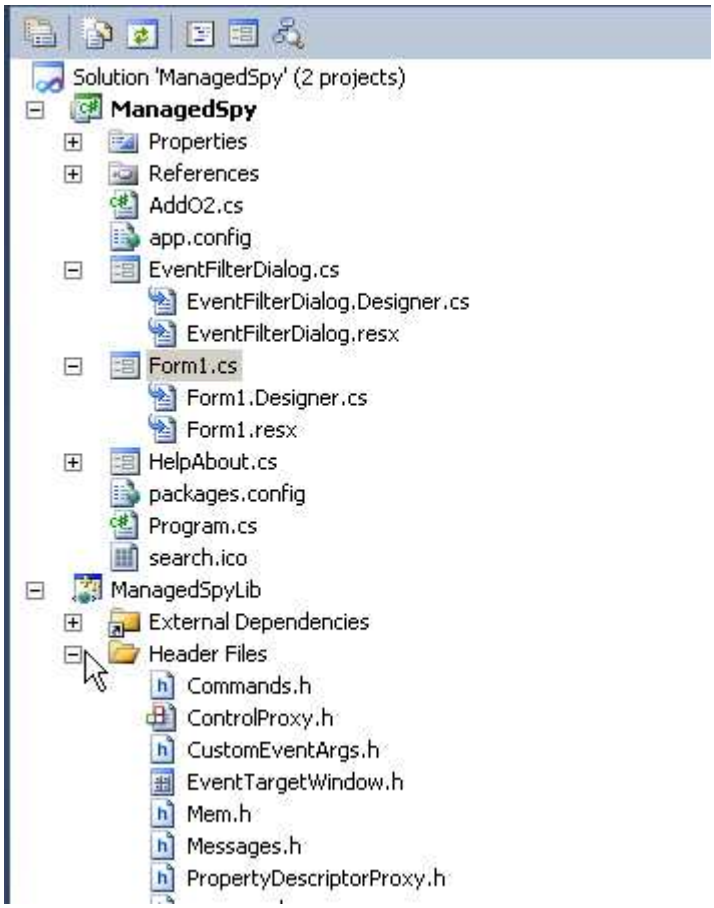
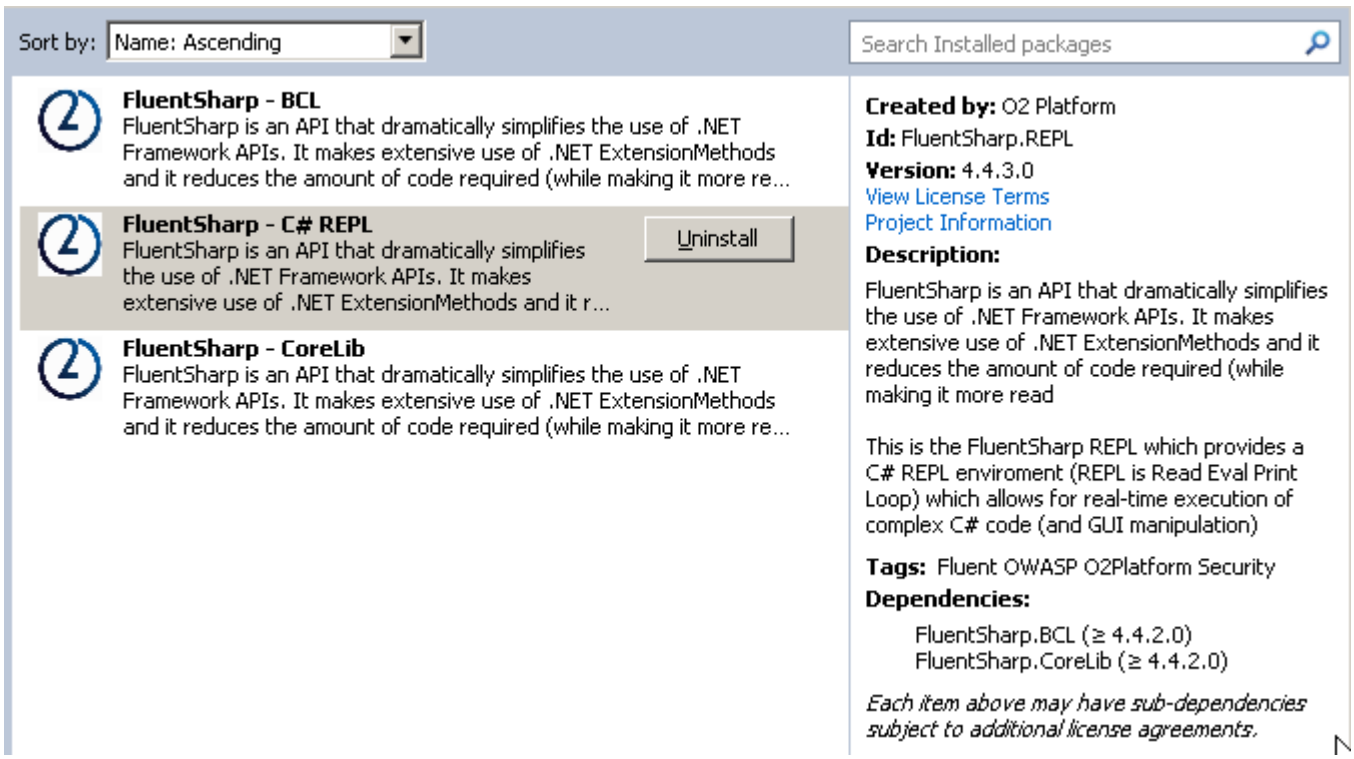
ManagedSpy - Adding new features to ManagedSpy.exe (part 1)

Using the source code



Add the O2 FluentSharp C# REPL reference (via NuGet)



Note: you only need to add the ***C# REPL*** one, since the other two (***CoreLib*** and ***BCL***) will be automatically installed

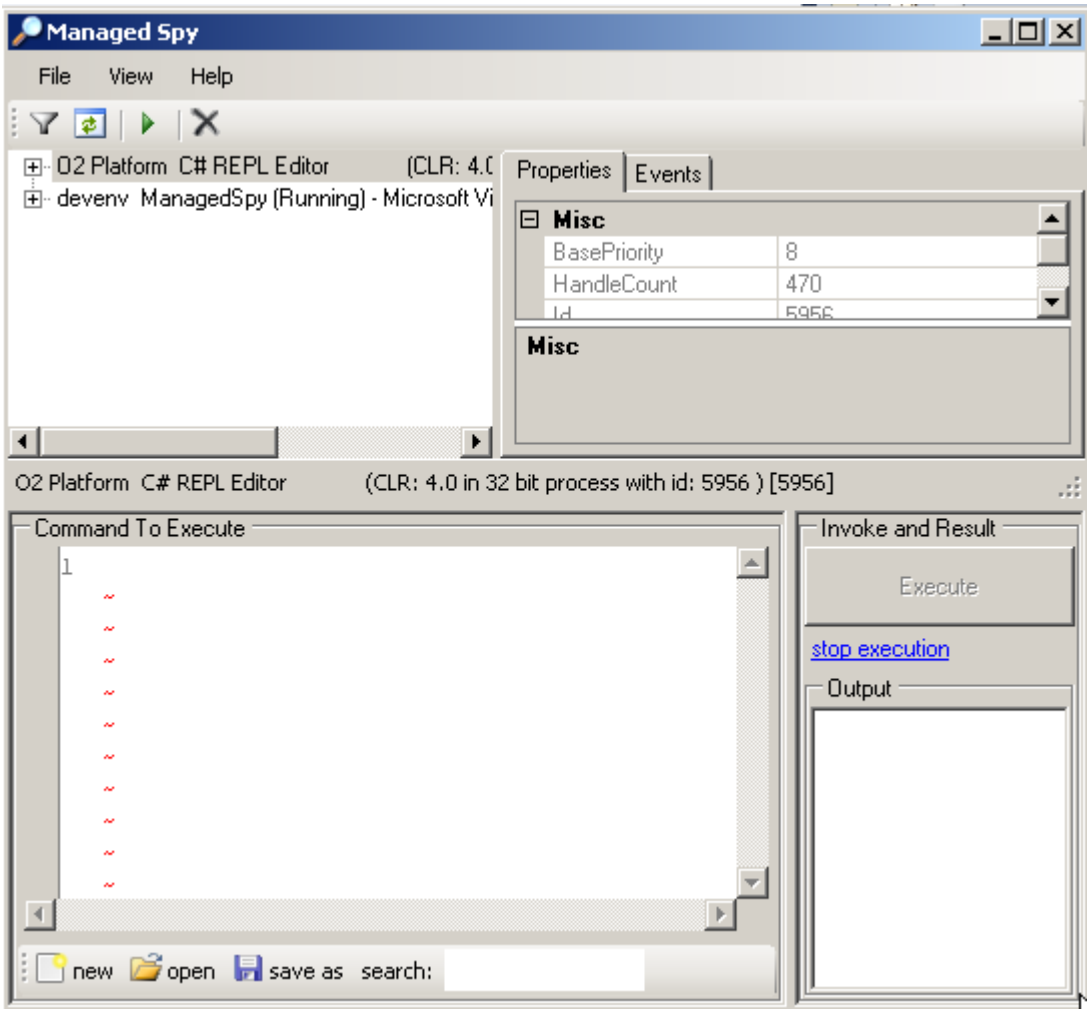
Open the Form1.cs file and add a C# REPL script editor

```
        public Form1()
```

```
        {
            InitializeComponent();

            //adding O2
            this.insert_Script();
        }
```

```
Form1.cs | Program.cs | ReadMe.txt | Commands.cpp | Messages.h | PropertyDescriptorPr
ManagedSpy.Form1                                   Form1()
22      public partial class Form1 : Form {
23
24          /// <summary>
25          /// Currently selected proxy -- used for event logging.
26          /// </summary>
27          private ControlProxy currentProxy = null;
28          EventFilterDialog dialog = new EventFilterDialog();
29
30          public Form1()
31          {
32              InitializeComponent();
33
34              //adding O2
35              this.insert_Script();
36          }
37
38          private void exitToolStripMenuItem_Click(object sender, Ever
39              Application.Exit();
40          }
```

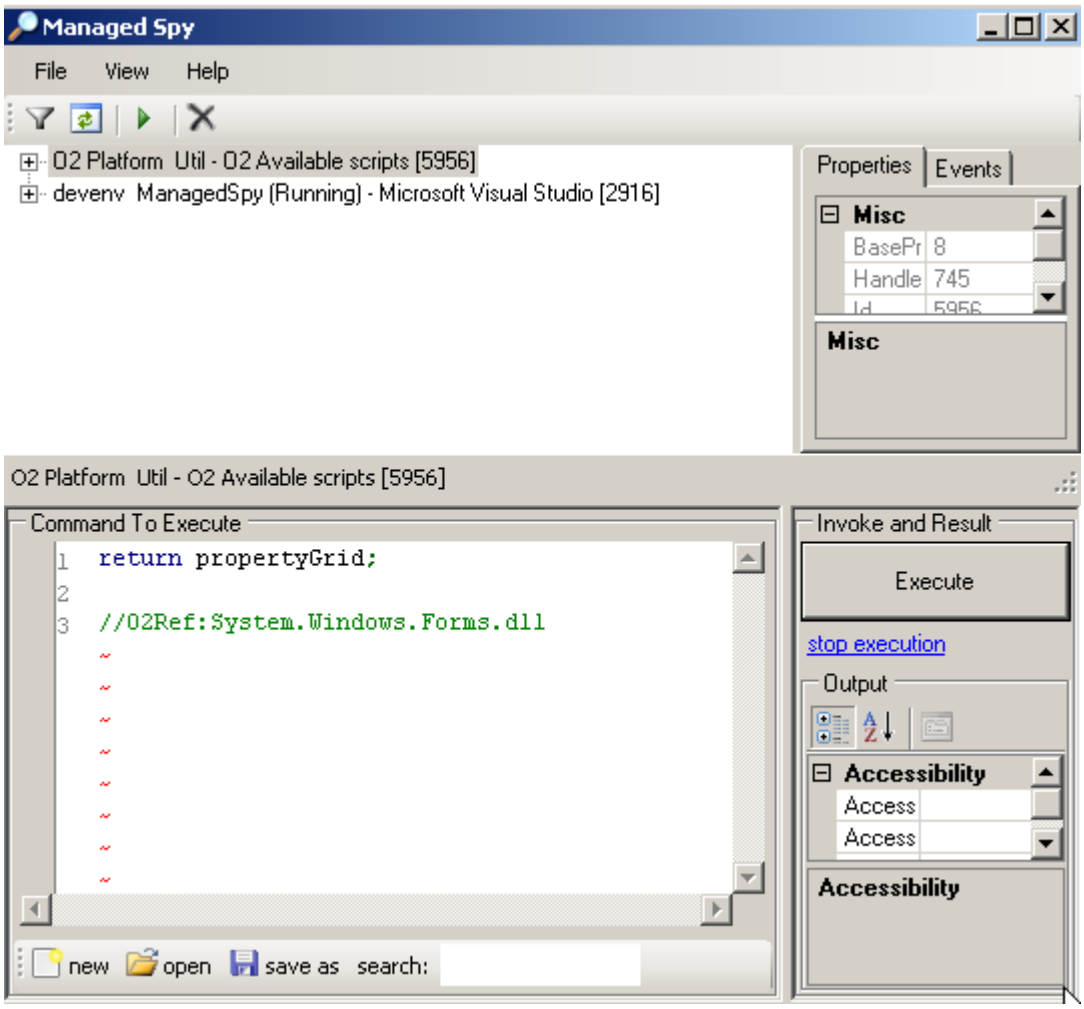
which will look like this:



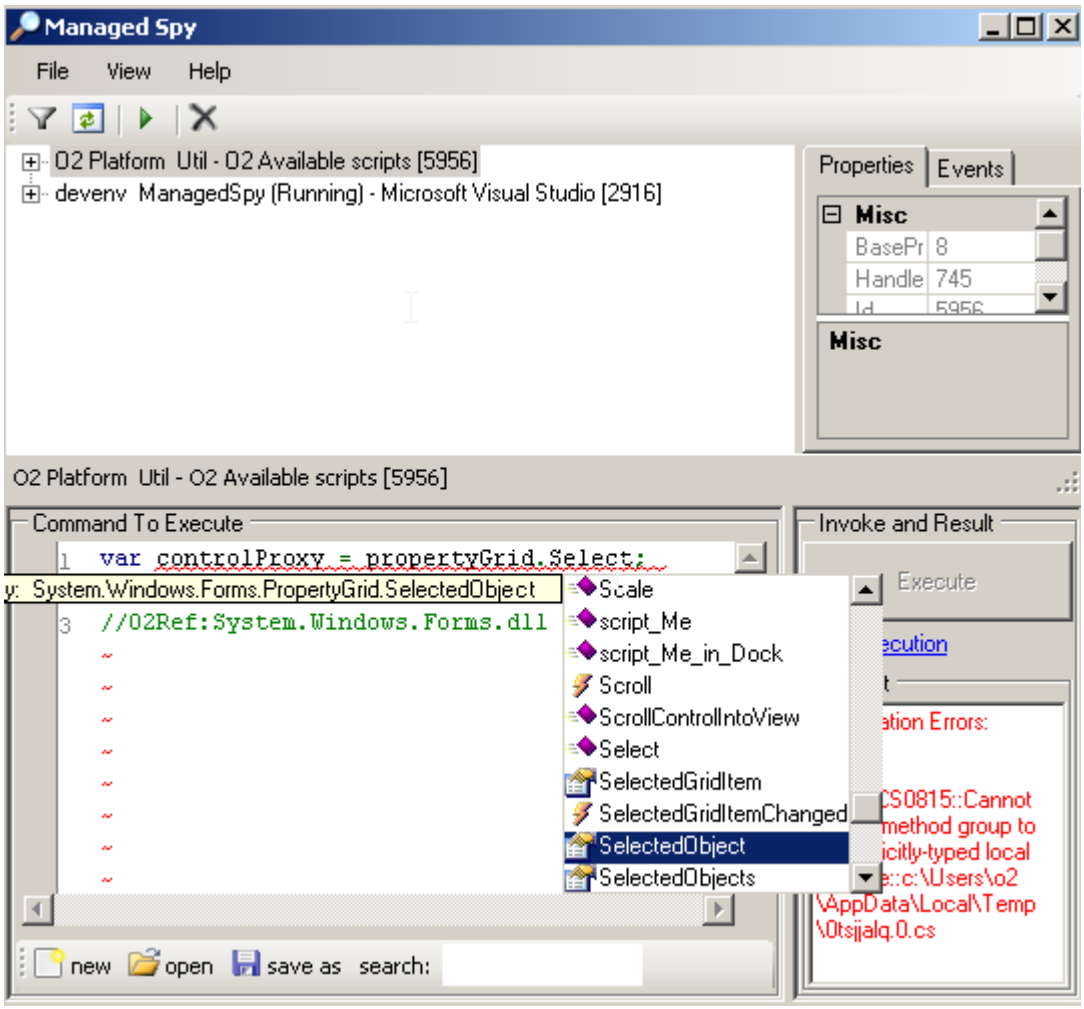
It's more useful if we add a Script_Me C# Gui with the property grid

```
public Form1()
{
    InitializeComponent();

    //adding O2
    this.insert_Below_Script_Me(propertyGrid);
}
```

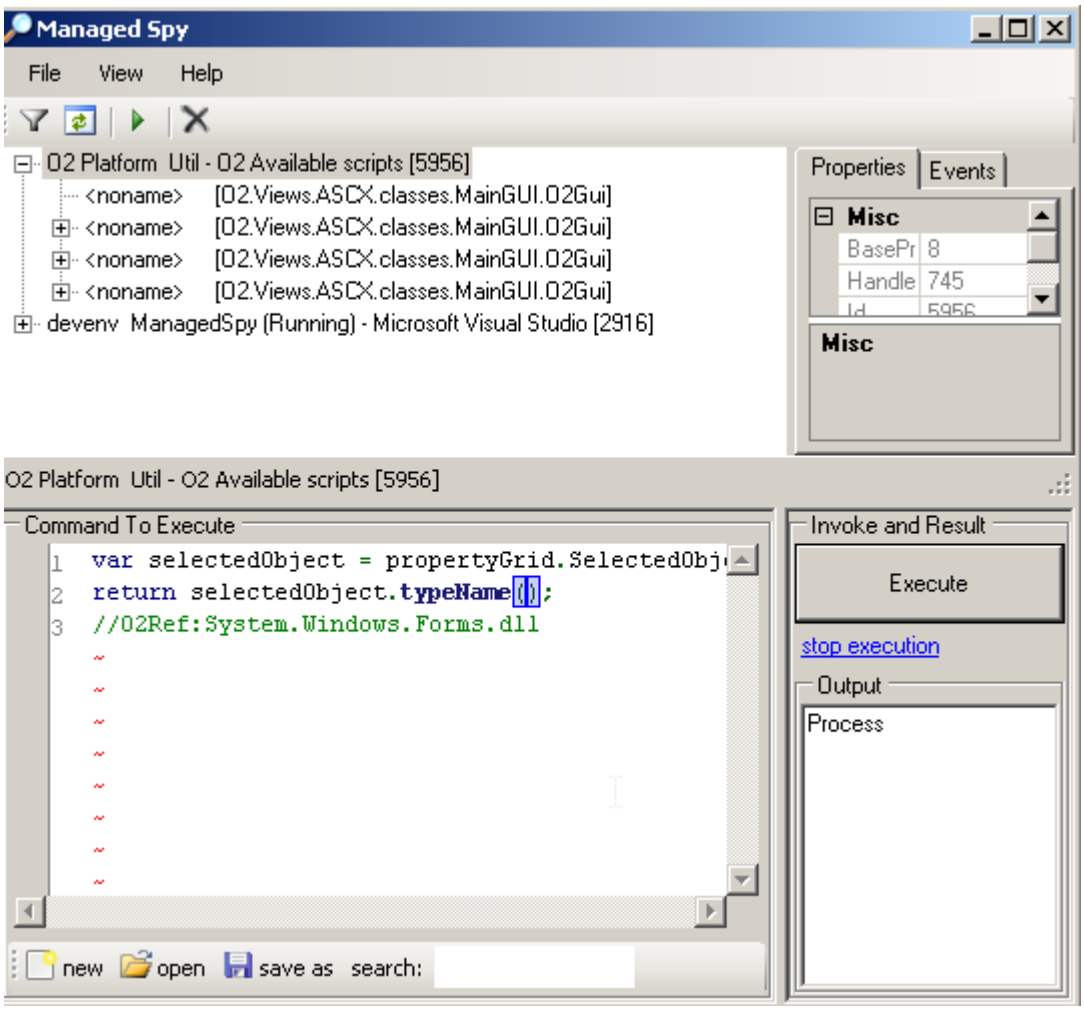
This Script_Me enviroment is one where a parameter (in this case the *propertyGrid*) is made available for scripting



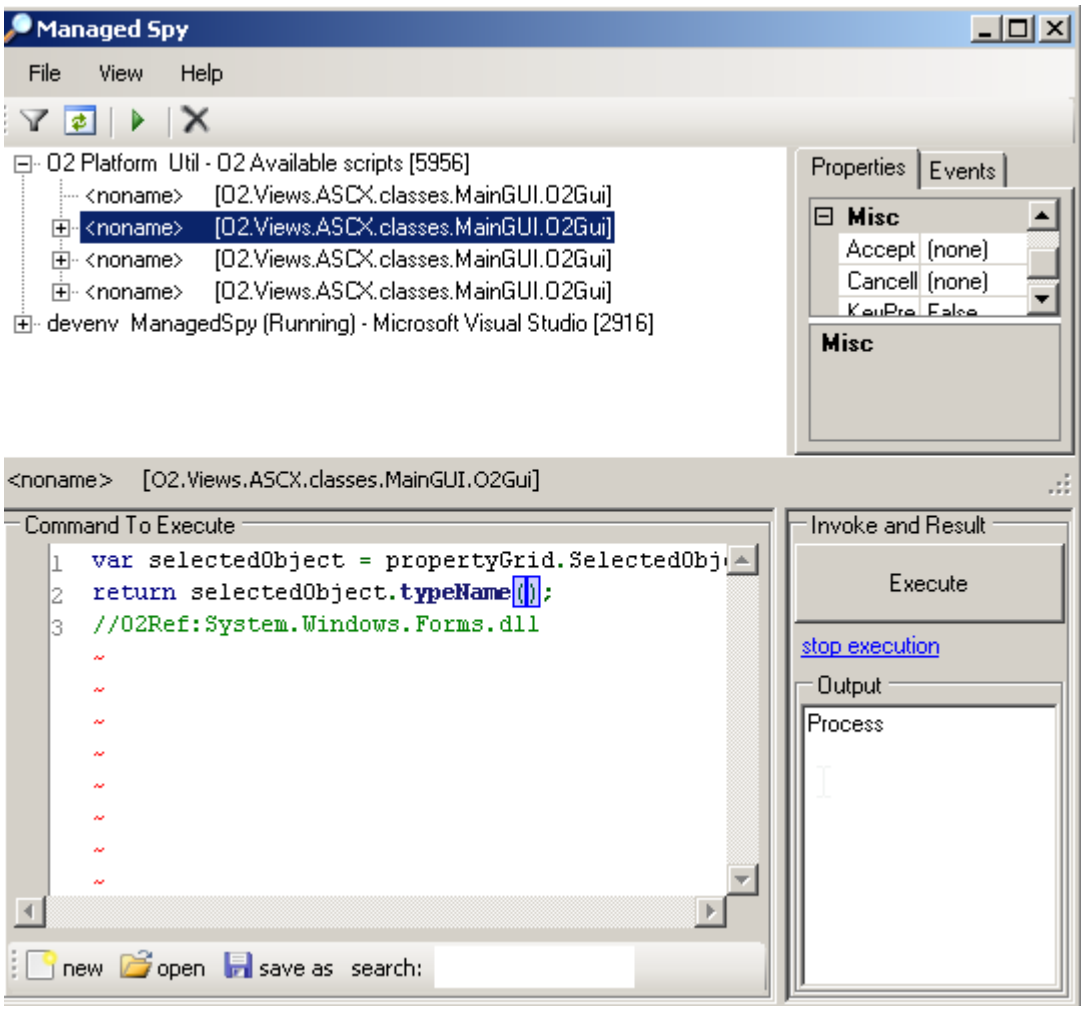
And there is Code Complete help:



What happens in the ManagedSpy control is that the root note is the selected process:

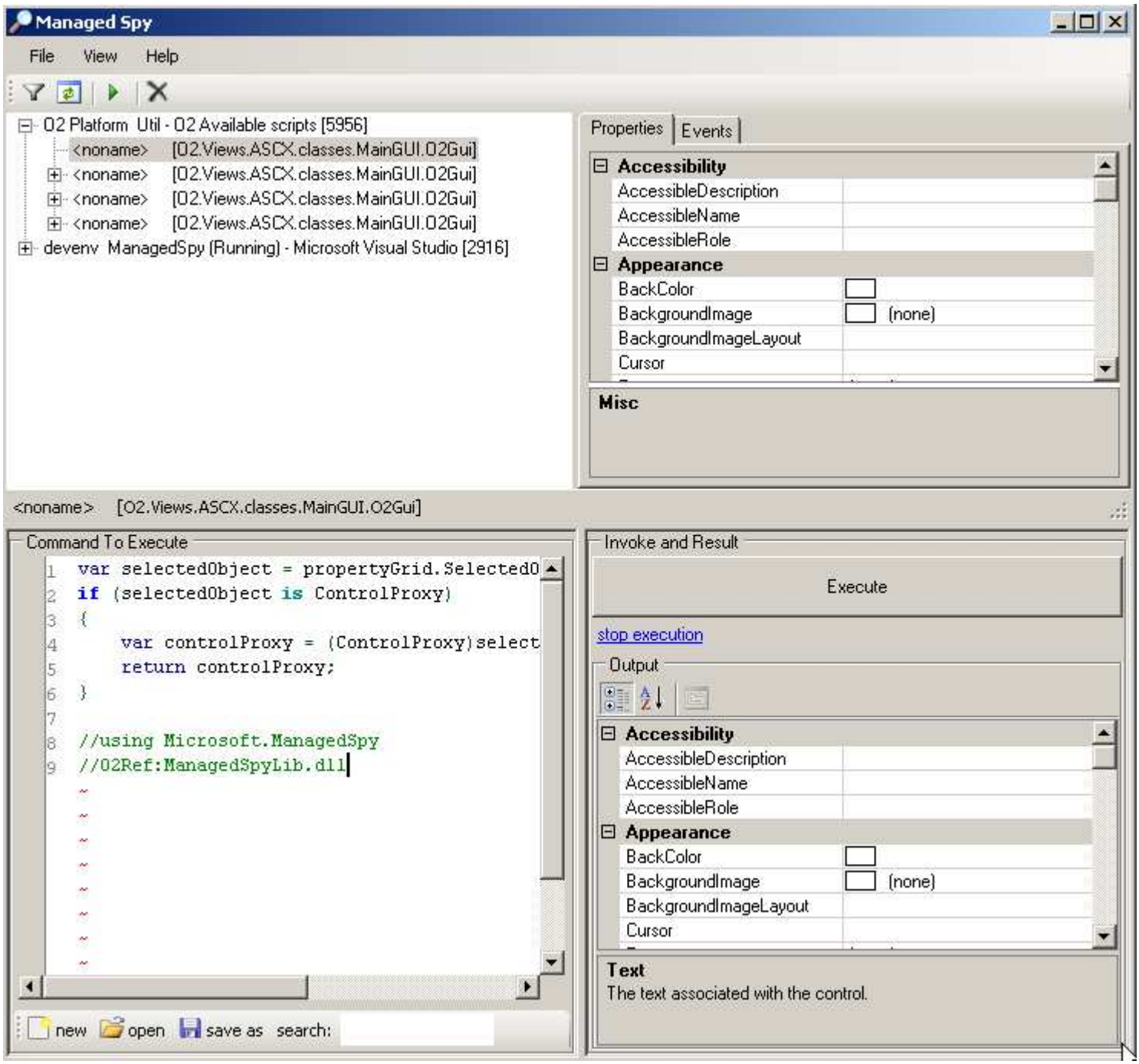


and the child nodes are the ControlProxy objects:



Importing the reference to ManagedSpyLib.dll and accessing the current ControlProxy object:

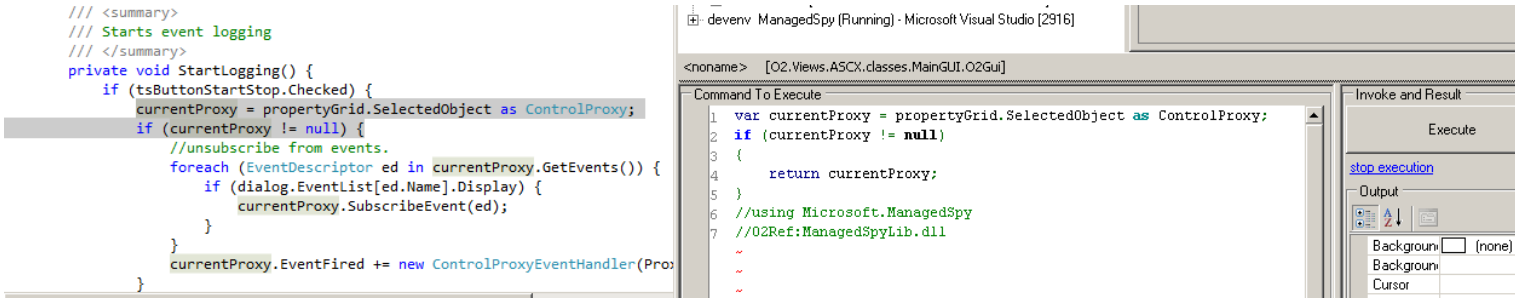
```
var selectedObject = propertyGrid.SelectedObject;
if (selectedObject is ControlProxy)
{
	var controlProxy = (ControlProxy)selectedObject;
	return controlProxy;
}

//using Microsoft.ManagedSpy
//O2Ref:ManagedSpyLib.dll
```



We can also use

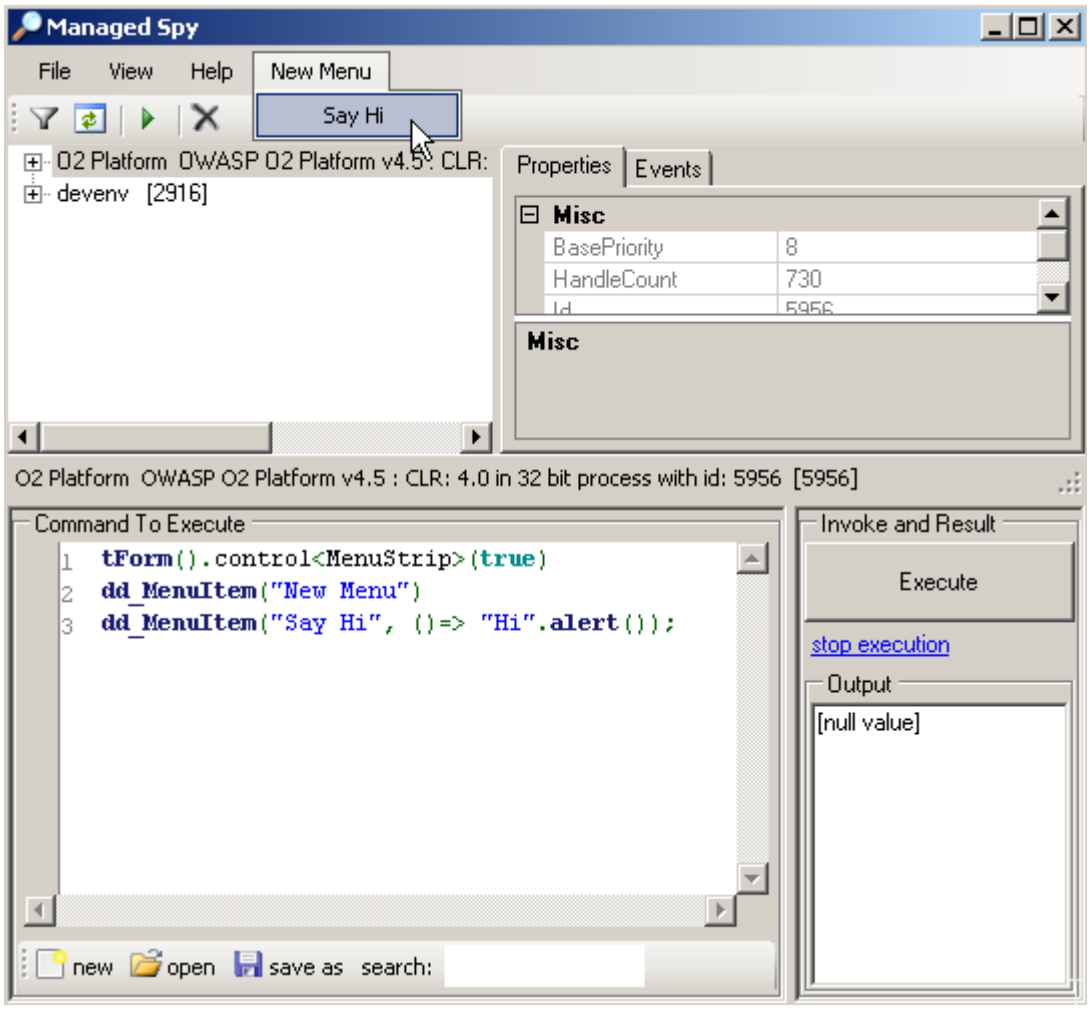
```
var currentProxy = propertyGrid.SelectedObject as ControlProxy;
if (currentProxy != null)
{
	return currentProxy;
}
//using Microsoft.ManagedSpy
//O2Ref:ManagedSpyLib.dll
```

Which is similar to the code used in the main ManagedSpy.exe

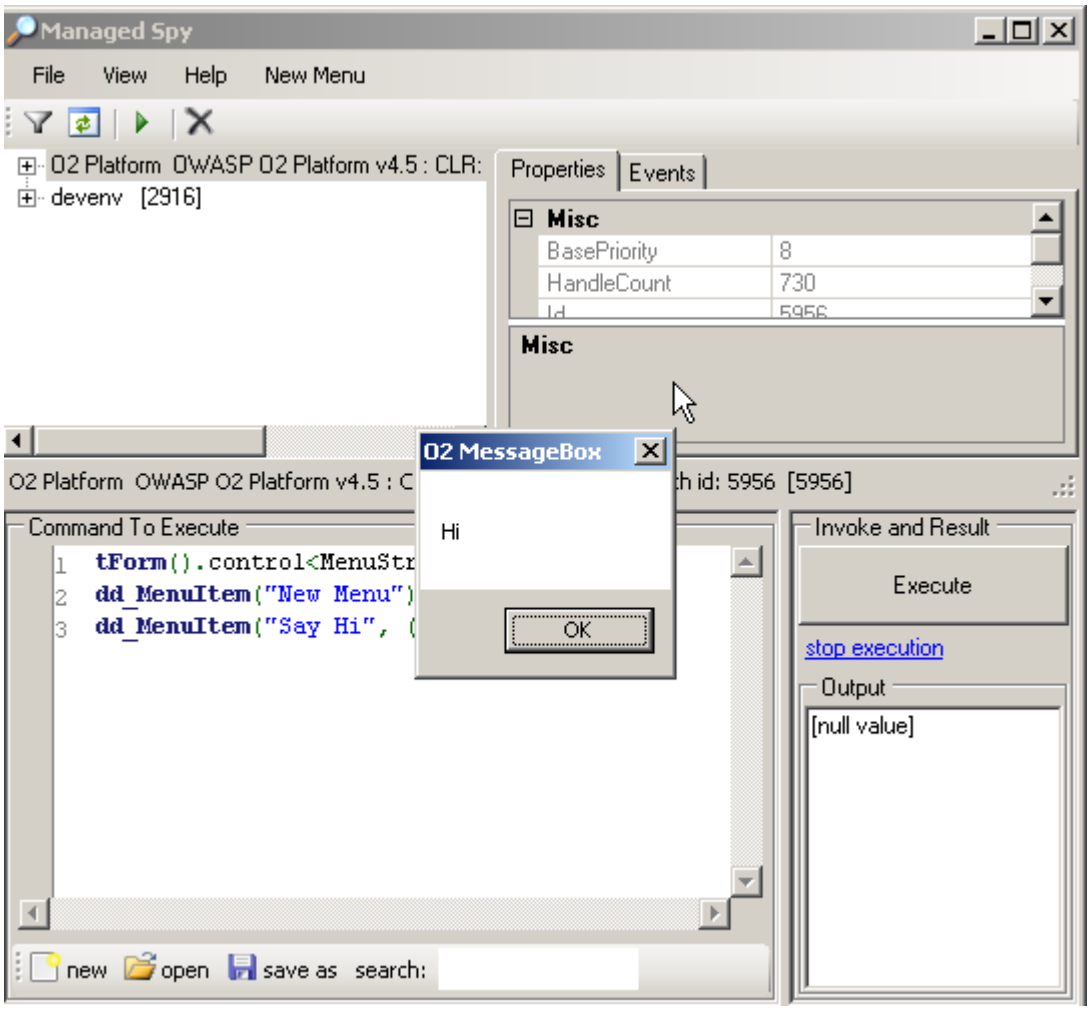


With the PropertyGrid references we add new features to the main GUI, like for example a menu and menu item

```
propertyGrid.parentForm().control<MenuStrip>(true)
                         .add_MenuItem("New Menu")
                         .add_MenuItem("Say Hi", ()=> "Hi".alert());
```

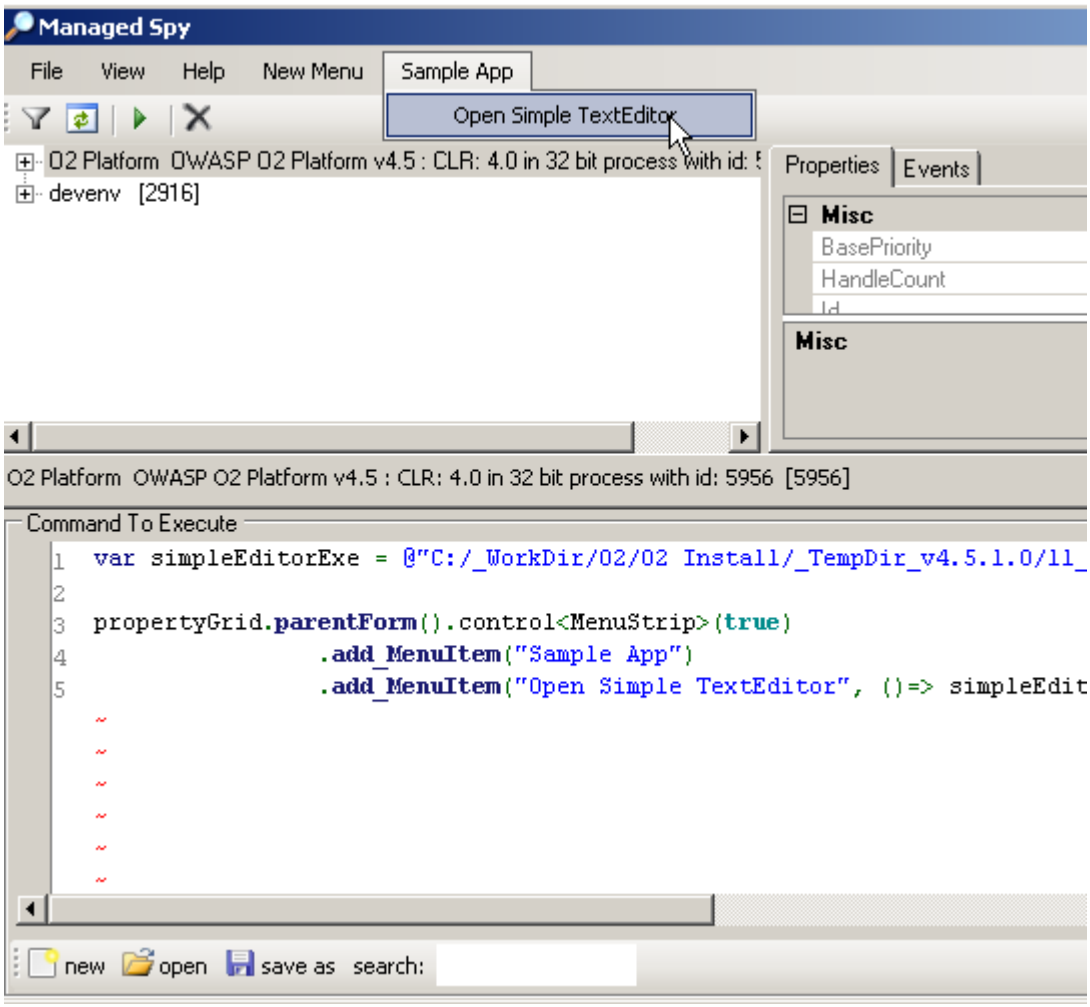


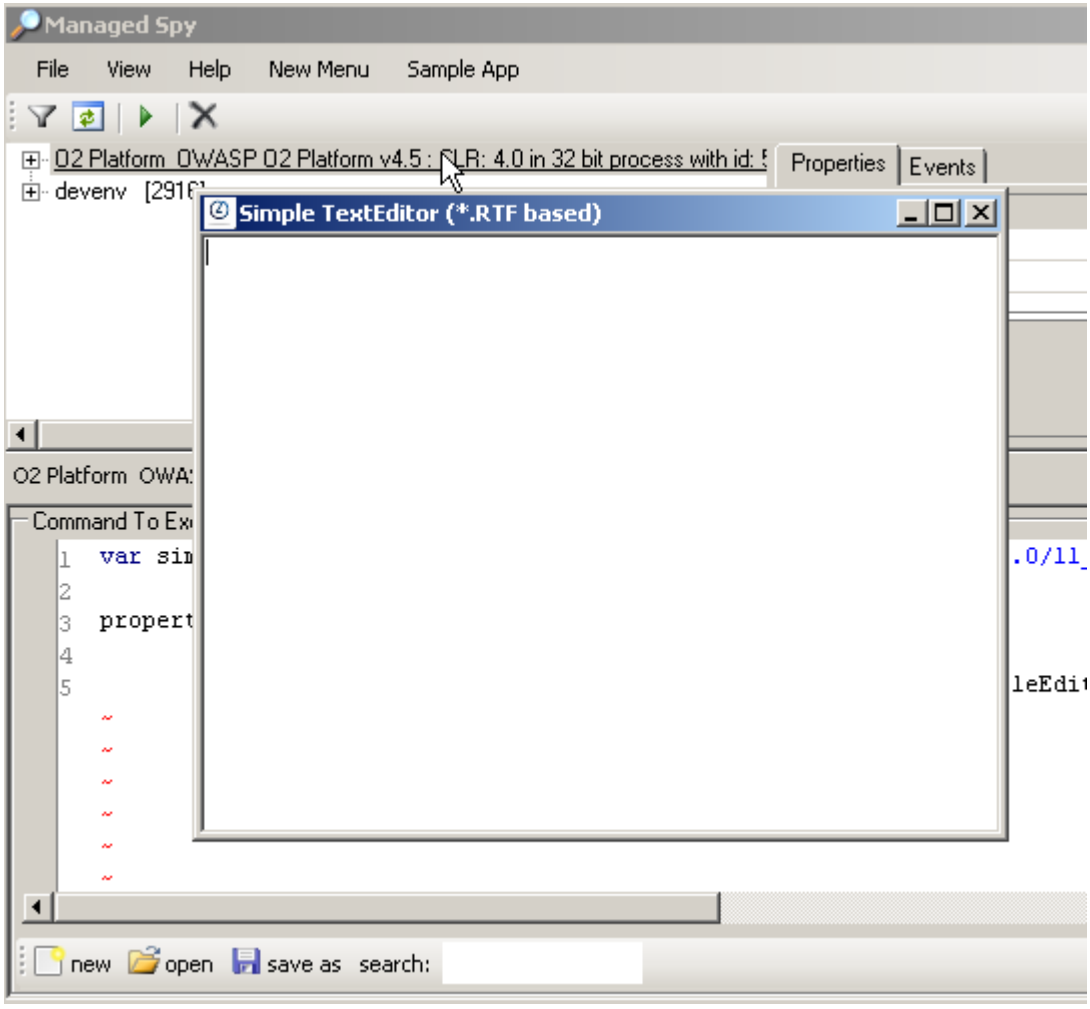
Which when clicked will invoke *"Hi".alert()*:



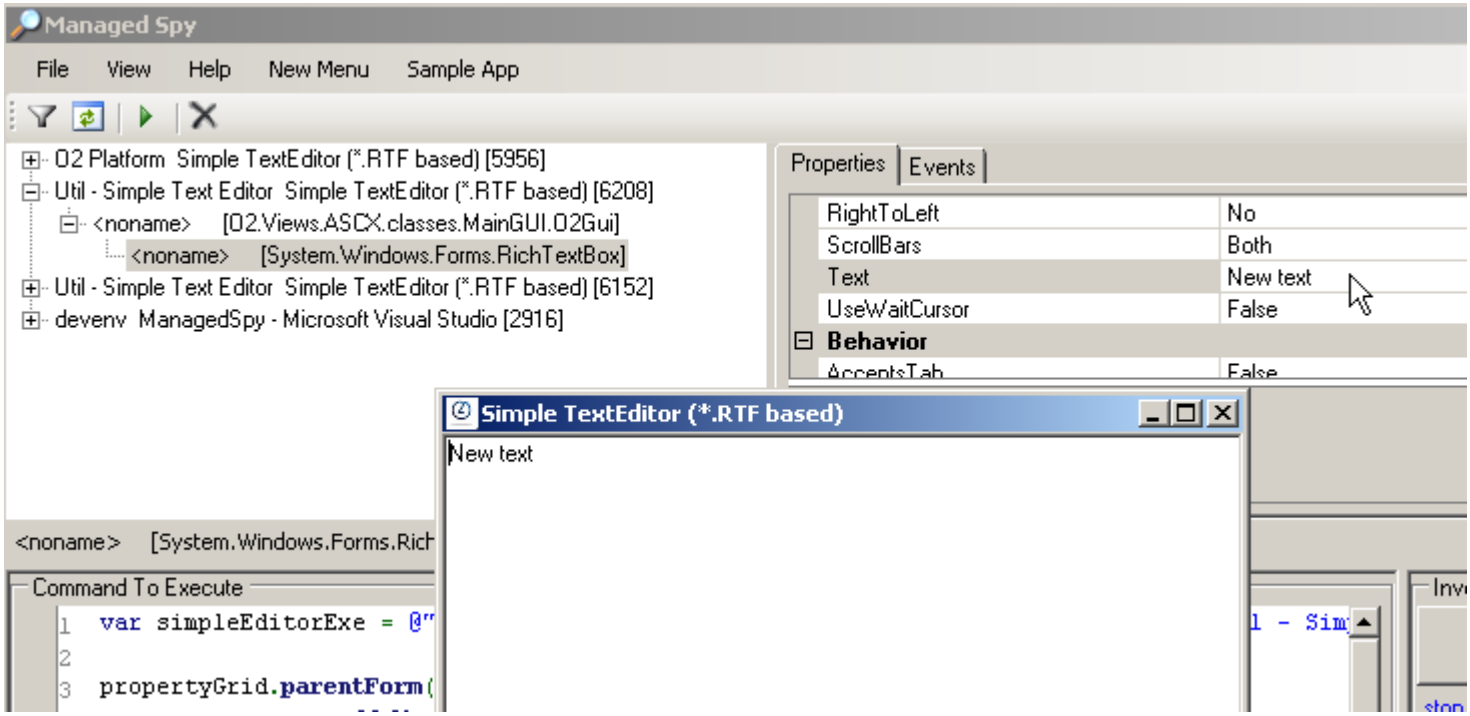
More usefull is a menu that opens an sample app:

```
var simpleEditorExe = @"C:/_WorkDir/O2/O2 Install/_TempDir_v4.5.1.0/11_17_2012/Util - Simple Text Editor
[18704]\Util - Simple Text Editor.exe";


propertyGrid.parentForm().control<MenuStrip>(true)
                         .add_MenuItem("Sample App")
                         .add_MenuItem("Open Simple TextEditor", ()=> simpleEditorExe.startProcess());
```


Managed Spy
File
View
Help
New Menu
Sample App
Open Simple TextEditor
O2 Platform OWASP O2 Platform v4.5 : CLR: 4.0 in 32 bit process with id: 5
devenv [2916]
Properties
Events
Misc
BasePriority
HandleCount
Misc
O2 Platform OWASP O2 Platform v4.5 : CLR: 4.0 in 32 bit process with id: 5956 [5956]
Command To Execute
1 var simpleEditorExe = @"C:/_WorkDir/02/02 Install/_TempDir_v4.5.1.0/ll_
2
3 propertyGrid.parentForm().control<MenuStrip>(true)
4 .add_MenuItem("Sample App")
5 .add_MenuItem("Open Simple TextEditor", ()=> simpleEdit
new
open
save as
search:

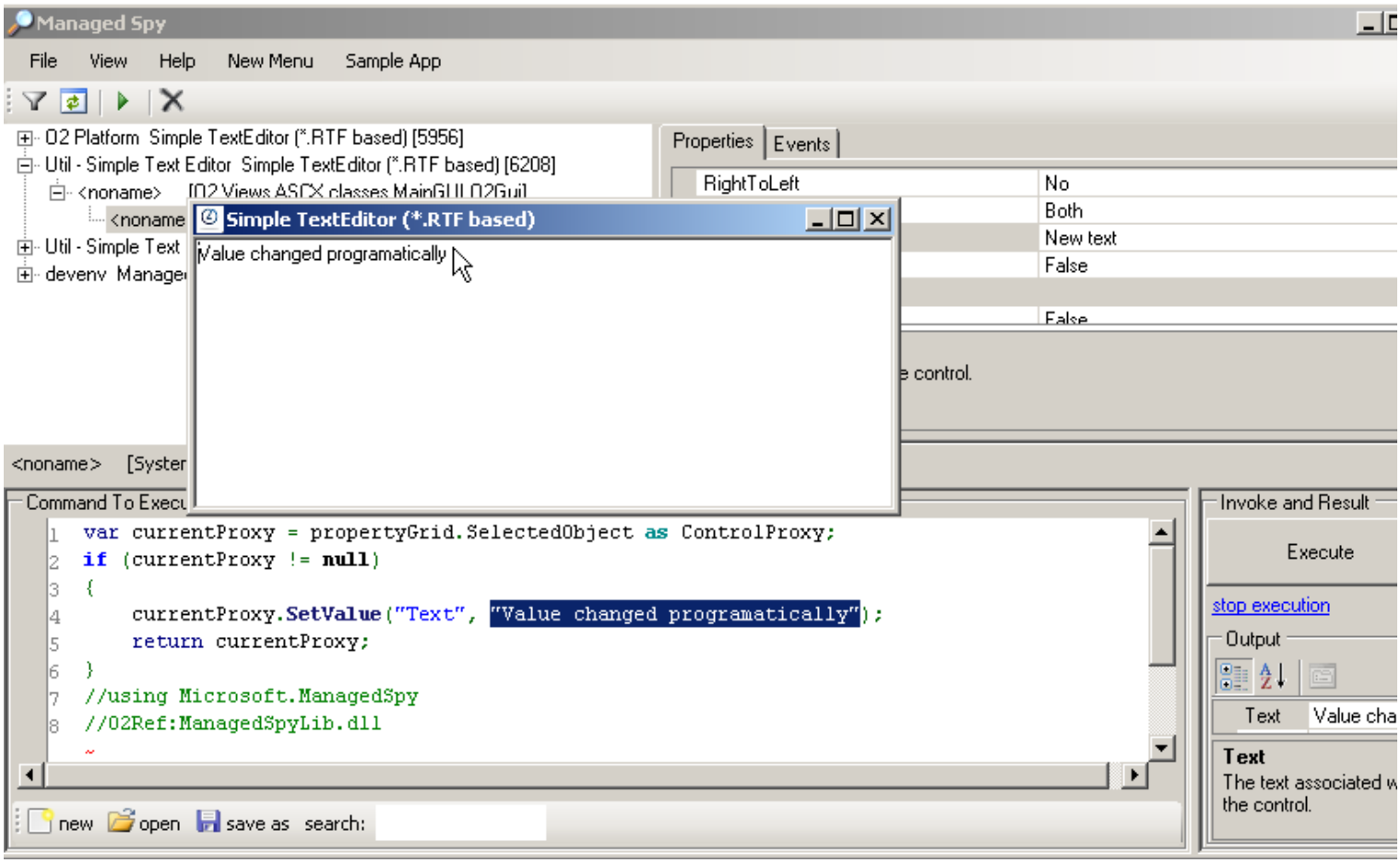

Managed Spy
File
View
Help
New Menu
Sample App
Properties
Events
Simple TextEditor (*.RTF based)
new
open
save as
search:

Click refresh, and note how we can edit the value of the TextBox via the original ManagedSpy.exe property grid

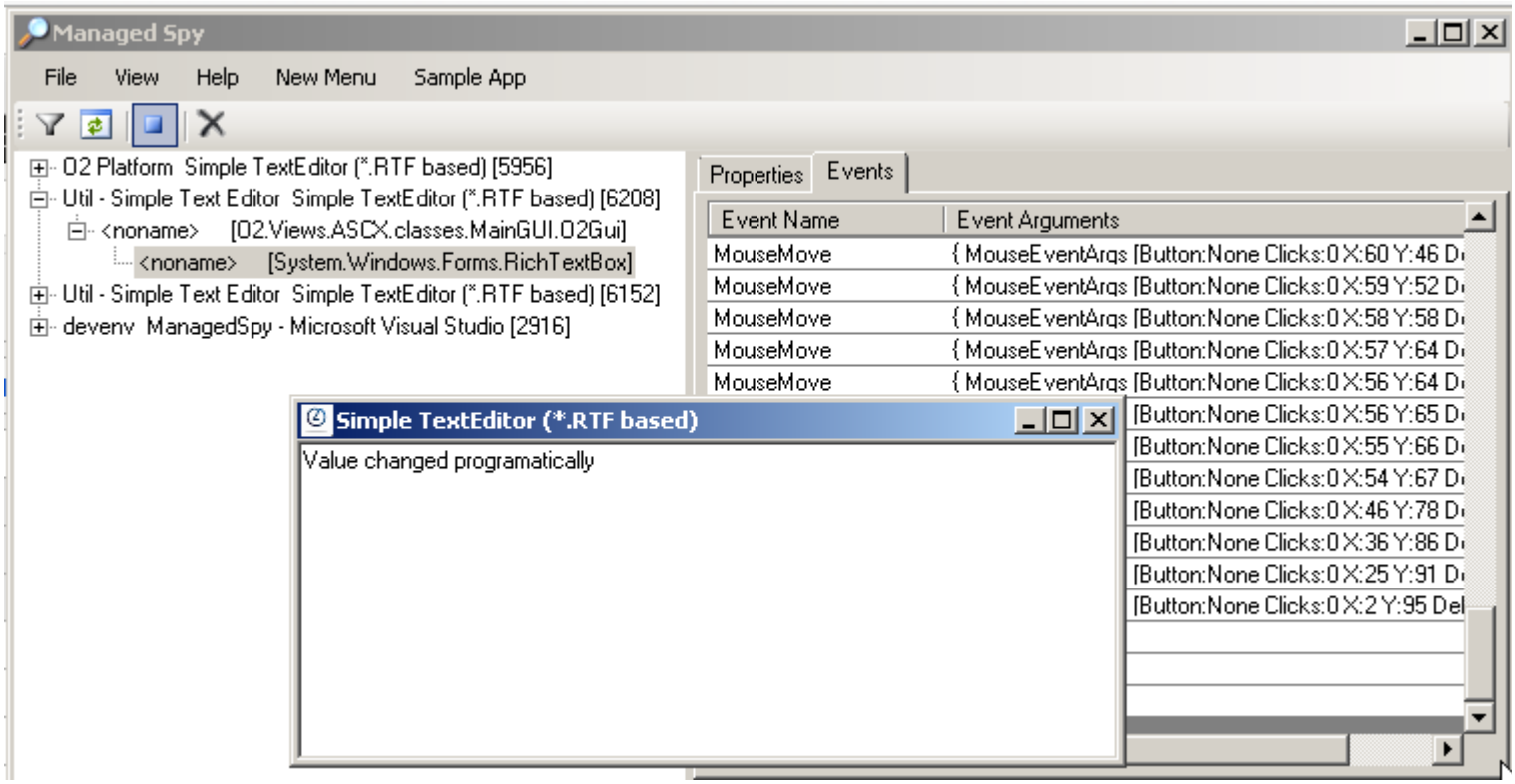


Or programatically

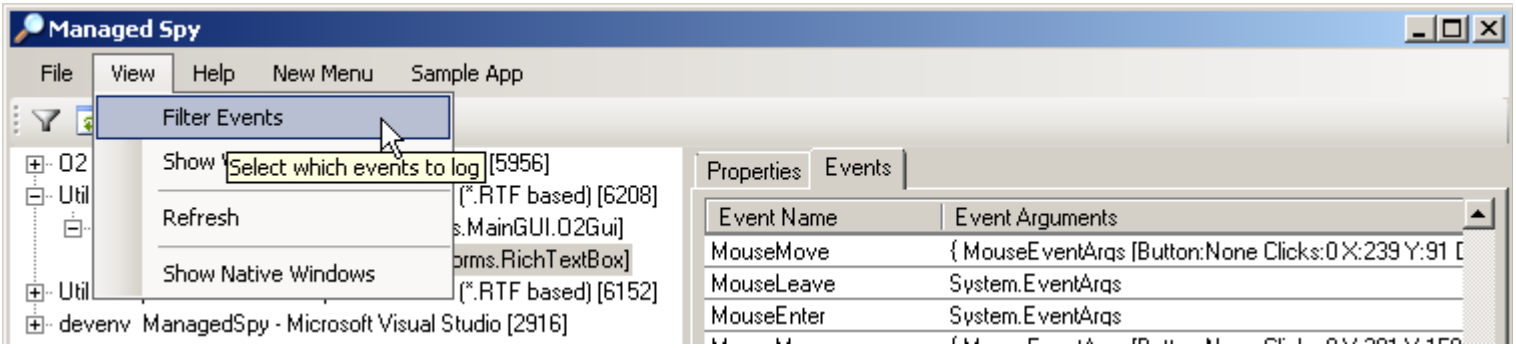
```
var currentProxy = propertyGrid.SelectedObject as ControlProxy;
if (currentProxy != null)
{
	currentProxy.SetValue("Text", "Value changed programatically");
	return currentProxy;
}
//using Microsoft.ManagedSpy
//O2Ref:ManagedSpyLib.dll
```



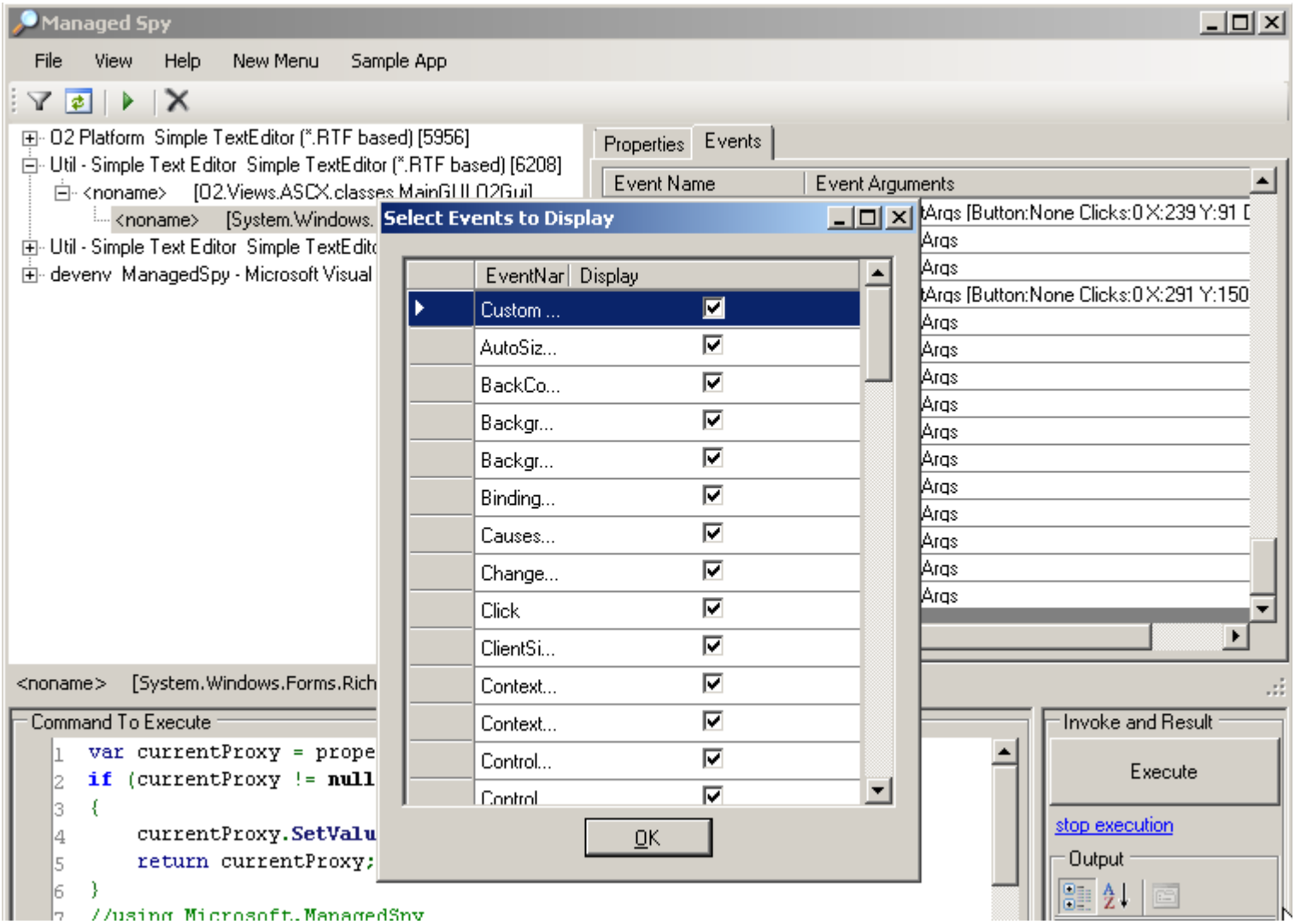
Open the Events Tab and press play to see the events:



The Events to capure are seleted via the View menu



which will open up Model Dialog:



This is a *EventFilterDialog*

```
22      public partial class Form1 : Form {
23
24          /// <summary>
25          /// Currently selected proxy -- used for event logging.
26          /// </summary>
27          private ControlProxy currentProxy = null;
28          EventFilterDialog dialog = new EventFilterDialog();
```

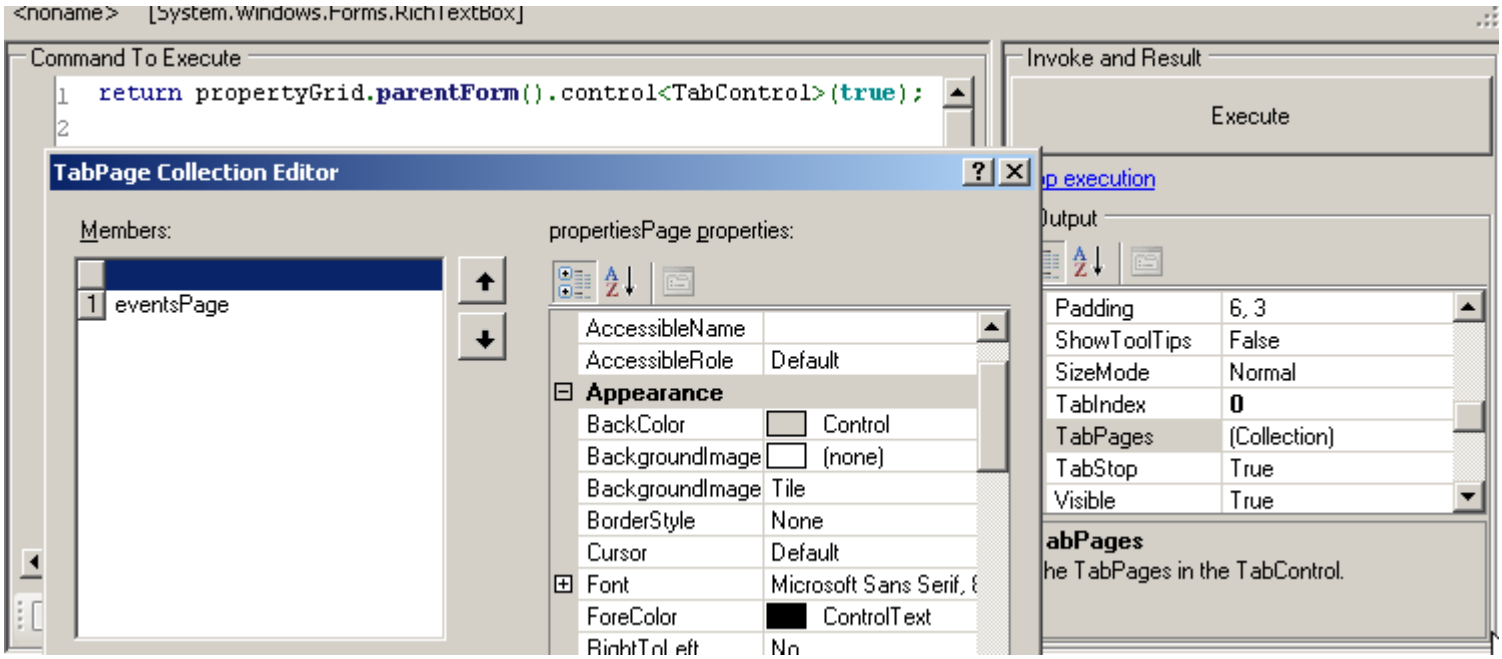
shown via

```
219         private void tsbuttonFilterEvents_Click(object sender, EventArgs e) {
220             dialog.ShowDialog();
221             StopLogging();
222             StartLogging();
223         }
```

Lets try putting in on a more usefull location

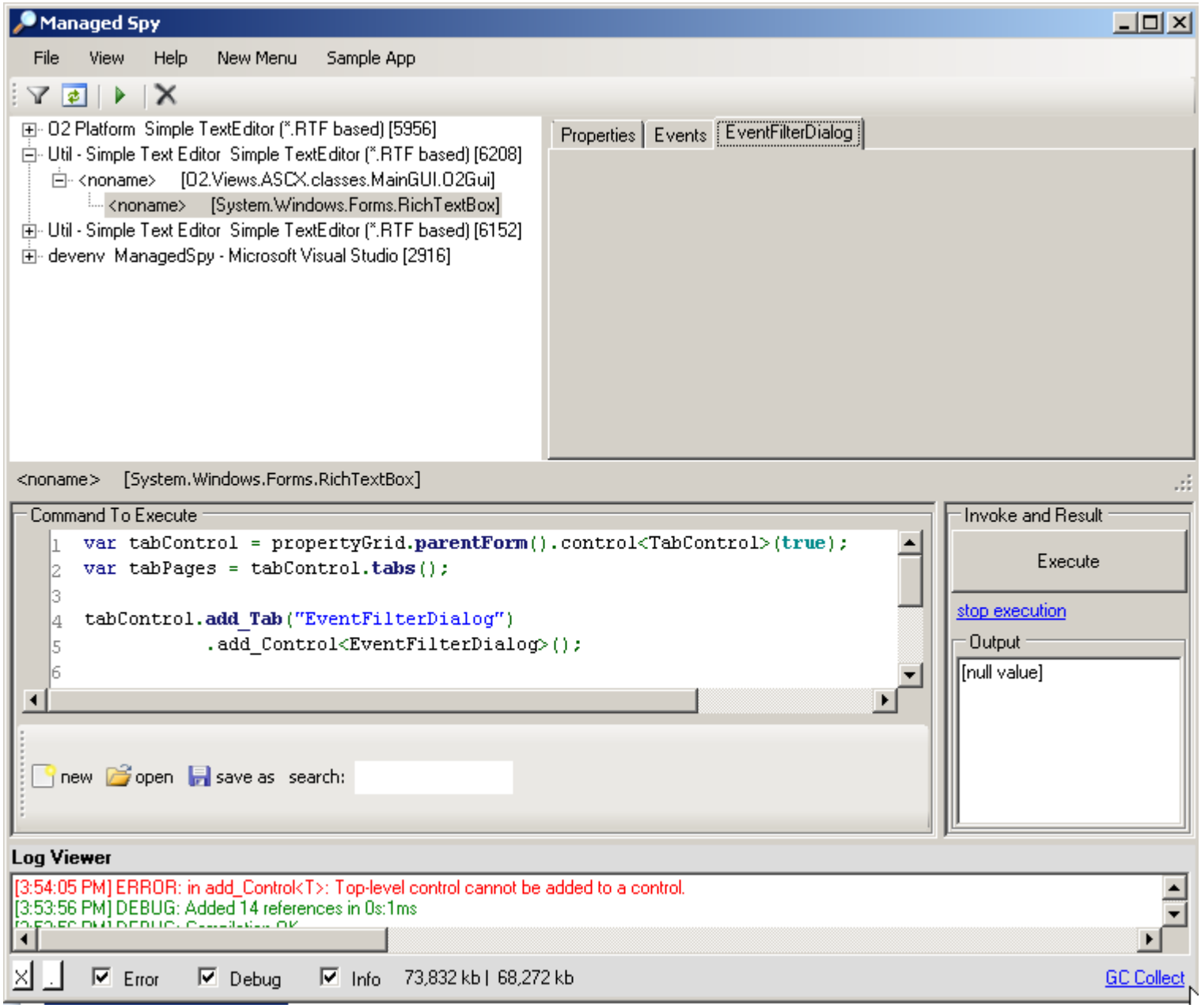
Here is the main TabControl:

```
return propertyGrid.parentForm().control<TabControl>(true);
```



We can add a new TabPage with an *`EventFilterDialog`* Object inside it

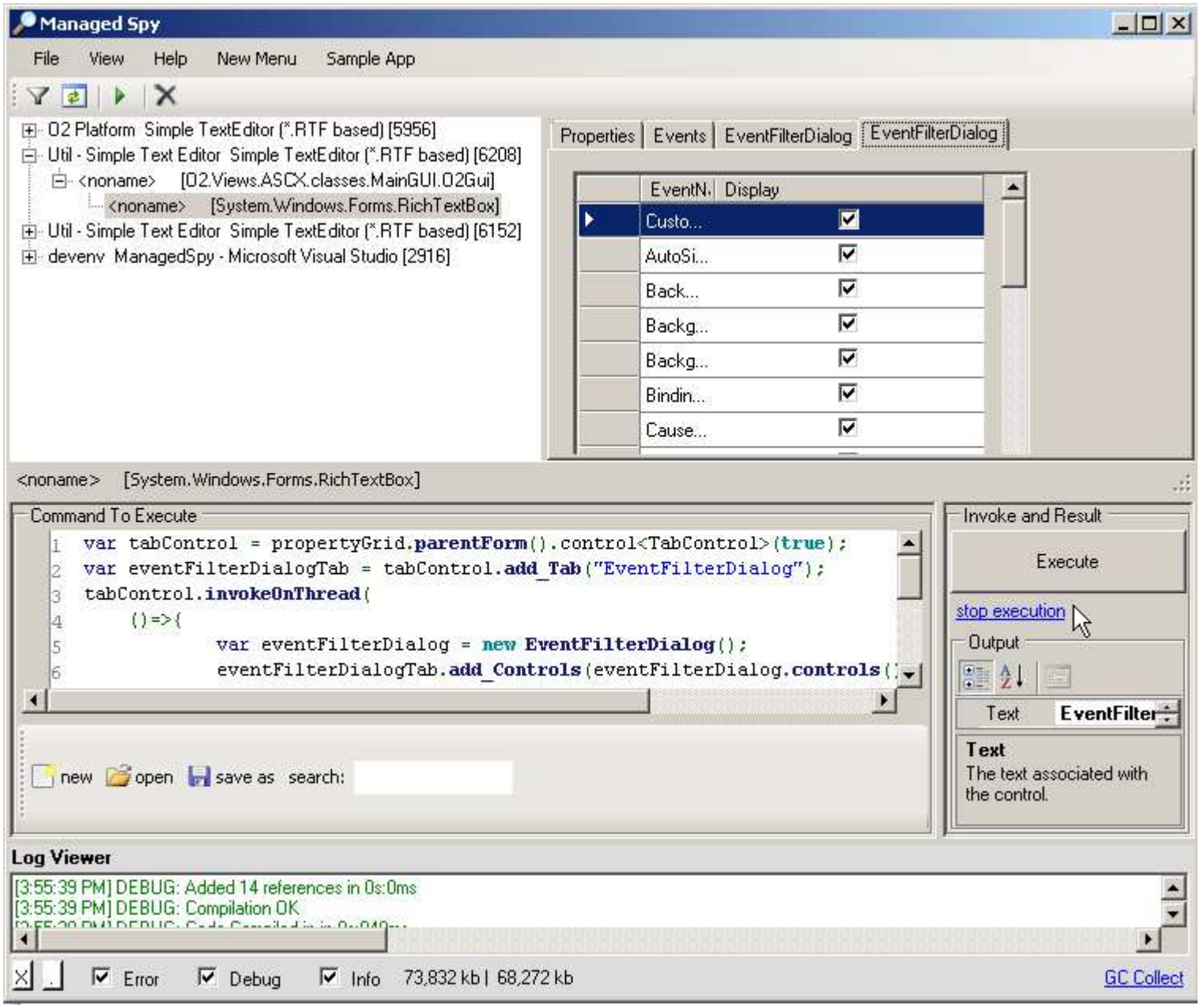
```
var tabControl = propertyGrid.parentForm().control<TabControl>(true);
tabControl.add_Tab("EventFilterDialog")
                .add_Control<EventFilterDialog>();


//using ManagedSpy;
//O2Ref:ManagedSpy.exe
```



The previous code didn't work because *`EventFilterDialog`* is a Form (and Forms cannot be added as child controls)

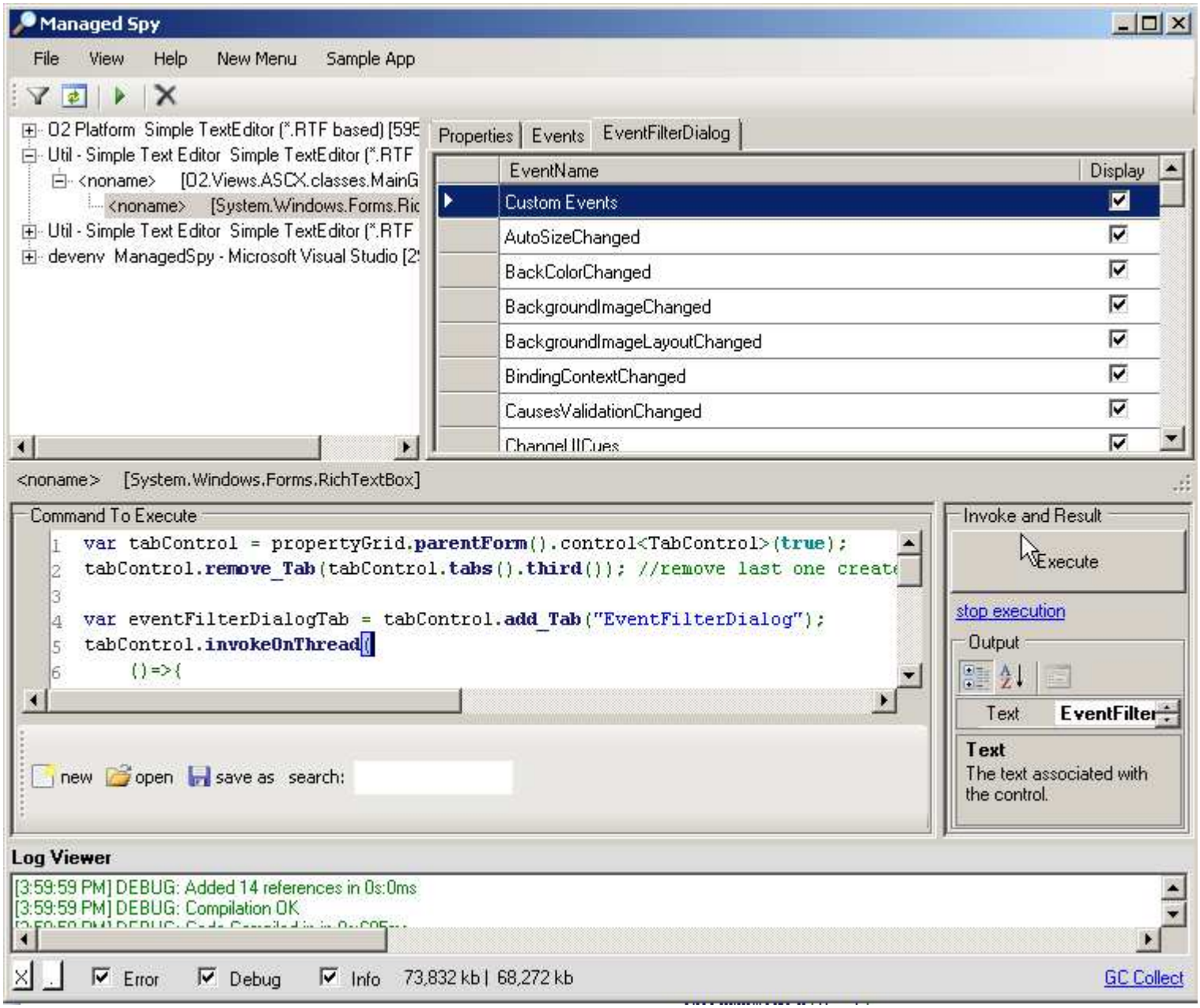
So we need to do something like this

```
var tabControl = propertyGrid.parentForm().control<TabControl>(true);
var eventFilterDialogTab = tabControl.add_Tab("EventFilterDialog");
tabControl.invokeOnThread(
       ()=>{
                    var eventFilterDialog = new EventFilterDialog();
                    eventFilterDialogTab.add_Controls(eventFilterDialog.controls());
             });
return eventFilterDialogTab;

//using ManagedSpy;
//O2Ref:ManagedSpy.exe
```



In fact we only want the DataGridView control (with better column width sizes)

```
var tabControl = propertyGrid.parentForm().control<TabControl>(true);
tabControl.remove_Tab(tabControl.tabs().third()); //remove last one created

var eventFilterDialogTab = tabControl.add_Tab("EventFilterDialog");
tabControl.invokeOnThread(
       ()=>{
                    var eventFilterDialog = new EventFilterDialog();
                    var dataGridView = eventFilterDialog.control<DataGridView>()
                                                        .columnWidth(0,-1)
                                                        .columnWidth(1,50);
                    eventFilterDialogTab.add_Control(dataGridView);
                    tabControl.select_Tab(eventFilterDialogTab);
             });
return eventFilterDialogTab;

//using ManagedSpy;
//O2Ref:ManagedSpy.exe
```



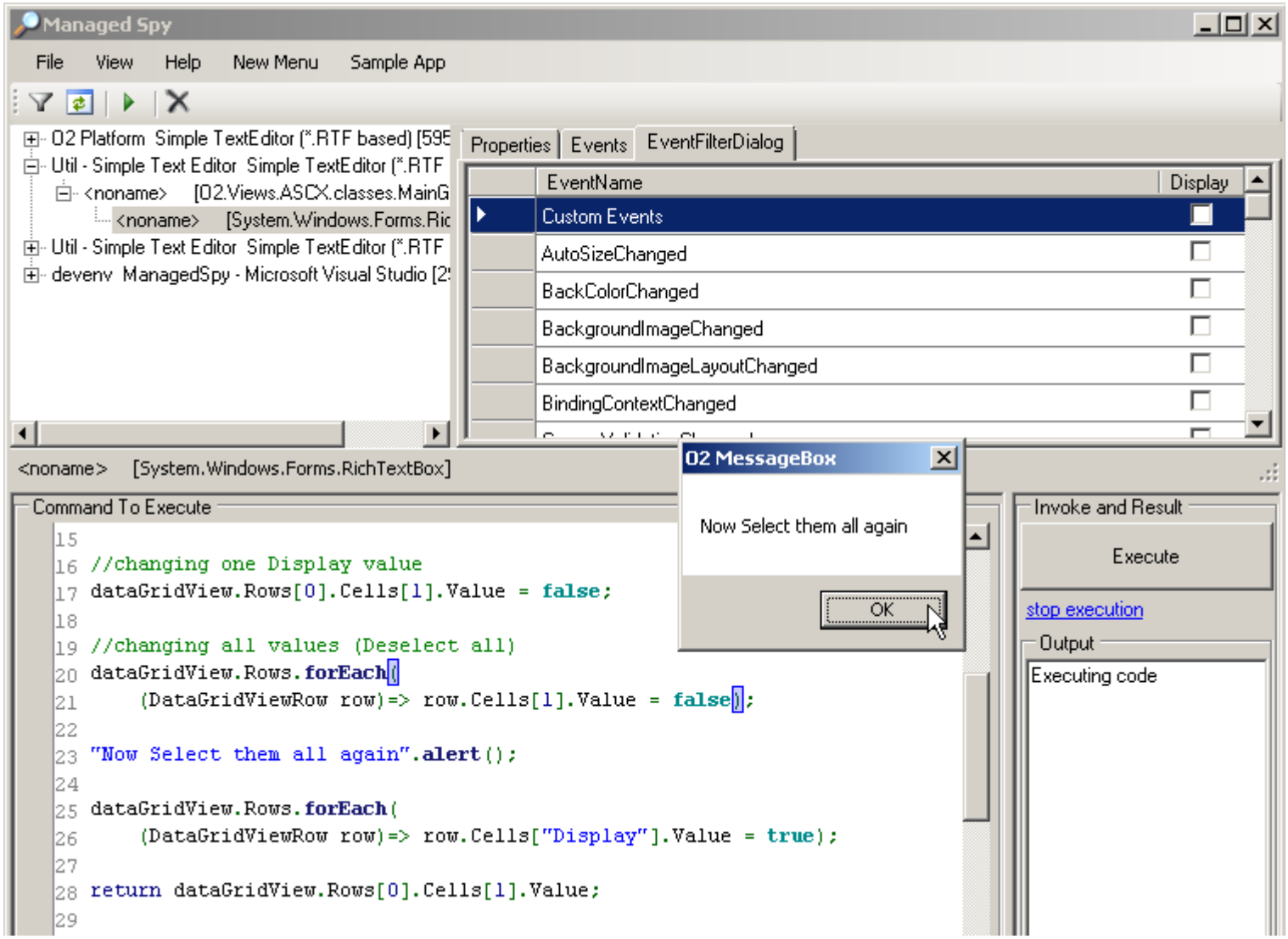
This code can be refactored to this:

```
var tabControl = propertyGrid.parentForm().control<TabControl>(true);
tabControl.remove_Tab(tabControl.tabs().third()); //remove last one created

var dataGridView = tabControl.newInThread<EventFilterDialog>()
                             .control<DataGridView>()
                                   .columnWidth(0,-1)
                                   .columnWidth(1,50);

tabControl.add_Tab("EventFilterDialog").add_Control(dataGridView);

tabControl.select_Tab(tabControl.tabs().third());

return "done";

//using ManagedSpy;
//O2Ref:ManagedSpy.exe.
```

Batch select or deselect

```
//changing one Display value
dataGridView.Rows[0].Cells[1].Value = false;

//changing all values (Deselect all)
dataGridView.Rows.forEach(
       (DataGridViewRow row)=> row.Cells[1].Value = false);
```

```
"Now Select them all again".alert();

dataGridView.Rows.forEach(
       (DataGridViewRow row)=> row.Cells["Display"].Value = true);
```

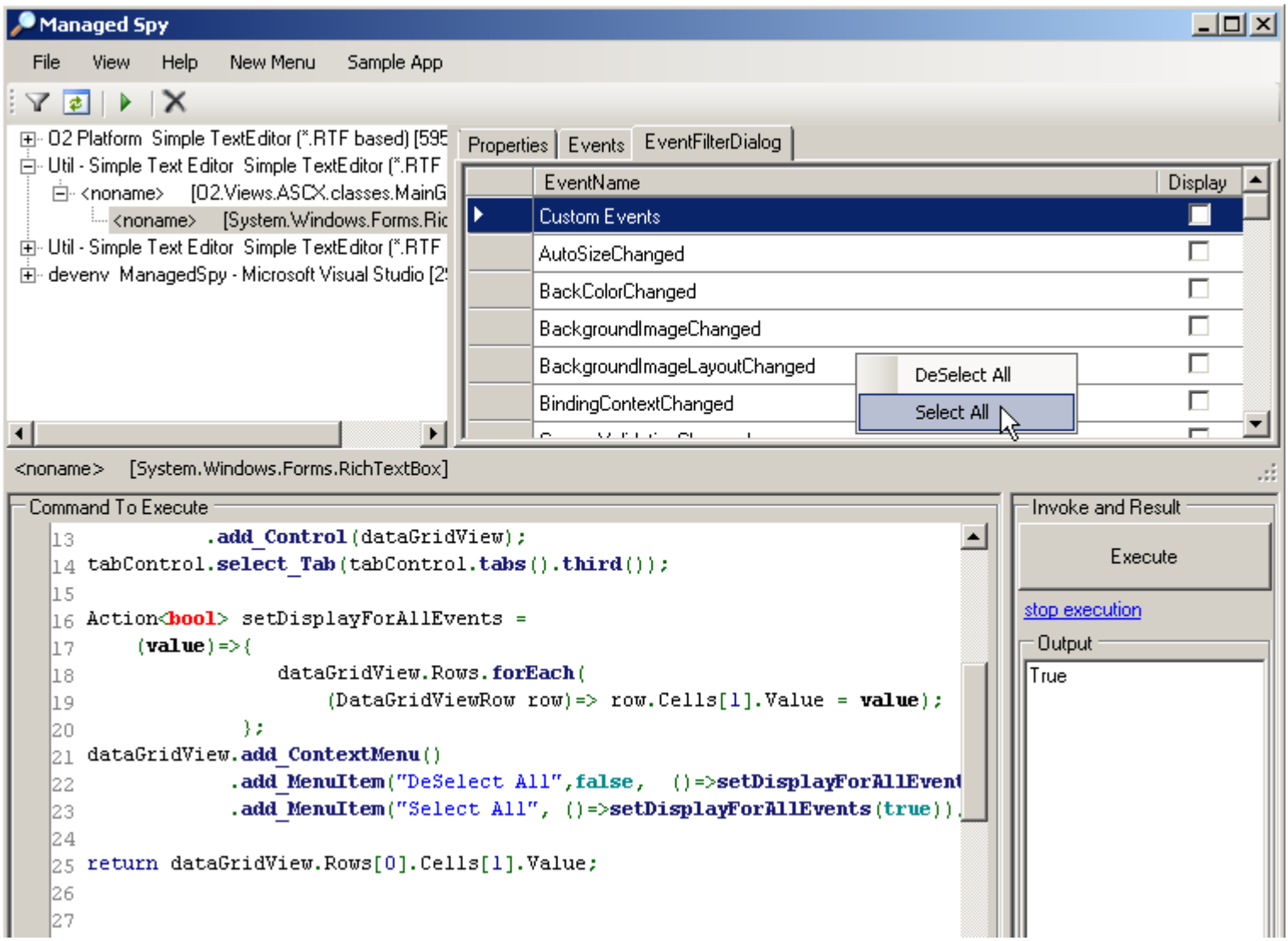


Previous code can be refactored to :

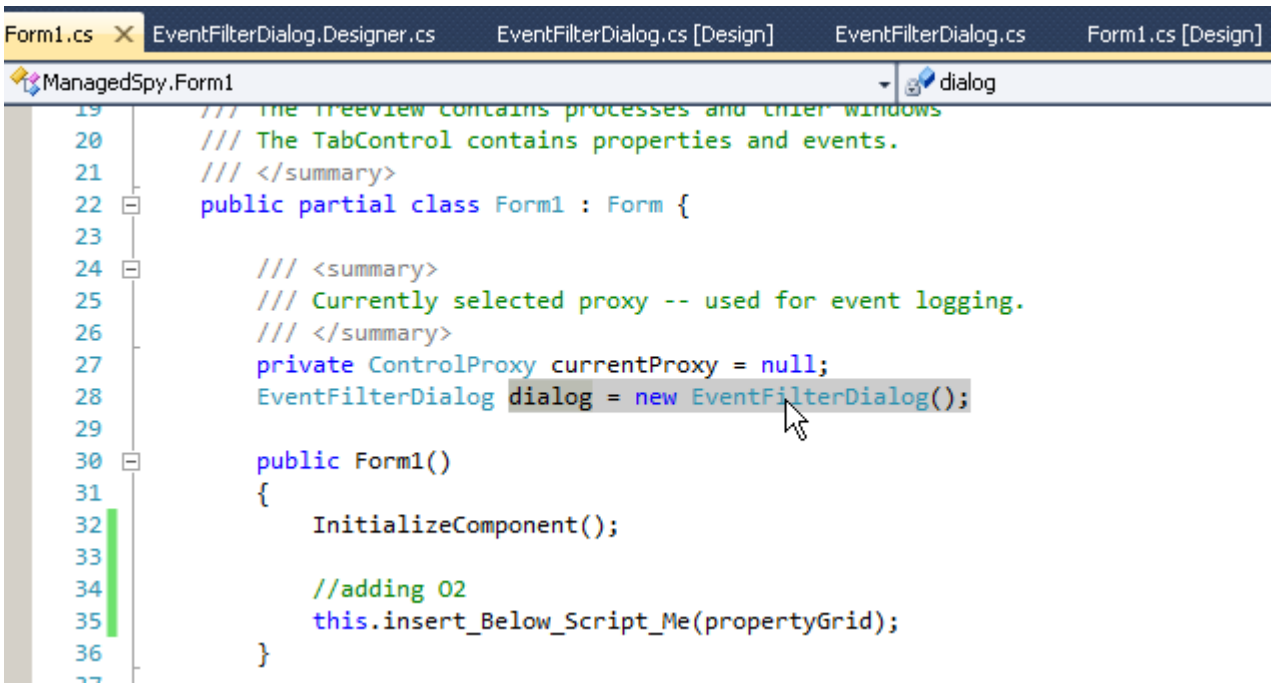
```
Action<bool> setDisplayForAllEvents =
       (value)=>{
                     dataGridView.Rows.forEach(
                            (DataGridViewRow row)=> row.Cells[1].Value = value);
                 };
setDisplayForAllEvents(false);
"Now Select them all again".alert();
setDisplayForAllEvents(true);
```

it is probably more usefull to have this Select/DeSelect capability as a context menu:

```
Action<bool> setDisplayForAllEvents =
       (value)=>{
                            dataGridView.Rows.forEach(
                                   (DataGridViewRow row)=> row.Cells[1].Value = value);
                     };
dataGridView.add_ContextMenu()
                    .add_MenuItem("DeSelect All", true,  ()=>setDisplayForAllEvents(false))
                    .add_MenuItem("Select All", ()=>setDisplayForAllEvents(true));
```



In order to make the changes in these checkboxes to be taken into account by the main ManagedSpy.exe form we need to set the value of the private *dialog* field from *Form1.cs*



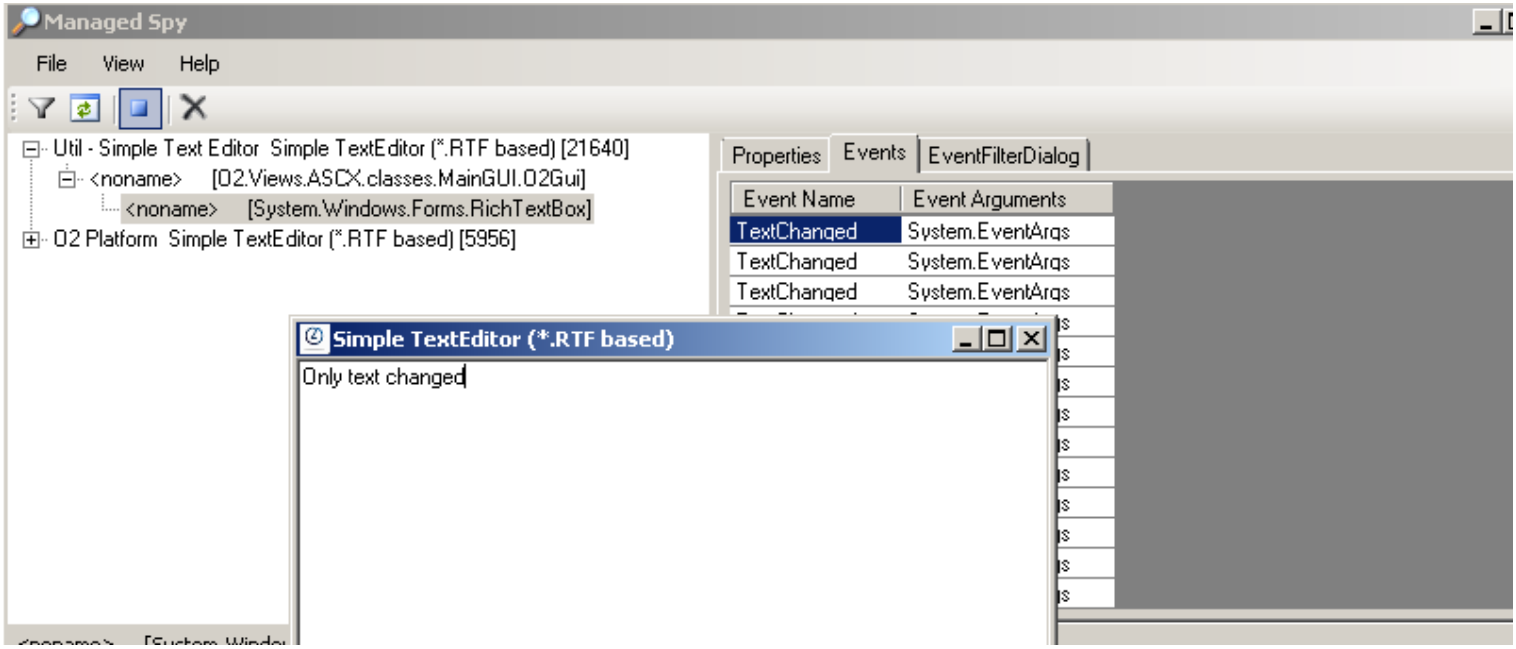
Which can be done like this:

```
var eventFilterDialog = tabControl.newInThread<EventFilterDialog>();
```

```
propertyGrid.parentForm().field("dialog", eventFilterDialog);
var dataGridView = eventFilterDialog.control<DataGridView>()
                                    .columnWidth(0,-1).columnWidth(1,50);
```

With this change we can see only the events we want (as selected from the 'EvenFilterDialog' tab)

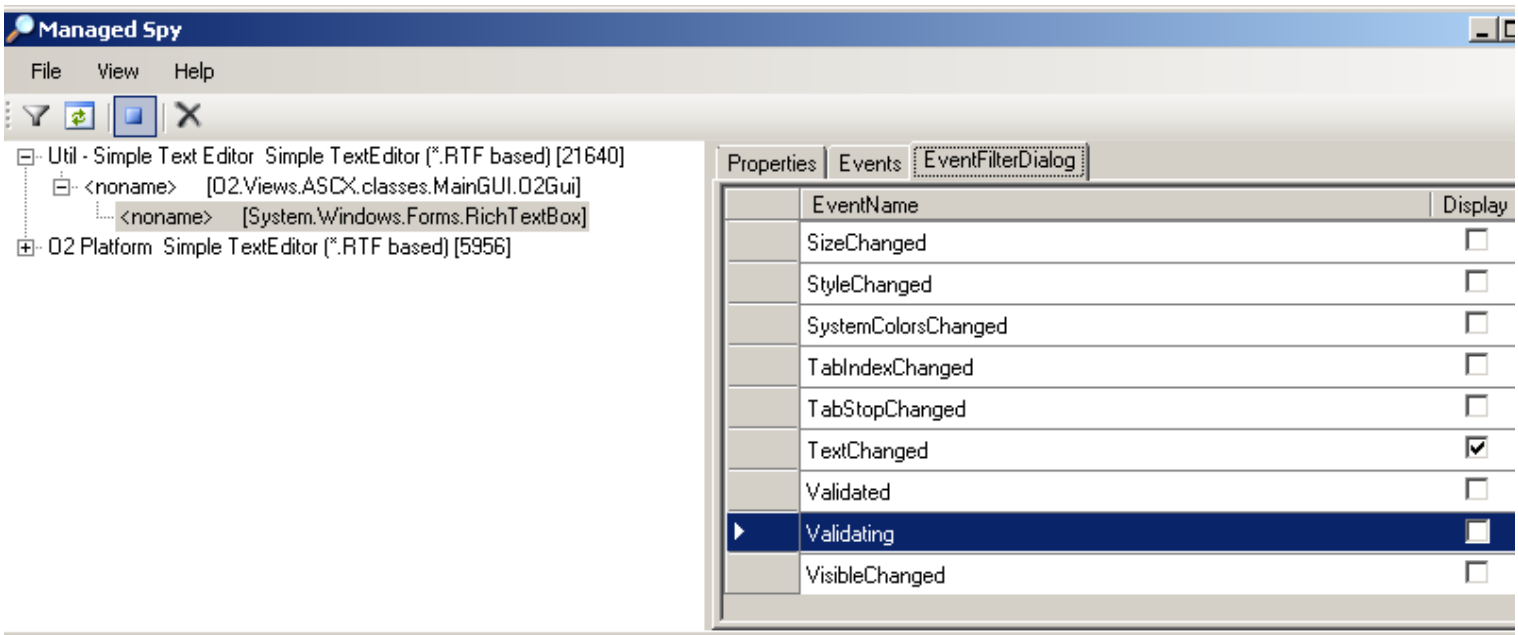
(at the moment we need to press the play and stop buttons to enable the filters from being activate)



We can detect changes in the dataGridView like this:

```
dataGridView.onClick(
       (row,cell)=>{
                        "User clicked on row: {0} cell {1}".info(row,cell);
                   });
```

.



And invoke the private methods ***StopLogging*** and ***StartLogging*** to set the rules

```
dataGridView.onClick(
       (row,cell)=>{
                                    if (cell ==1)
                                    {
                                           var form1 = (Form1)propertyGrid.parentForm();

                                           form1.invoke("StopLogging");
                                           form1.invoke("StartLogging");

                                    }
                             });
```

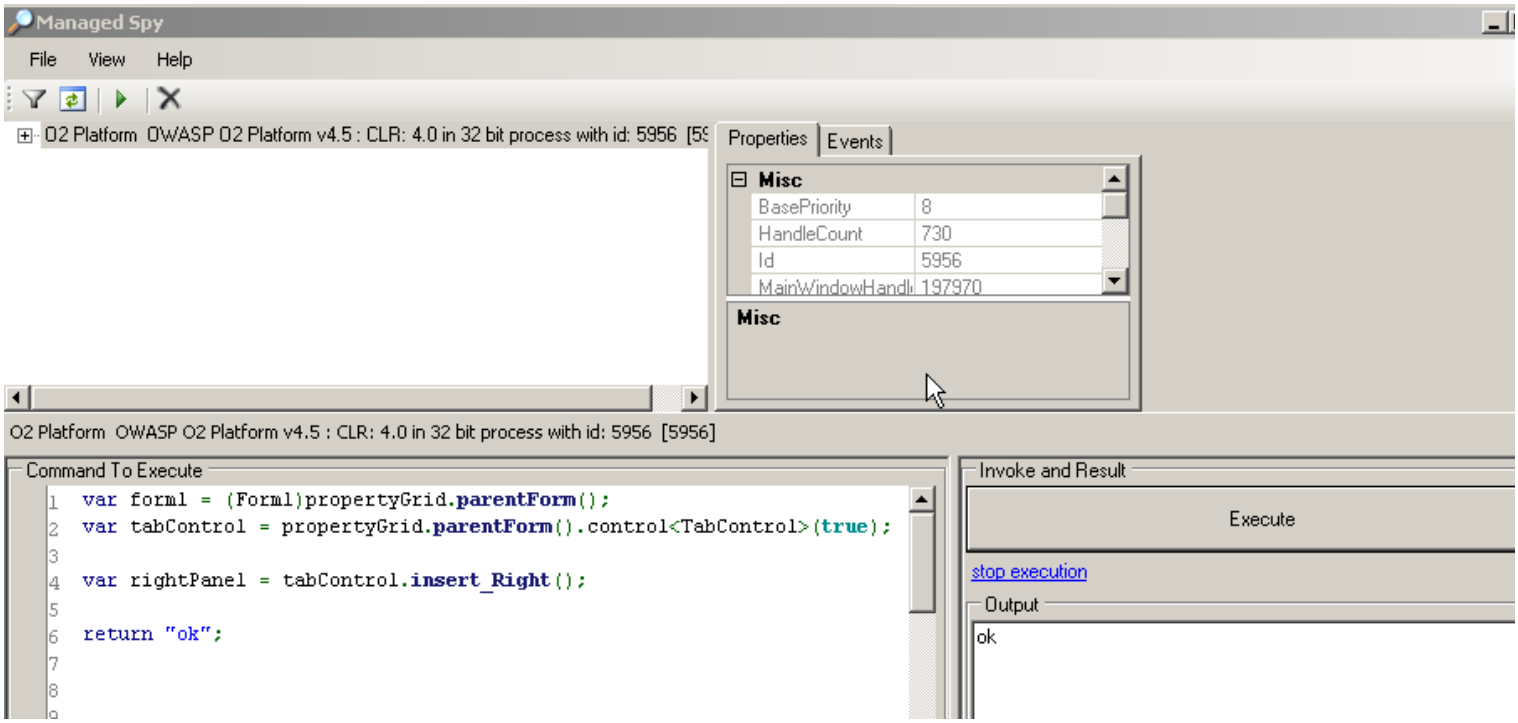
Note: it is better to use this to reuse the `eventFilterDialog` created by Form1

```
var eventFilterDialog = (EventFilterDialog)propertyGrid.parentForm().field("dialog");

var dataGridView = eventFilterDialog.control<DataGridView>()
                                                                  .columnWidth(0,-1).columnWidth(1,50);
```

After using the *EventFilterDialog* on a *TabPage* and needed to check on the 'Events' tab to see the Events, it feels that the *EventFilterDialog* should be always visible.

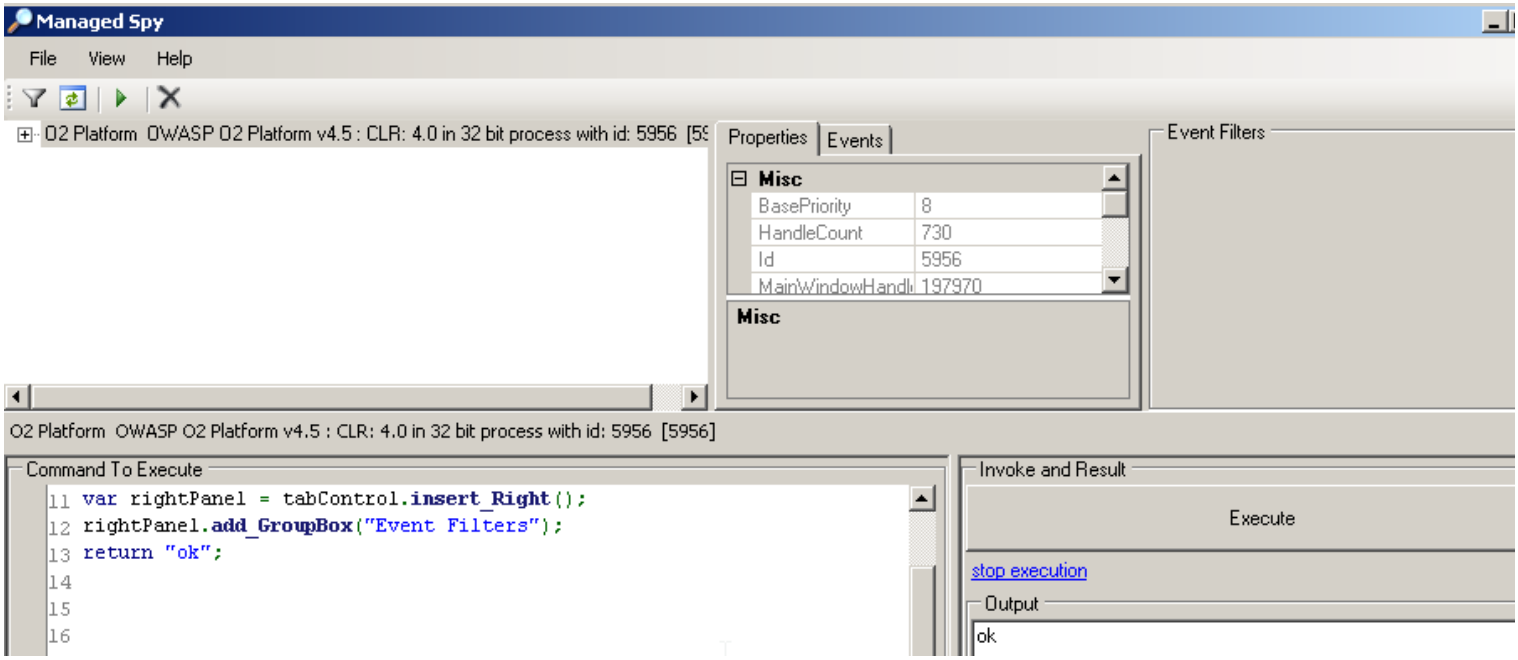
Lets try putting it on the right of the TabControl

```
var form1 = (Form1)propertyGrid.parentForm();
var tabControl = propertyGrid.parentForm().control<TabControl>(true);

var rightPanel = tabControl.insert_Right();

return "ok";
```



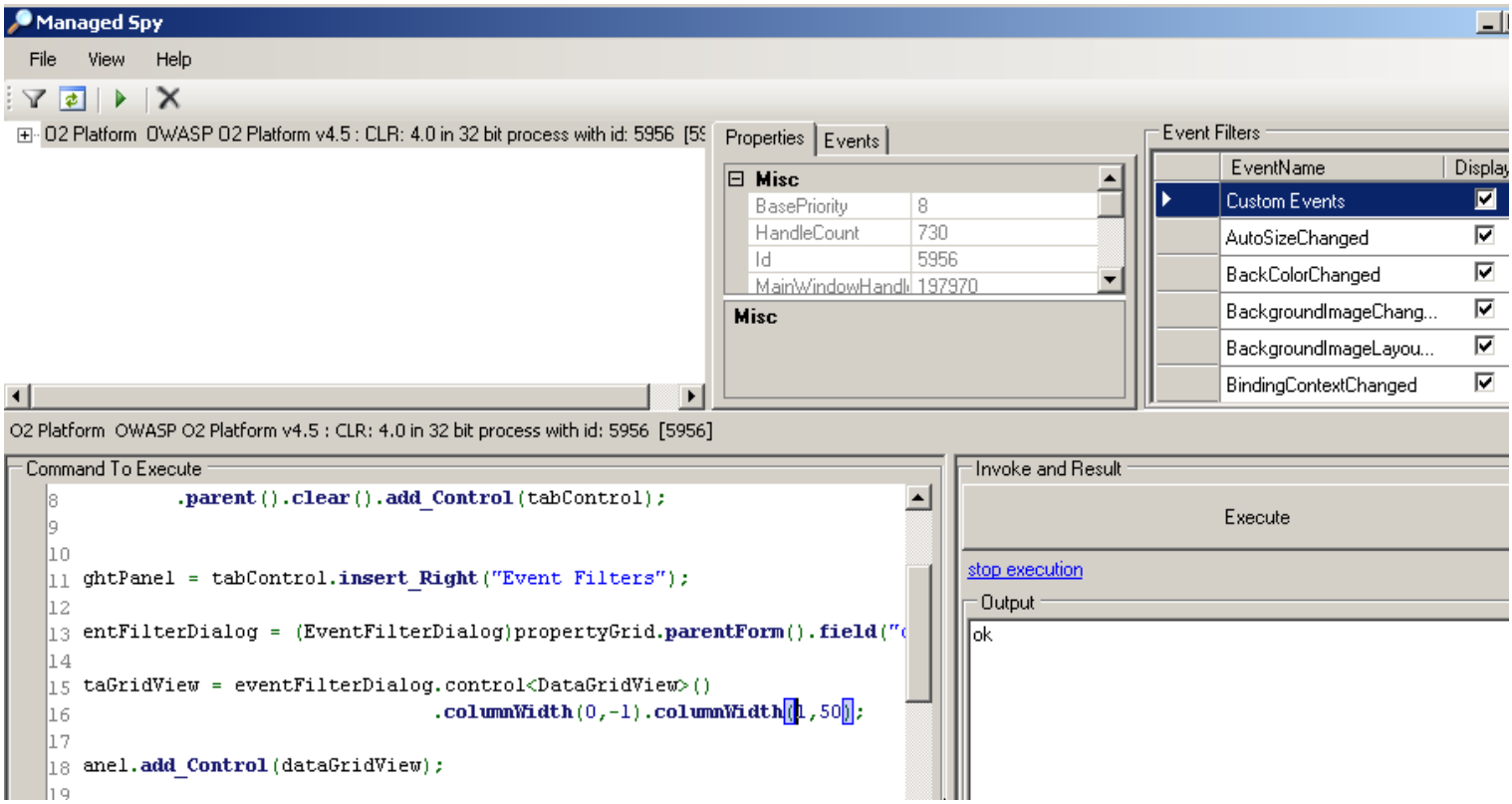
adding a title

```
var rightPanel = tabControl.insert_Right();
rightPanel.add_GroupBox("Event Filters");
```



Adding the DataGridView to the GroupBox (on the right)

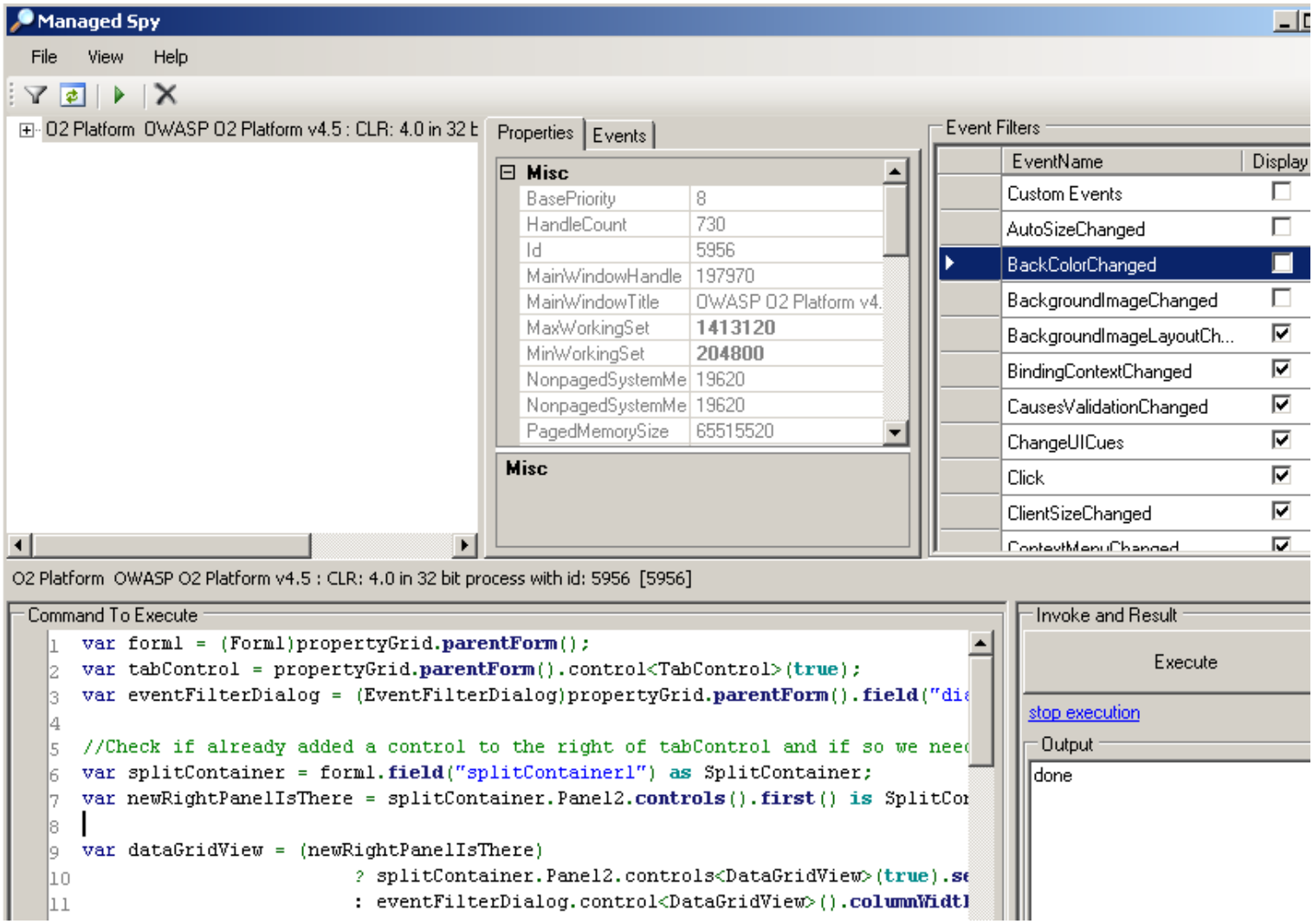
```
var rightPanel = tabControl.insert_Right("Event Filters");
var eventFilterDialog = (EventFilterDialog)propertyGrid.parentForm().field("dialog");
var dataGridView = eventFilterDialog.control<DataGridView>()
                                    .columnWidth(0,-1).columnWidth(1,50);
rightPanel.add_Control(dataGridView);
```



Here is a refectored version that correctly percists the DataGridView values on each execution

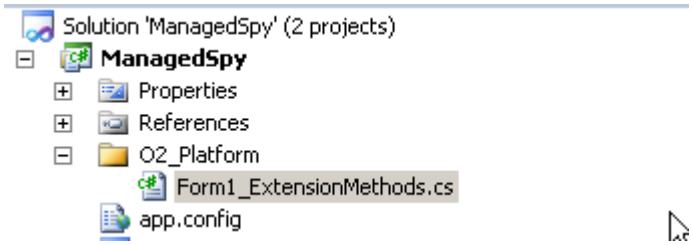
```
var form1 = (Form1)propertyGrid.parentForm();
var tabControl = propertyGrid.parentForm().control<TabControl>(true);
```

```
var eventFilterDialog = (EventFilterDialog)propertyGrid.parentForm().field("dialog");

//Check if already added a control to the right of tabControl and if so we need to  remove it
var splitContainer = form1.field("splitContainer1") as SplitContainer;
var newRightPanelIsThere = splitContainer.Panel2.controls().first() is SplitContainer;

var dataGridView = (newRightPanelIsThere)
                                        ? splitContainer.Panel2.controls<DataGridView>(true).second()
                                        : eventFilterDialog.control<DataGridView>().columnWidth(0,-
1).columnWidth(1,50);

if (newRightPanelIsThere)
       tabControl.splitContainer().parent().clear().add_Control(tabControl);


var rightPanel = tabControl.insert_Right("Event Filters");

rightPanel.add_Control(dataGridView);
```



Before we go much futher here, lets start adding some of the new features to the main ManagedLib.dll

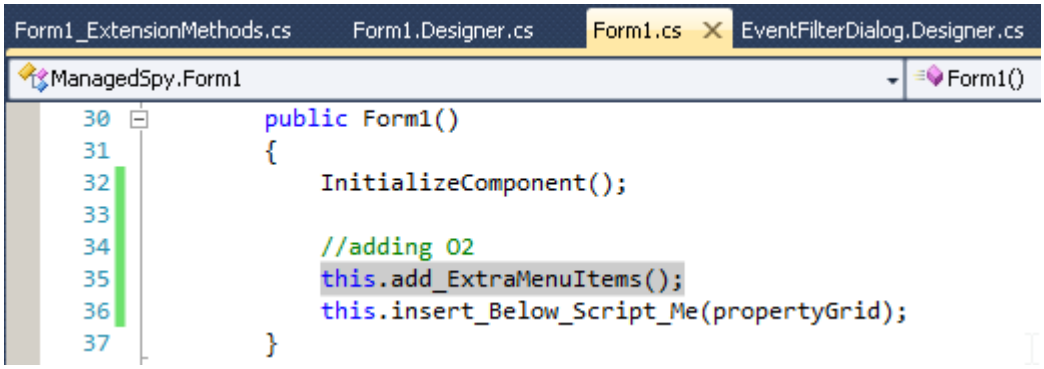
add a new folder and file:



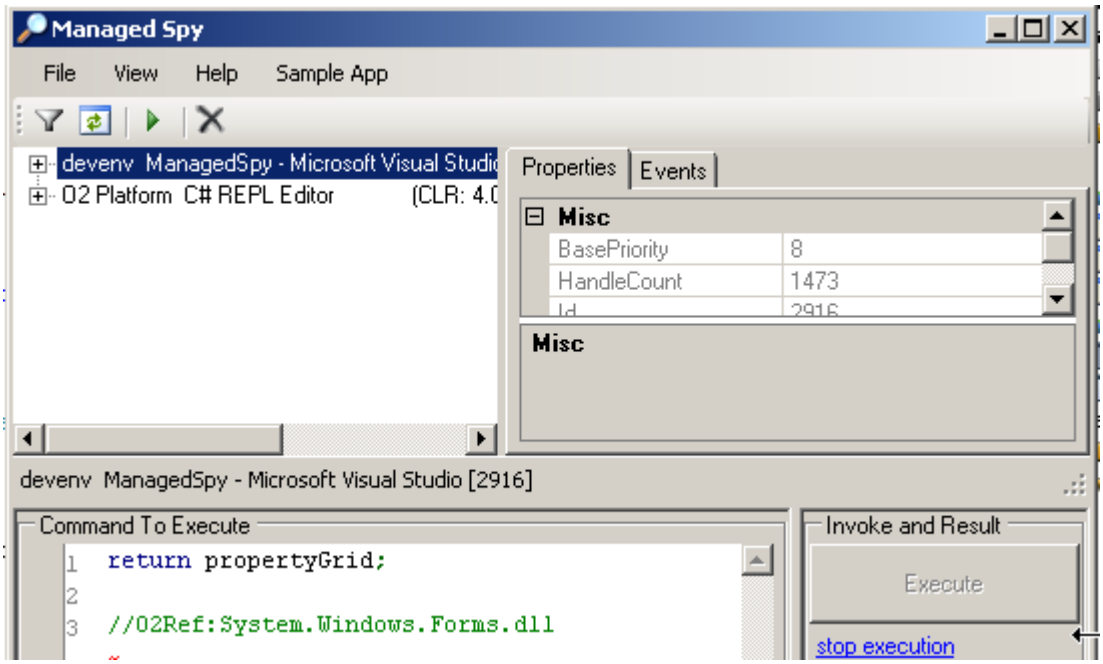
With this extension method

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using O2.DotNetWrappers.ExtensionMethods;

namespace ManagedSpy
{
    public static class Form1_ExtensionMethods
    {
        public static string TestFile1 { get; set; }

        static Form1_ExtensionMethods()
        {
            TestFile1 = @"C:/_WorkDir/O2/O2 Install/_TempDir_v4.5.1.0/11_17_2012/Util - Simple Text
Editor [18704]\Util - Simple Text Editor.exe";
        }
        public static Form1 add_ExtraMenuItems(this Form1 form1)
        {

            form1.control<MenuStrip>(true)
                     .add_MenuItem("Sample App")
                              .add_MenuItem("Open Simple TextEditor", ()=> TestFile1.startProcess());
            return form1;
        }
    }
}
```

Which is invoked on the Form1 ctor:



and puts the new control by default in the top menu (on new instances on ManagedSpy.exe)


Managed Spy
File
View
Help
Sample App
devenv ManagedSpy - Microsoft Visual Studio
O2 Platform C# REPL Editor
(CLR: 4.0
Properties
Events
Misc
BasePriority
8
HandleCount
1473
Misc
devenv ManagedSpy - Microsoft Visual Studio [2916]
Command To Execute
1 return propertyGrid;
2
3 //O2Ref:System.Windows.Forms.dll
Invoke and Result
Execute
stop execution