Using APL+Win with Events from VisualAPL

Summary

This topic illustrates how to use APL+Win from VIsualAPL when the APL+Win ActiveX server “Notify” or “SysNotify” events will be involved in the solution. These APL+Win events are useful when it is necessary to provide the calling environment, in this case VisualAPL, with real-time information about an APL+Win processing request made by the calling environment.

For example, real-time information transmitted by APL+Win using the Notify event to VIsualAPL might be;

- Error messages due to APL+Win validation of input argument information provided by the calling environment
- Error messages due to unanticipated errors which occur when APL+Win is satisfying the processing request made by the calling environment
- Processing progress information made available to the calling environment while APL+Win is satisfying the processing request made by the calling environment.

APL+Win Notify and SysNotify Event Handler Method Signatures

The APL+Win event handler syntax for the “Notify” and “SysNotify’ events was developed before .Net existed. The APL+Win event handler syntax for these events is of the form:

```
Event_handler_name(ref object Code, ref object Data, out object Result)
```

The .Net “standard” event handler syntax is:

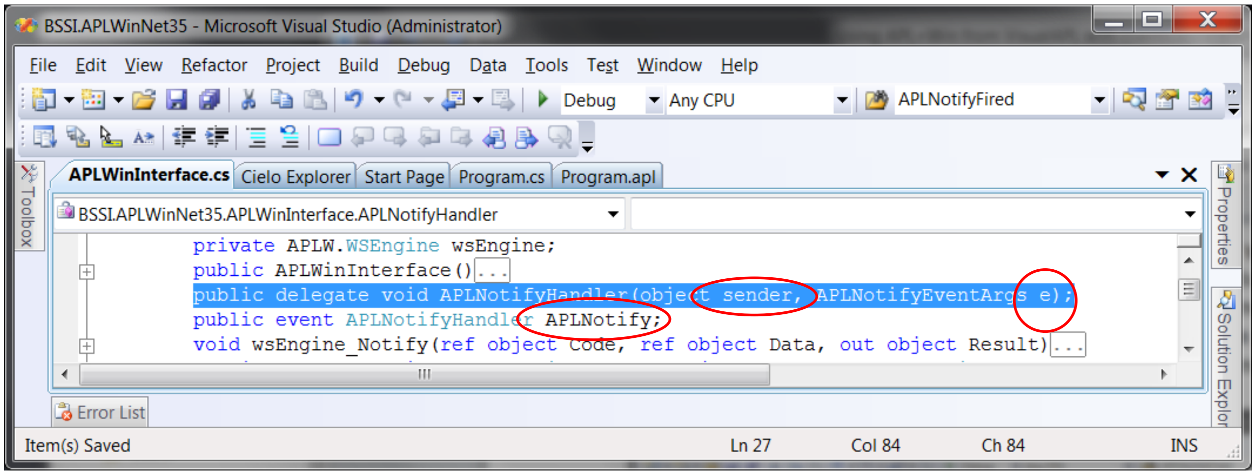
```
Event_handler_Name(object sender, …EventArgs e)
```

Generally C# can properly identify and utilize the ‘native’ APL+Win event handler syntax, but VisualAPL for Visual Studio 2008 cannot because Microsoft documentation of the mechanism to support the ref and out prefixes in the arguments of an event handler method were not available to non-Microsoft programming language developers such as the VisualAPL team.

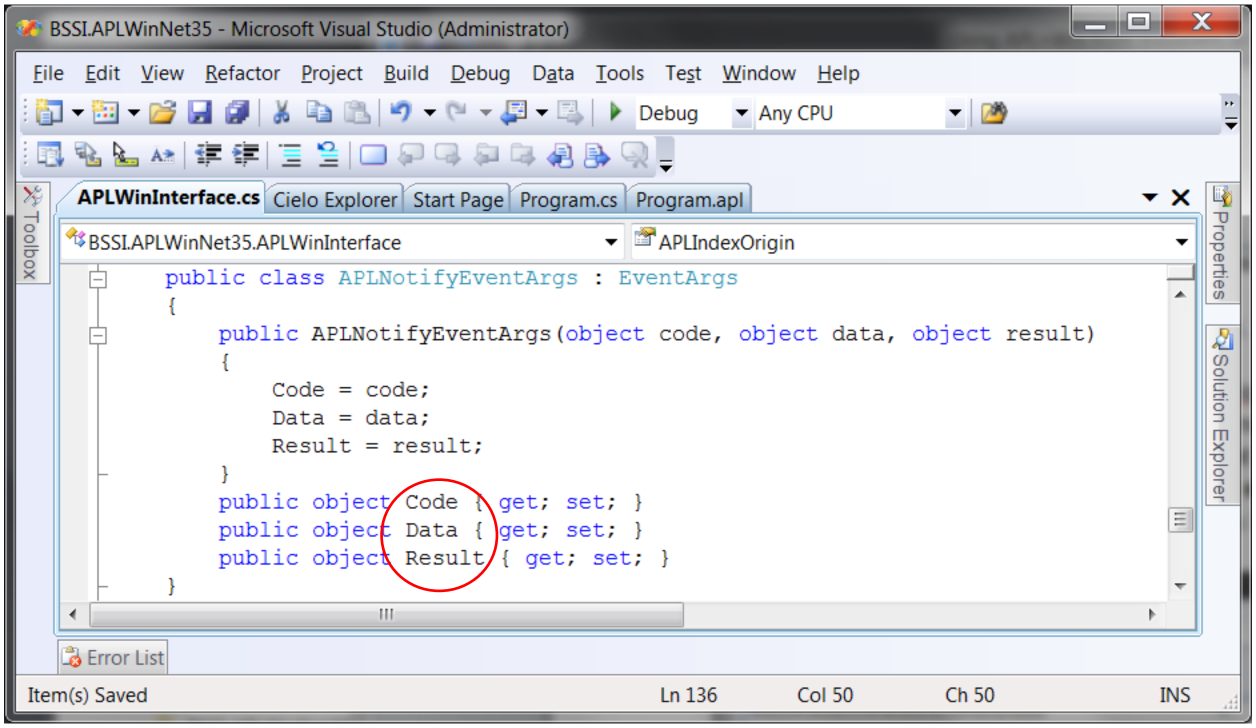
C# Helper .Net Assembly
Rather than modify APL+Win which would have to potential to 'break' existing Win32 application systems developed using APL+Win, .Net provides a convenient and powerful alternative.

This topic introduces a C# 'helper' .Net assembly, "BSSI.APLWinNet35" which will:

❖ Provide new events, APLNotify and APLSysNotify, with .Net 'standard' event handler method signatures and with the same functionality as the native APL+Win ActiveX server events, Notify and SysNotify. The 'helper' .Net assembly defines the "APLNotifyEventArgs" and "APLSysNotifyEventArgs" classes which encapsulate the APL+Win ActiveX server 'native' Notify and SysNotify event arguments into a .Net 'standard' event arguments data type.

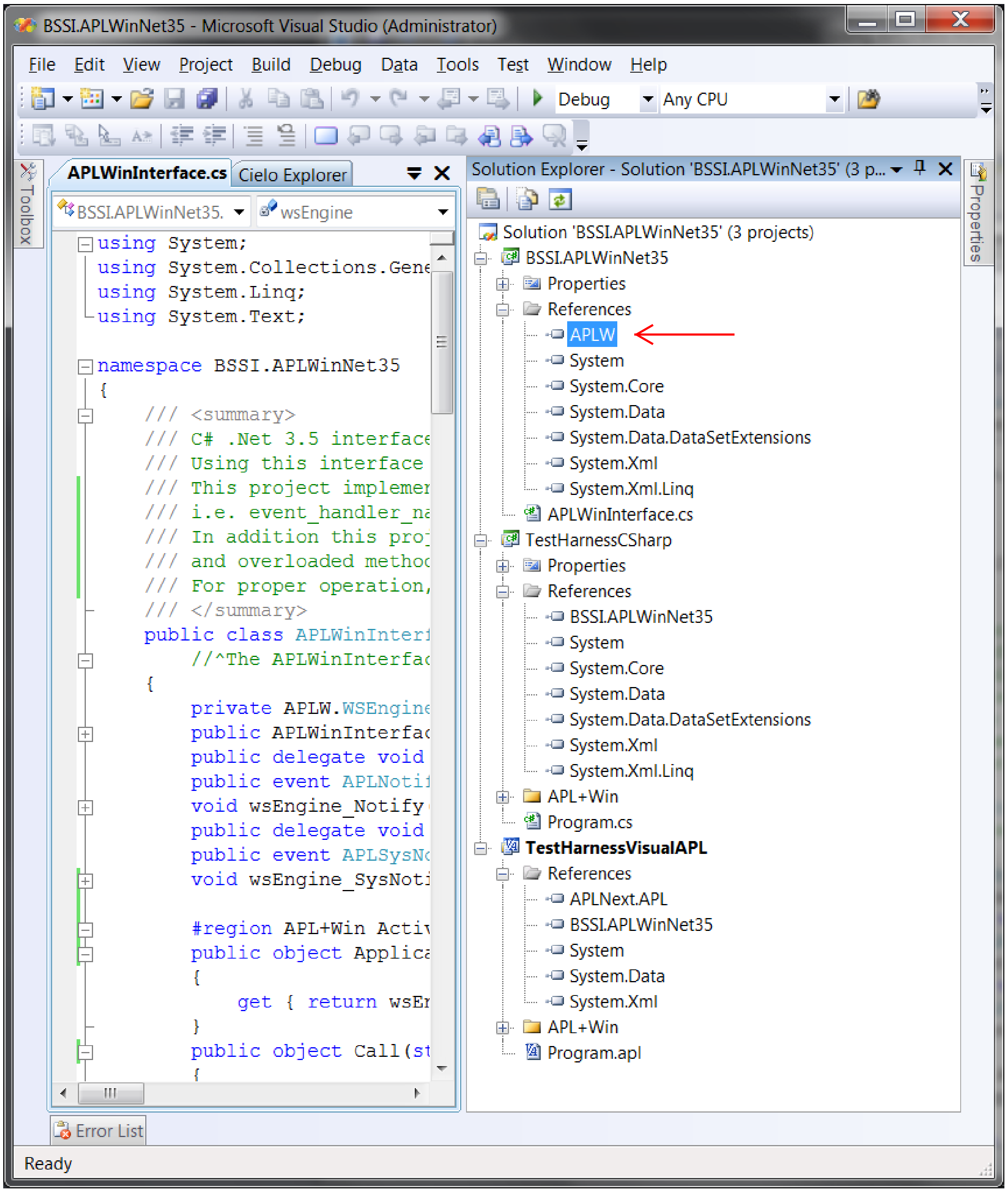

BSSI.APLWinNet35 - Microsoft Visual Studio (Administrator)
File Edit View Refactor Project Build Debug Data Tools Test Window Help
Debug
Any CPU
APLNotifyFired
APLWinInterface.cs
Cielo Explorer
Start Page
Program.cs
Program.apl
BSSI.APLWinNet35.APLWinInterface.APLNotifyHandler
Toolbox
Properties
Solution Explorer
Error List
Item(s) Saved
Ln 27
Col 84
Ch 84
INS

```
        private APLW.WSEngine wsEngine;
        public APLWinInterface()...
        public delegate void APLNotifyHandler(object sender, APLNotifyEventArgs e);
        public event APLNotifyHandler APLNotify;
        void wsEngine_Notify(ref object Code, ref object Data, out object Result)...
```



```
    public class APLNotifyEventArgs : EventArgs
    {
        public APLNotifyEventArgs(object code, object data, object result)
        {
            Code = code;
            Data = data;
            Result = result;
        }
        public object Code { get; set; }
        public object Data { get; set; }
        public object Result { get; set; }
    }
```

❖ Fully encapsulate the use of the APL+Win ActiveX server APLW.dll COM component, so that the calling environment projects will not have to reference the APLW.dll COM component. This feature can benefit any .Net calling environment based on C#, VB.Net or VisualAPL.



BSSI.APLWinNet35 - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Data Tools Test Window Help
Debug
Any CPU
Toolbox
APLWinInterface.cs
Cielo Explorer
BSSI.APLWinNet35.
wsEngine
Solution Explorer - Solution 'BSSI.APLWinNet35' (3 p...
Properties
Error List
Ready

```
using System;
using System.Collections.Gene
using System.Linq;
using System.Text;

namespace BSSI.APLWinNet35
{
    /// <summary>
    /// C# .Net 3.5 interface
    /// Using this interface
    /// This project implemen
    /// i.e. event_handler_na
    /// In addition this proj
    /// and overloaded method
    /// For proper operation,
    /// </summary>
    public class APLWinInterf
        //^The APLWinInterfac
    {
        private APLW.WSEngine
        public APLWinInterfac
        public delegate void
        public event APLNotif
        void wsEngine_Notify
        public delegate void
        public event APLSysNo
        void wsEngine_SysNoti

        #region APL+Win Activ
        public object Applica
        {
            get { return wsEn
        }
        public object Call(st
        {
```

```
Solution 'BSSI.APLWinNet35' (3 projects)
  BSSI.APLWinNet35
    Properties
    References
      APLW
      System
      System.Core
      System.Data
      System.Data.DataSetExtensions
      System.Xml
      System.Xml.Linq
    APLWinInterface.cs
  TestHarnessCSharp
    Properties
    References
      BSSI.APLWinNet35
      System
      System.Core
      System.Data
      System.Data.DataSetExtensions
      System.Xml
      System.Xml.Linq
    APL+Win
    Program.cs
  TestHarnessVisualAPL
    References
      APLNext.APL
      BSSI.APLWinNet35
      System
      System.Data
      System.Xml
    APL+Win
    Program.apl
```

- Provide an option to augment the APL+Win ActiveX server with additional methods, properties and events for the convenience of the calling environment. For example the ‘helper’ .Net assembly can include methods or properties which have ‘friendly’ names, perform common operations or add validation.

  - The APLWorkspaceLoad method performs a common operation and adds validation:

```
public void APLWorkspaceLoad(string wsPath)
{
    if (System.IO.File.Exists(wsPath))
    {
       wsEngine.SysCommand("LOAD " + wsPath.Substring(0,wsPath.Length-3));
       //^Remove the ".w3" suffix
    }
    else
    {
       throw new Exception("APL+Win workspace not accessible: "+wsPath);
    }
      }
```

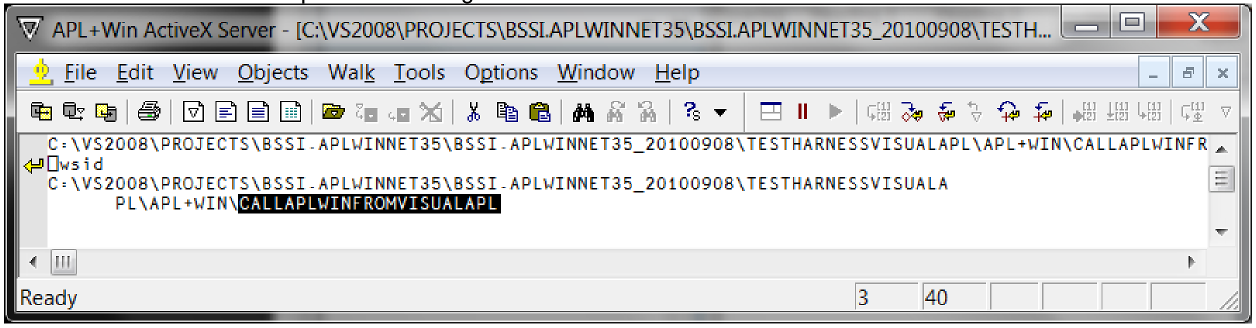
  - The APLInextOrigin public property simplifies the calling environment’s access of this APL+Win system variable:

```
     public Int32 APLIndexOrigin
{
                 get { return (Int32) wsEngine.get_SysVariable("IO"); }
     set { wsEngine.set_SysVariable("IO", value); }
}
```

VisualAPL Environment Calling APL+Win 'Helper' .Net Assembly:

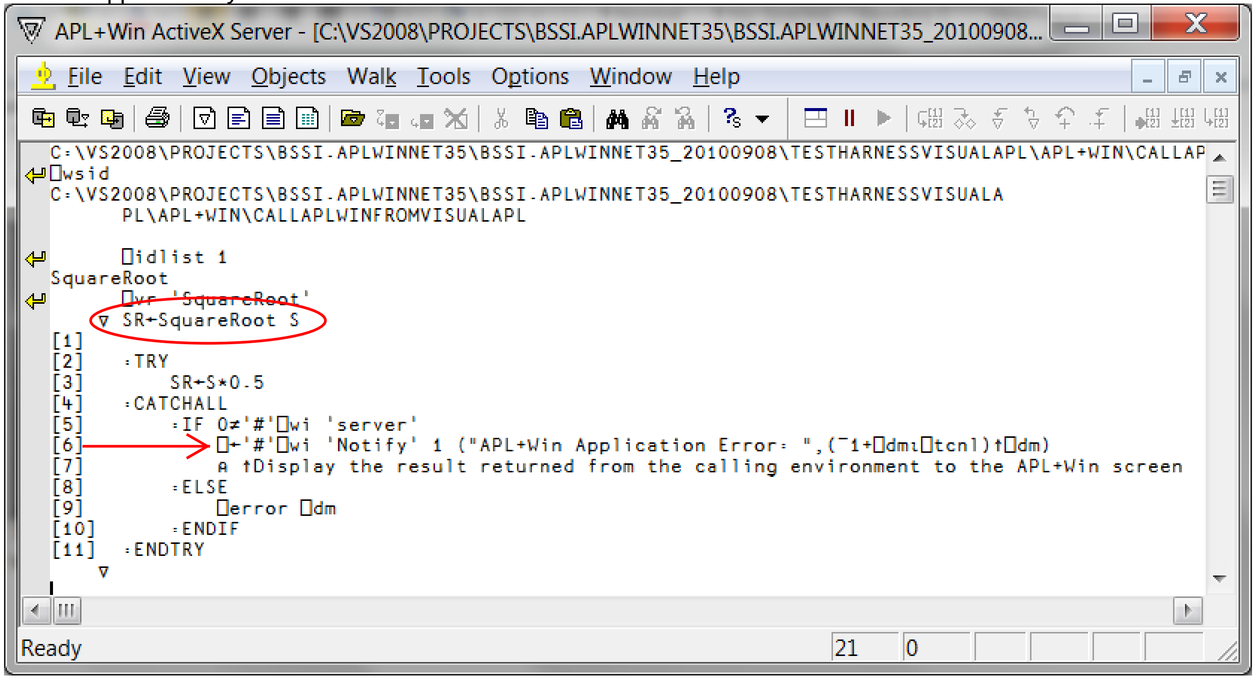
The 'business rules' of this application system are supported by APL+Win as an ActiveX server. This means that:

- ❖ APL+Win is registered as an ActiveX server (COM component) on the target machine
- ❖ There is an APL+Win workspace on the target machine


APL+Win ActiveX Server - [C:\VS2008\PROJECTS\BSSI.APLWINNET35\BSSI.APLWINNET35_20100908\TESTH...
File Edit View Objects Walk Tools Options Window Help
Ready
3
40

```
C:\VS2008\PROJECTS\BSSI.APLWINNET35\BSSI.APLWINNET35_20100908\TESTHARNESSVISUALAPL\APL+WIN\CALLAPLWINFR
⎕wsid
C:\VS2008\PROJECTS\BSSI.APLWINNET35\BSSI.APLWINNET35_20100908\TESTHARNESSVISUALA
      PL\APL+WIN\CALLAPLWINFROMVISUALAPL
```

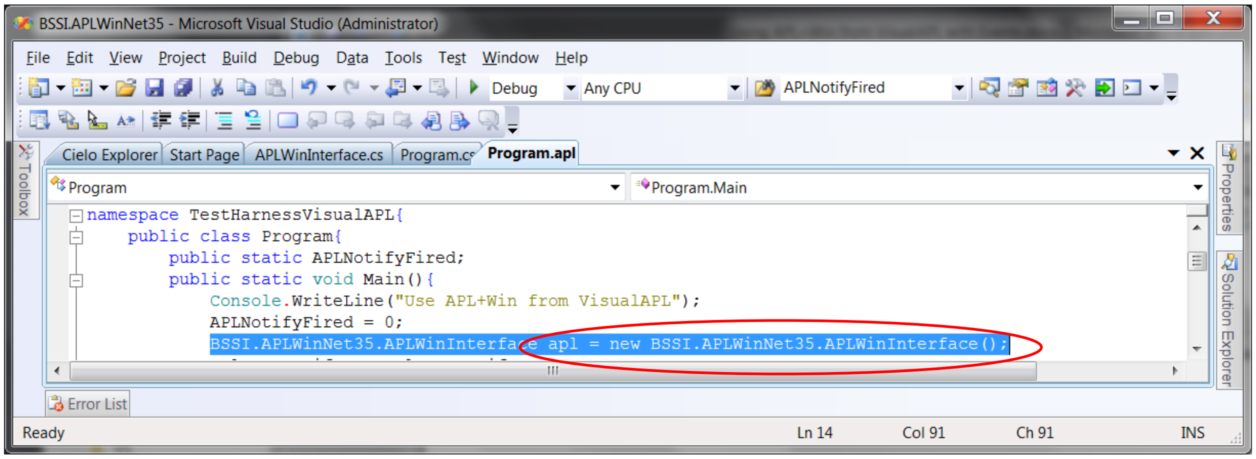
- ❖ That APL+Win workspace contains APL+Win functions which accept arguments and return results associated with the application system's business rules.


APL+Win ActiveX Server - [C:\VS2008\PROJECTS\BSSI.APLWINNET35\BSSI.APLWINNET35_20100908...
File Edit View Objects Walk Tools Options Window Help
Ready
21
0

```
C:\VS2008\PROJECTS\BSSI.APLWINNET35\BSSI.APLWINNET35_20100908\TESTHARNESSVISUALAPL\APL+WIN\CALLAP
⎕wsid
C:\VS2008\PROJECTS\BSSI.APLWINNET35\BSSI.APLWINNET35_20100908\TESTHARNESSVISUALA
      PL\APL+WIN\CALLAPLWINFROMVISUALAPL

      ⎕idlist 1
SquareRoot
      ⎕vr 'SquareRoot'
    ∇ SR←SquareRoot S
[1]
[2]   :TRY
[3]       SR←S*0.5
[4]   :CATCHALL
[5]       :IF 0≠'#'⎕wi 'server'
[6]           ⎕←'#'⎕wi 'Notify' 1 ("APL+Win Application Error: ",(¯1+⎕dm⍳⎕tcnl)↑⎕dm)
[7]           ⍝ ↑Display the result returned from the calling environment to the APL+Win screen
[8]       :ELSE
[9]           ⎕error ⎕dm
[10]      :ENDIF
[11]  :ENDTRY
    ∇
```

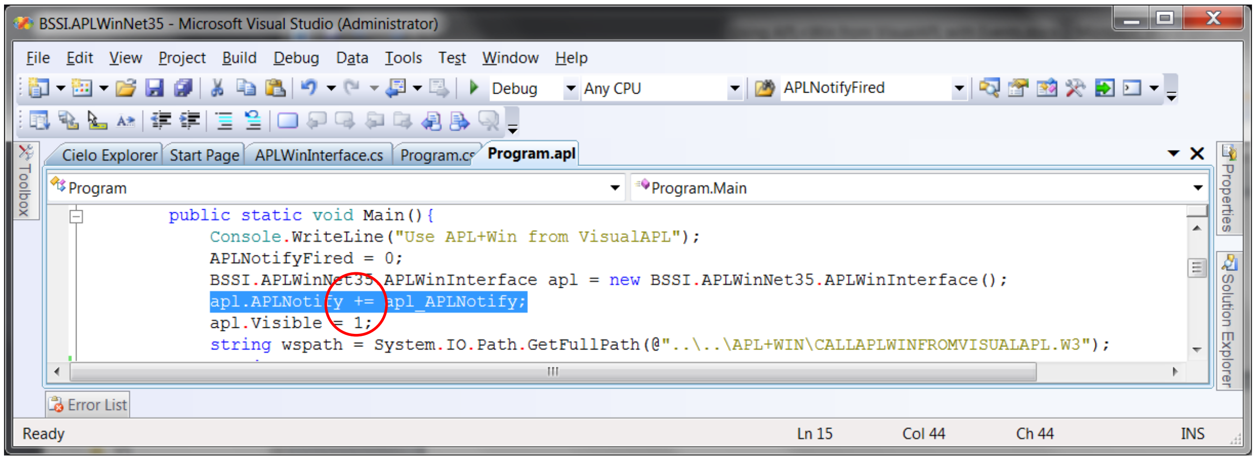
Notice how APL+Win reports errors to the calling environment by using the Notify event.

The VisualAPL calling environment creates an instance of APL+Win as an ActiveX server indirectly using the C# ‘Helper’ .Net assembly:

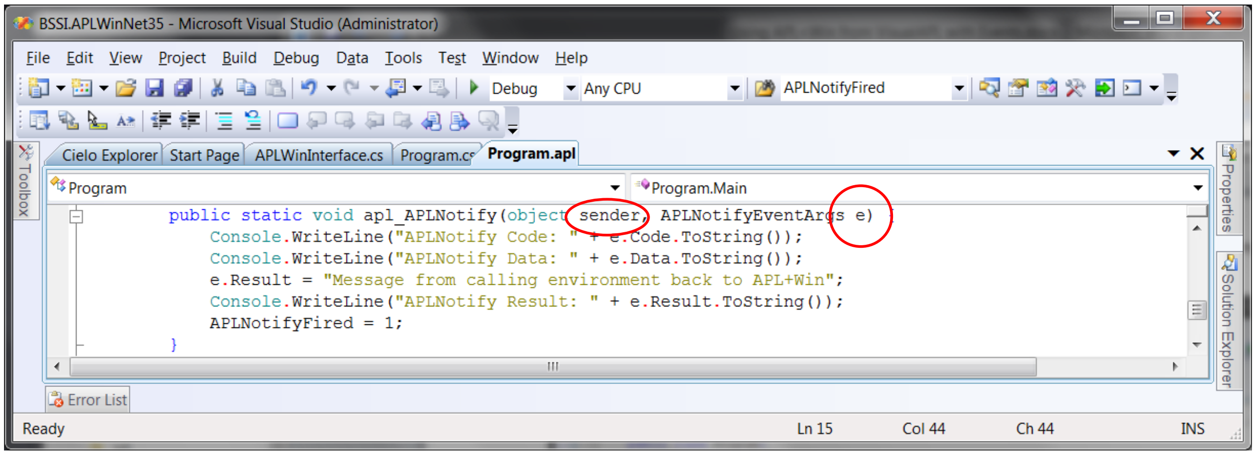


```
namespace TestHarnessVisualAPL{
    public class Program{
        public static APLNotifyFired;
        public static void Main(){
            Console.WriteLine("Use APL+Win from VisualAPL");
            APLNotifyFired = 0;
            BSSI.APLWinNet35.APLWinInterface apl = new BSSI.APLWinNet35.APLWinInterface();
```

The VisualAPL calling environment subscribes to the APL+Win ActiveX server Notify event indirectly by subscribing to the C# “Helper” .Net assembly ‘APLNotify’ event. The ‘APLNotify’ event handler syntax uses .Net ‘standard’ syntax which can be consumed by VisualAPL:

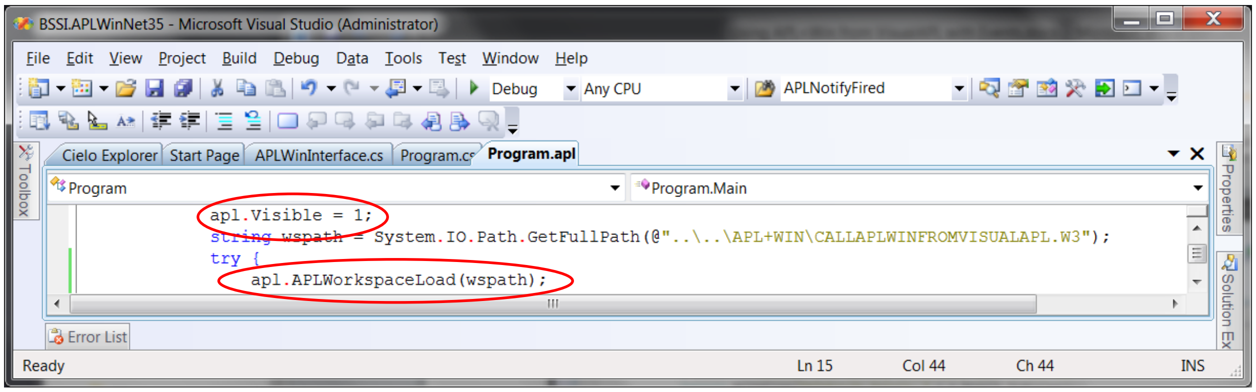


```
        public static void Main(){
            Console.WriteLine("Use APL+Win from VisualAPL");
            APLNotifyFired = 0;
            BSSI.APLWinNet35.APLWinInterface apl = new BSSI.APLWinNet35.APLWinInterface();
            apl.APLNotify += apl_APLNotify;
            apl.Visible = 1;
            string wspath = System.IO.Path.GetFullPath(@"..\..\APL+WIN\CALLAPLWINFROMVISUALAPL.W3");
```



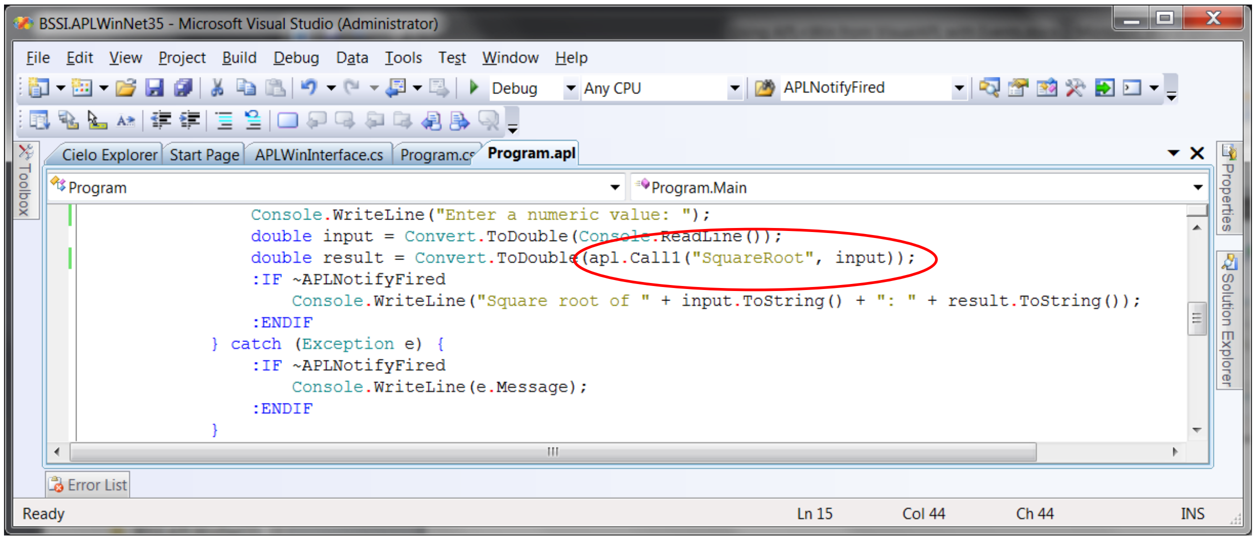
```
        public static void apl_APLNotify(object sender, APLNotifyEventArgs e)
            Console.WriteLine("APLNotify Code: " + e.Code.ToString());
            Console.WriteLine("APLNotify Data: " + e.Data.ToString());
            e.Result = "Message from calling environment back to APL+Win";
            Console.WriteLine("APLNotify Result: " + e.Result.ToString());
            APLNotifyFired = 1;
        }
```

The VisualAPL calling environment makes the APL+Win ActiveX server visible by using the Visible public property of the C# “Helper’ .Net assembly and uses the “APLWorkspaceLoad()” method of the C# ‘Helper’ .Net assembly to load the APL+Win application-specific workspace:



```
            apl.Visible = 1;
            string wspath = System.IO.Path.GetFullPath(@"..\..\APL+WIN\CALLAPLWINFROMVISUALAPL.W3");
            try {
                apl.APLWorkspaceLoad(wspath);
```

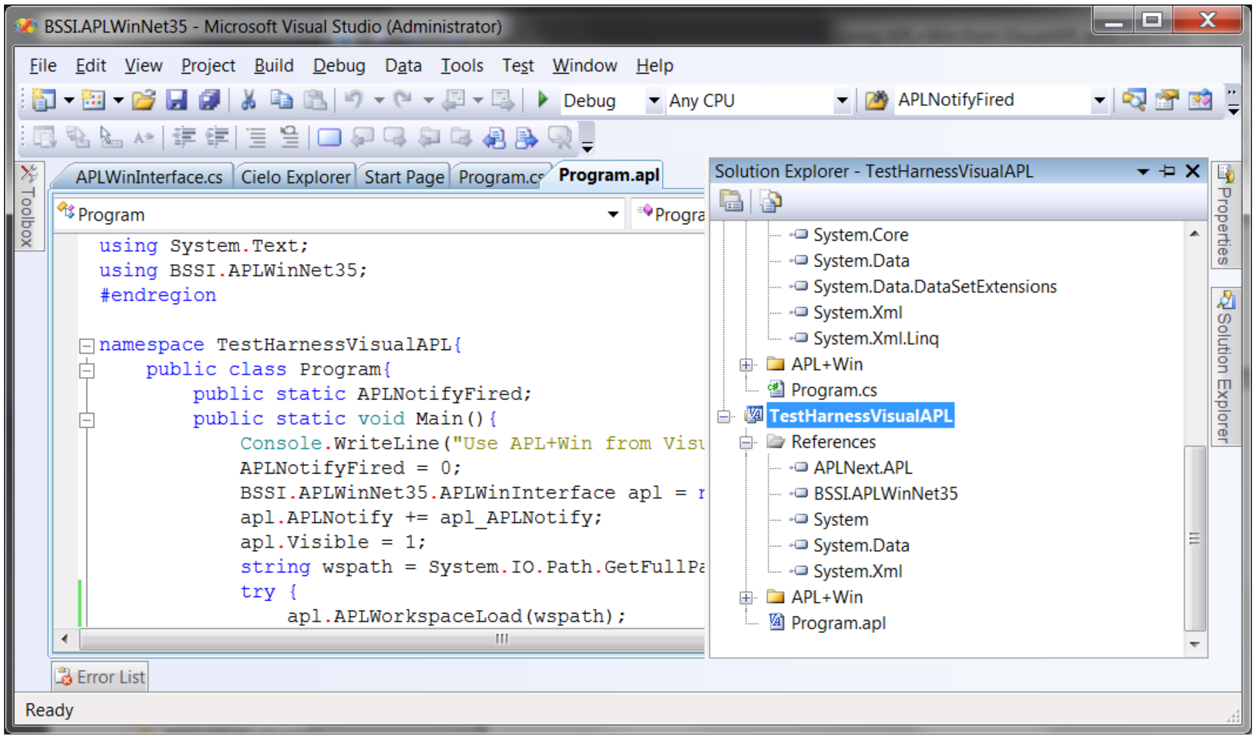
The VisualAPL calling environment then requests user input of an application-specific value, converts it to an appropriate numeric value and makes a request to the APL+Win ActiveX server to process the input and return an appropriate value or error message according to the ‘business rules’ of the application system:



```
                Console.WriteLine("Enter a numeric value: ");
                double input = Convert.ToDouble(Console.ReadLine());
                double result = Convert.ToDouble(apl.Call1("SquareRoot", input));
                :IF ~APLNotifyFired
                    Console.WriteLine("Square root of " + input.ToString() + ": " + result.ToString());
                :ENDIF
            } catch (Exception e) {
                :IF ~APLNotifyFired
                    Console.WriteLine(e.Message);
                :ENDIF
            }
```

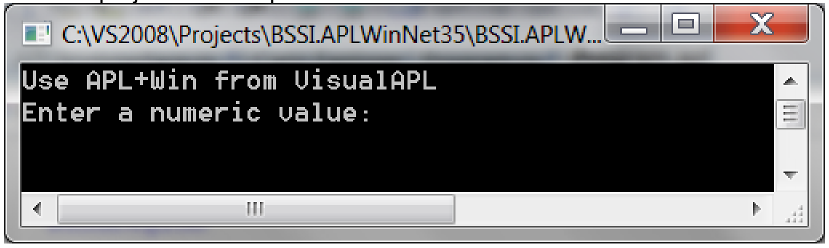
An analogous TestHarnessCSharp project is also included in the Visual Studio solution to illustrate the use of the C# “Helper” .Net assembly with C# as the calling environment. Note that for purposes of this example in order to maintain a parallel C# test harness, strong data typing was used in the VisualAPL test harness, although this is not actually necessary.

Running the VisualAPL Sample Application:
The TestHarnessVisualAPL project should be set as the ‘Start Up Project’ in the Visual Studio Solution Explorer:



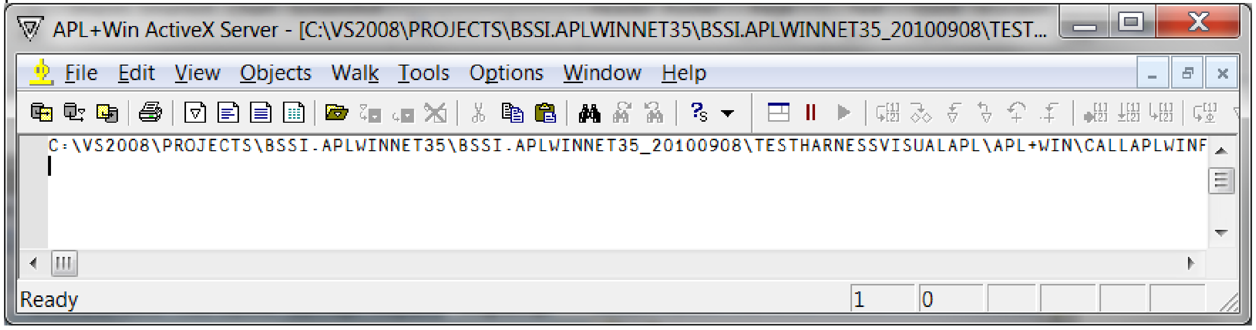
```
using System.Text;
using BSSI.APLWinNet35;
#endregion

namespace TestHarnessVisualAPL{
    public class Program{
        public static APLNotifyFired;
        public static void Main(){
            Console.WriteLine("Use APL+Win from Visu
            APLNotifyFired = 0;
            BSSI.APLWinNet35.APLWinInterface apl = r
            apl.APLNotify += apl_APLNotify;
            apl.Visible = 1;
            string wspath = System.IO.Path.GetFullPa
            try {
                apl.APLWorkspaceLoad(wspath);
```

Use the Visual Studio F5 keystroke to start an instance of the “TestHarnessVisualAPL project which is a simple ‘console project’ that will present the ‘Command Window’ user interface:



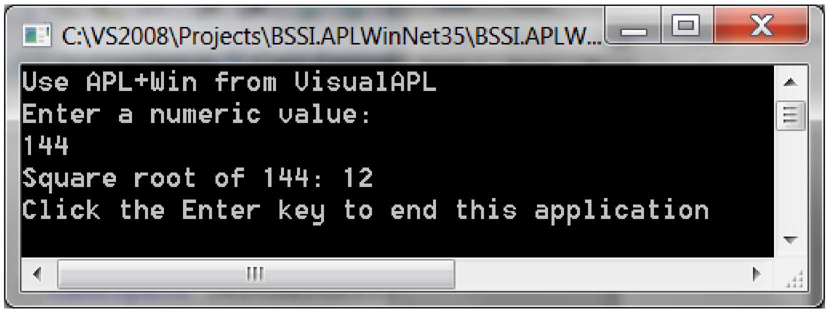
```
Use APL+Win from VisualAPL
Enter a numeric value:
```

Because this project set the Visible property of the APL+Win ActiveX server to 1, an instance of APL+Win will also be presented:


APL+Win ActiveX Server - [C:\VS2008\PROJECTS\BSSI.APLWINNET35\BSSI.APLWINNET35_20100908\TEST...
File Edit View Objects Walk Tools Options Window Help
Ready
1
0

```
C:\VS2008\PROJECTS\BSSI.APLWINNET35\BSSI.APLWINNET35_20100908\TESTHARNESSVISUALAPL\APL+WIN\CALLAPLWINF
```

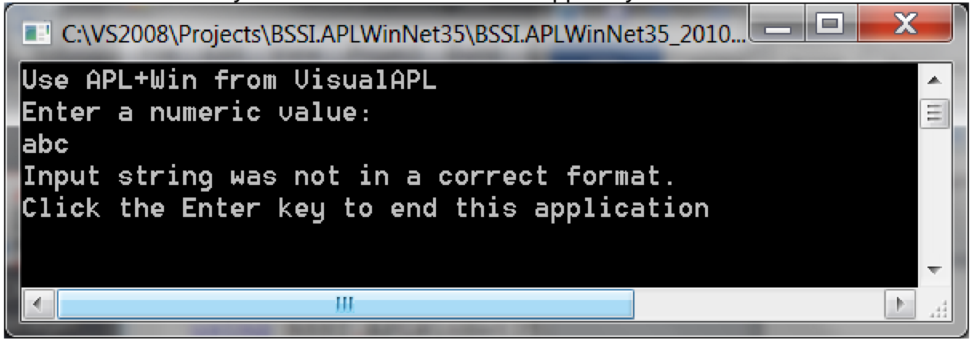
In the “Command Window” enter a valid numeric value, e.g. 144 and click the Enter key to display the result of the APL+Win ‘business rules’ for this application system:



```
C:\VS2008\Projects\BSSI.APLWinNet35\BSSI.APLW...
Use APL+Win from VisualAPL
Enter a numeric value:
144
Square root of 144: 12
Click the Enter key to end this application
```

Next click the Enter key again to close the instance of APL+Win ActiveX server and end the debugging session.

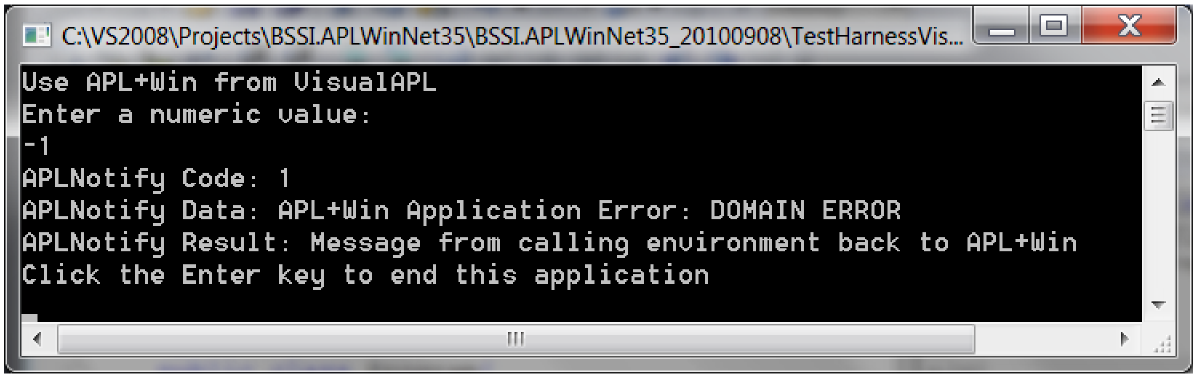
Next follow this debugging procedure again, but this time enter an invalid numeric value into the “Command Window” and click the Enter key to see the error condition trapped by VisualAPL:



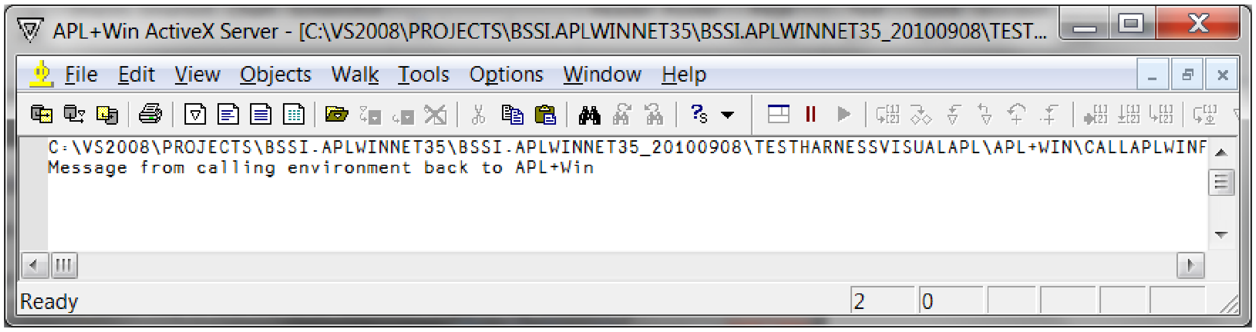
```
C:\VS2008\Projects\BSSI.APLWinNet35\BSSI.APLWinNet35_2010...
Use APL+Win from VisualAPL
Enter a numeric value:
abc
Input string was not in a correct format.
Click the Enter key to end this application
```

Next click the Enter key again to close the instance of APL+Win ActiveX server and end the debugging session.

Next follow this debugging procedure again, bu this time enter a negative numeric value into the ‘Command Window” and click the Enter key to see the error condition trapped by APL+Win ActiveX server and observe the response from the VisualAPL calling environment back to APL+Win. This returned information might be used in a more sophisticiated application system to enable APL+Win to properly recover from an error condition and maintain application system functionality and data integrity.



```
C:\VS2008\Projects\BSSI.APLWinNet35\BSSI.APLWinNet35_20100908\TestHarnessVis...
Use APL+Win from VisualAPL
Enter a numeric value:
-1
APLNotify Code: 1
APLNotify Data: APL+Win Application Error: DOMAIN ERROR
APLNotify Result: Message from calling environment back to APL+Win
Click the Enter key to end this application
```


APL+Win ActiveX Server - [C:\VS2008\PROJECTS\BSSI.APLWINNET35\BSSI.APLWINNET35_20100908\TEST...
File Edit View Objects Walk Tools Options Window Help
Ready
2
0

```
C:\VS2008\PROJECTS\BSSI.APLWINNET35\BSSI.APLWINNET35_20100908\TESTHARNESSVISUALAPL\APL+WIN\CALLAPLWINF
Message from calling environment back to APL+Win
```

Finally click the Enter to close APL+Win and end this debugging session.