

Problem getting events from C# to VJ++

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2004-11/2776.html>

From: Jeff Van Epps (*jve86_at_lordbah.com*)

Date: 11/12/04

Date: 12 Nov 2004 04:53:45 -0800

We've been unable to get events working going from C# to VJ++. We think that the C# component is being exposed properly as a ConnectionPoint, and the Advise() from the VJ++ side seems to be working, because the delegate on the C# side has 1 item in its invocation list after the Advise(). However when we try to make the callback, we get:

```
System.NullReferenceException: Object reference not set to an instance
of an object
   at System.RuntimeType.InvokeDispMethod(String name, BindingFlags
invokeAttr, Object target, Object[] args, Boolean[] byrefModifiers,
Int32 culture, String[] namedParameters)
   at System.RuntimeType.InvokeMember(String name, BindingFlags
invokeAttr, Binder binder, Object target, Object[] args,
ParameterModifier[] modifiers, CultureInfo culture, String[]
namedParameters)
   at System.RuntimeType.ForwardCallToInvokeMember(String memberName,
BindingFlags flags, Object target, Int32[] aWrapperTypes, MessageData&
msgData)
   at IWeatherEvents.OnWeatherChanged(Int32 nTemperature)
   at TemperatureComponent.StateChange(Object sender, EventArgs args)
   at System.Windows.Forms.Timer.OnTick(EventArgs e)
   at System.Windows.Forms.Timer.DebuggableCallback(IntPtr hWnd, Int32
msg, IntPtr idEvent, IntPtr dwTime)
```

We'd appreciate help. We have a significant code base of VJ++ and we've determined that the JCLA has too many issues, so we're going to have to be hybrid for a little while. I think this is supposed to work.

```
ITemp.idl
```

```
-----
////
```

```
import "oidl.idl";
```

```
interface ITemp; // defined below
interface IWeatherEvents; // defined below

[
  uuid(ABA21482-DDE2-4cf8-9BF9-D95464AC77C2),
  helpstring("ITemp type library"),
  version(1.0)
]
library LTempLib
{
  importlib ("stdole2.tlb");

  // out going interface //////////////////////////////////
  [
    object,
    uuid(36EAE5EA-633E-474f-B6F1-5CDE5552487E),
    helpstring("IWeatherEvents"),
    oleautomation
  ]
  interface IWeatherEvents : IUnknown
  {
    [helpstring("IWeatherEvents callback")]
    HRESULT OnWeatherChanged( [in] long nTemperature );
  }

  [
    object,
    uuid(9F0C939C-99E2-4a07-94AC-9D84E25AF8EF),
    helpstring("ITemp"),
    oleautomation,
    pointer_default(unique)
  ]
  interface ITemp : IUnknown
  {
    [propput,helpstring("Set the temp")]
    HRESULT Temperature([in] float t);
    [propget,helpstring("Get the temp")]
    HRESULT Temperature([out,retval] float *t);
    [helpstring("Display temp")]
    HRESULT DisplayCurrentTemperature();
  };

  //////////////////////////////////
  interface ITemp;
  [
    uuid(33FE55E3-C85B-4475-B7B4-0B5196FAB0E7)
  ]
  coclass InsideCOM
  {
    interface ITemp;
    [source] interface IWeatherEvents;
```

```
}
}
```

Class1.cs

```
-----
using System;
using System.Windows.Forms;
using System.Runtime.InteropServices;

// Outgoing interface which the sink implements
[InterfaceType(ComInterfaceType.InterfaceIsIDispatch)]
[GuidAttribute("36EAE5EA-633E-474f-B6F1-5CDE5552487E")]
public interface IWeatherEvents
{
    void OnWeatherChanged( int nTemperature );
}

// delegate representing the OnWeatherChanged method
// of the outgoing exposed to COM aware clients.
public delegate void WeatherChangedDelegate( int nTemperature );

// [InterfaceType(ComInterfaceType.InterfaceIsDual)]
//public interface ITemperature
//{
// float Temperature { get; set; }
// void DisplayCurrentTemperature();
//
//}

// this is required to tell typelibrary generation tool REGASM.EXE
// to export the public members of TemperatureComponent into a default
// Class Interface in the generated typelibrary
[
    //ComSourceInterfacesAttribute(typeof(IWeatherEvents)),
    ComSourceInterfaces("IWeatherEvents"),
    GuidAttribute("33FE55E3-C85B-4475-B7B4-0B5196FAB0E7"),
    ProgId("TemperatureComponent.TemperatureComponentInterface")
]

// this tells the type lib generation tools to add this class
// along with all of its methods and properties to the typelib file.
//[ClassInterface(ClassInterfaceType.AutoDual)]
public class TemperatureComponent : LTempLib.ITemp
{
    private float m_fTemperature = 0;
    private Timer _t;

    // define an event associated with the WeatherChangedDelegate
    private event WeatherChangedDelegate OnWeatherChanged;

    // Public Constructor
```

```

public TemperatureComponent()
{
    m_fTemperature = 30.0f;
    _t = new Timer();
    _t.Interval = 1000; // 1 seconds
    _t.Tick += new EventHandler(StateChange);
    _t.Enabled = true;
}

private void StateChange(object sender, EventArgs args)
{
    m_fTemperature++;
    if( null != OnWeatherChanged )
    {
        System.Delegate[] foo = OnWeatherChanged.GetInvocationList(); //
just used to check in debugger that there is one thing to invoke
        Object bar = OnWeatherChanged.Target; // didn't tell me anything
helpful just _ComObject
        OnWeatherChanged( (int)m_fTemperature ); // This call causes the
exception
    }
}

//WeatherIndications GetWeatherIndications();

//Public Property Accessors (Defines get/set methods)
public float Temperature
{
    get { return m_fTemperature; }
    set { m_fTemperature = value; }
}/* end Temperature get/set property */

// Public Method that displays the Current Temperature
public void DisplayCurrentTemperature()
{
    String strTemp = String.Format("The current temperature at
Marlinspike is : " +
    "{0:####} degrees fahrenheit", m_fTemperature);
    MessageBox.Show(strTemp,"Today's Temperature");
}/* end DisplayCurrentTemperature */

// Another public method that returns an enumerated type
//public WeatherIndications GetWeatherIndications()
//{
//    if(m_fTemperature > 70)
//    {
//        return WeatherIndications.Sunny;
//    }
//    else
//    {
//        // // Let's keep this simple and just return Cloudy

```

```
// return WeatherIndications.Cloudy;
// }
//}/* end GetWeatherIndications */

}/* end class Temperature */
```

Form1.java (VJ++)

```
import com.ms.wfc.app.*;
import com.ms.wfc.core.*;
import com.ms.wfc.ui.*;
import com.ms.wfc.html.*;
import com.ms.com.*;
```

```
import itemp.ITemp;
import itemp.IWeatherEvents;
import itemp.InsideCOM;
```

```
/**
```

```
 * This class can take a variable number of parameters on the command
 * line. Program execution begins with the main() method. The class
 * constructor is not invoked unless an object of type 'Form1' is
 * created in the main() method.
 */
```

```
public class Form1 extends Form implements itemp.IWeatherEvents
```

```
{
```

```
    Object o;
    ITemp it;
    private int cookie = -1;
```

```
    private IWeatherEvents m_iEventListener = null;
```

```
    public Form1()
```

```
    {
```

```
        System.setOut( com.ms.debug.Debugger.out );
        System.setErr( com.ms.debug.Debugger.out );
        // Required for Visual J++ Form Designer support
        initForm();
```

```
        try
```

```
        {
```

```
            //TemperatureComponent t = new TemperatureComponent();
            com.ms.com._Guid g = new com.ms.com._Guid(
            "{33FE55E3-C85B-4475-B7B4-0B5196FAB0E7}");
            o = com.ms.win32.Ole32.CoCreateInstance(
```

```
                g,
                null,
                com.ms.win32.win.CLSCTX_INPROC_SERVER,
                InsideCOM.iid );
```

```
            it = (ITemp)o;
```

```
ICornerPoint cp = null;

ICornerPointContainer container = (ICornerPointContainer)o;
cp = container.FindCornerPoint( IWeatherEvents.iid );
//cookie = cp.Advise( IUnknown )
com.ms.com.ComLib.makeProxyRef(this );
cookie = cp.Advise( this ); // didn't seem to matter whether
proxyref or not

} catch( Exception e )
{
e.printStackTrace();
String s = e.getMessage();
com.ms.wfc.util.Debug.println( s );
}

// TODO: Add any constructor code after initForm call
}
public void OnWeatherChanged( int nTemperature )
{
// nTemperature;
MessageBox.show("got change event"); // Never reaches this point
label1.setText( String.valueOf(nTemperature) );
}

/**
 * Form1 overrides dispose so it can clean up the
 * component list.
 */
public void dispose()
{
super.dispose();
components.dispose();

// CUnknown iun = (CUnknown)o;
// iun.Release();
ICornerPoint cp = null;
ICornerPointContainer container = (ICornerPointContainer)o;
cp = container.FindCornerPoint( IWeatherEvents.iid );
cp.Unadvise(cookie);

}

private void button1_click(Object source, Event e)
{
it.setTemperature( (float) 12.3 );
float x = it.getTemperature();
it.DisplayCurrentTemperature();
}
```

```
/**
 * NOTE: The following code is required by the Visual J++ form
 * designer. It can be modified using the form editor. Do not
 * modify it using the code editor.
 */
Container components = new Container();
Button button1 = new Button();
Label label1 = new Label();

private void initForm()
{
    this.setText("Form1");
    this.setAutoScaleBaseSize(new Point(5, 13));
    this.setClientSize(new Point(292, 273));

    button1.setLocation(new Point(24, 24));
    button1.setSize(new Point(136, 23));
    button1.setTabIndex(0);
    button1.setText("Help me...");
    button1.addOnClick(new EventHandler(this.button1_click));

    label1.setLocation(new Point(24, 80));
    label1.setSize(new Point(168, 32));
    label1.setTabIndex(1);
    label1.setTabStop(false);
    label1.setText("help m. . . . ");

    this.setNewControls(new Control[] {
        label1,
        button1 });
}

/**
 * The main entry point for the application.
 *
 * @param args Array of parameters passed to the application
 * via the command line.
 */
public static void main(String args[])
{
    Application.run(new Form1());
}
}
```