles' clause, you must declare a variable for the specific type of object that has the event, using the 'WithEvents' clause. This is best done outside of the Sub Main area, so that the variable is available to the handler Sub. Then you must put the 'Handles' statement at the end of the handler Sub routine's definition line, followed by specifying the event that it is to handle, using the variable you declared using the 'WithEvents' statement. Below are 2 example lines using above objects.

Public WithEvents oLightSwitch As LightSwitch

Public Sub oLightSwitch\_OnToggle(oSwitch As LightSwitch, On As Boolean) Handles oLightSwitch.OnToggle

Here is a List of objects in Inventor that have access to some events (not all):

ApplicationEvents, AssemblyEvents, BrowserPanelsEvents, CameraEvents, DockableWindowsEvents, DocumentEvents, DrawingEvents, DrawingViewEvents, FileAccessEvents, FileDialogEvents, FileManagerEvents, FileUIEvents, HelpEvents, InteractionEvents, KeyboardEvents, ManipulatorEvents, MeasureEvents, ModelingEvents, ModelStateEvents, MouseEvents, PartEvents, PluginLicenseManagerEvents, PresentationEvents, ReferenceKeyEvents, RepresentationEvents, SearchBoxEvents, SelectEvents, SketchEvents, StyleEvents, TransactionEvents, TriadEvents, UserInputEvents, UserInterfaceEvents, WebBrowserDialogEvents

Example below saved within an assembly, either ran manually or at document open, then listens for 'edit component', then shows iLogic form. Created for other post.

Sub Main

'When you enter 'Edit mode' of a component, will trigger Sub below and show local iLogic Form named "Form"

'When main assembly closed, will trigger Sub below to 'Remove' event handlers.

Dim oAppEvents As ApplicationEvents = ThisApplication.ApplicationEvents

AddHandler oAppEvents.OnNewEditObject, AddressOf ApplicationEvents\_OnNewEditObject

AddHandler oAppEvents.OnCloseDocument, AddressOf ApplicationEvents\_OnCloseDocument

End Sub

Public Sub ApplicationEvents\_OnNewEditObject(EditObject As Object, BeforeOrAfter As Inventor.EventTimingEnum, Context As Inventor.NameValueMap, ByRef HandlingCode As Inventor.HandlingCodeEnum)

If EditObject IsNot ThisAssembly.Document Then iLogicForm.Show("Form")

End Sub

Public Sub ApplicationEvents\_OnCloseDocument(DocumentObject As Inventor.\_Document, FullDocumentName As String, BeforeOrAfter As Inventor.EventTimingEnum, Context As Inventor.NameValueMap, ByRef HandlingCode As Inventor.HandlingCodeEnum)

```
If DocumentObject Is ThisAssembly.Document Then
    Dim oAppEvents As ApplicationEvents = ThisApplication.ApplicationEvents
    RemoveHandler oAppEvents.OnNewEditObject, AddressOf ApplicationEvents_OnNewEditObject
    MsgBox("Removed 'OnNewEditObject' Event handler.", , "")
    RemoveHandler oAppEvents.OnCloseDocument, AddressOf ApplicationEvents_OnCloseDocument
    MsgBox("Removed 'OnCloseDocument' Event handler.", , "")
End If
End Sub
```