Access APL+Win as a COM Object from .Net

The System.Runtime.InteropServices .Net namespace provides support for COM in .Net. Using this namespace a Win32 COM object (ActiveX server), e.g. APL+Win (aplwco.dll), can be accessed from .Net.

When a type library of a COM object is referenced by a .Net project in Visual Studio 2008, an 'Interop' .dll for that type library is automatically created for that project .

To facilitate easy use of APL+Win as a COM object, a C# class library (.Net assembly) has been created which will:

- Create an instance of the APL+Win com object when an instance of this .Net class is created
- Provide .Net methods which provide support for all the methods and properties of the APL+Win com
- Provide a .Net method to close the APL+Win com object use the Microsoft-recommended methodology

Here is the outline of steps to use this "APLComCsHelper.dll" class library:

- Before APL+Win can be accessed from .Net as a COM object, aplwco.dll and APLW.exe (or their run-time equivalents) must be registered as COM objects on the machine. The APL+Win Windows Interface documentation provides instructions for performing this registration. This documentation is installed to the programmer's machine when APL+Win is installed.
- Download and then open the "APLComCsHelper" solution in Visual Studio 2008.
- In addition to the "APLComCsHelper.dll", this solution contains two 'test harness' console projects. The console project which is marked in the Solution Explorer as the "StartUp Project" will use the "APLComCsHelper.dll". In Visual Studio 2008, Debug the "APLComCsHelper" solution to check that the 'test harness' console project runs properly. If it does not run properly, it is likely that Aplwco.dll or aplw.exe are not properly registered on the programmer's machine or the reference to the APLW type library in the "APLComCsHelper.dll" project in the solution needs modification.
- Examine the source code of the "test harness" console projects to see how the "APLComCsHelper.dll' is used. There is a VisualAPL and a C# console project provided illustrating the use of APL+Win as a come from VisualAPL and C# respectively.

Some potential uses for accessing APL+Win as a COM object from .Net:

- Use APL+Win functions which have not yet been converted to VisualAPL
- Use APL+Win functions from C#
- Move data from APL+Win to the .Net environment in VisualAPL or C#. With appropriate user-defined variables or functions in the APL+Win workspace.

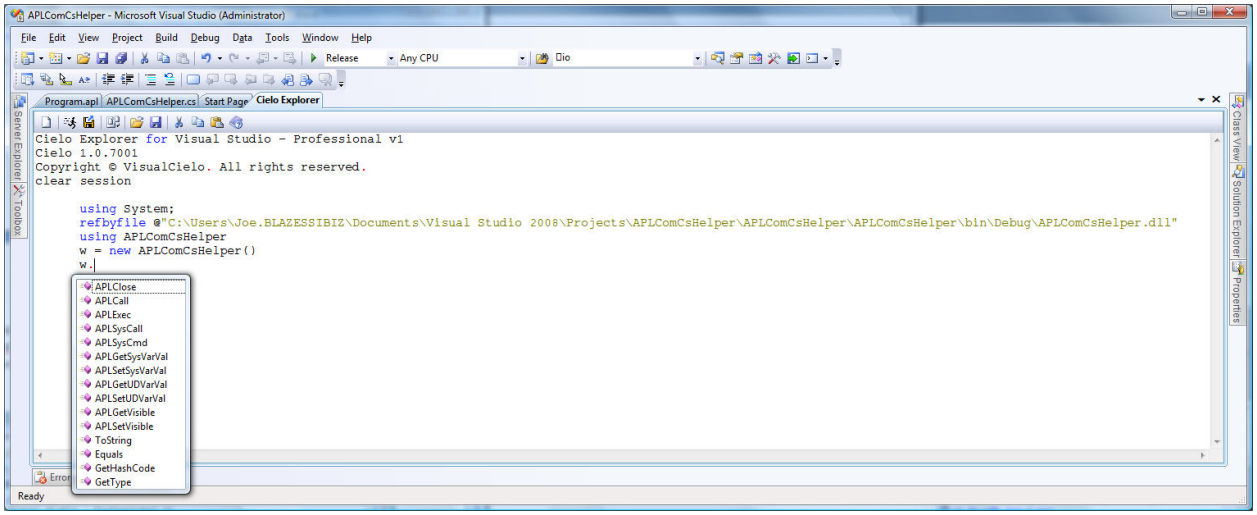
Provided below are examples using the “APLComCsHelper.dll” class library in the Cielo Explorer.

Notice how the refbyfile @”…” path points to the location of the “APLComCsHelper.dll” in the debugged solution. This path will vary for each programmer’s machine. As an alternative, the Cielo Explorer “Import Assembly” command button can be used to browse to the “APLComCsHelper.dll”. The “APLComCsHelper.dll” makes it easy to use APL+Win as a COM object in the Cielo Explorer.

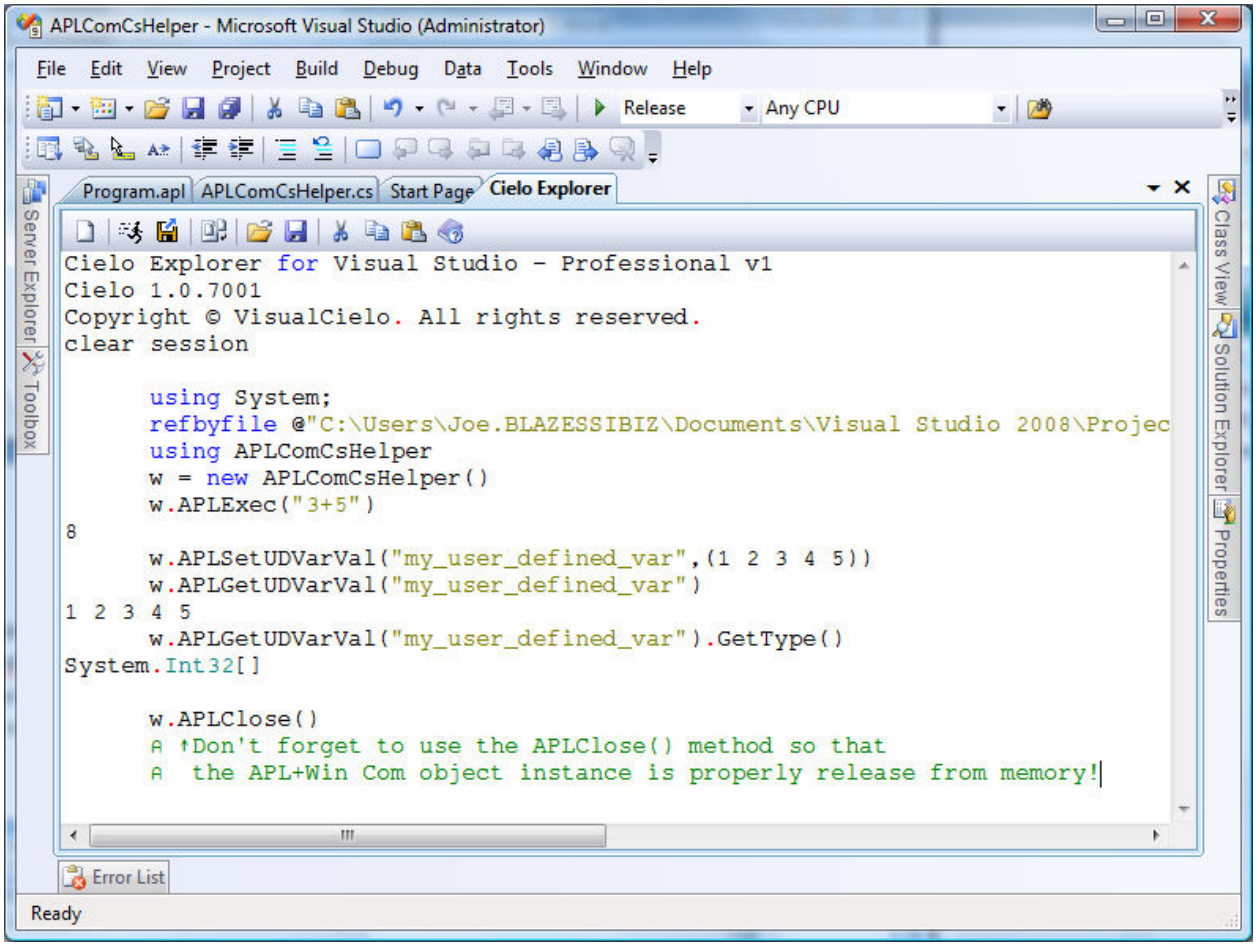
Note that the legacy wi implementation in VisualAPL cannot support the creation of instances of COM objects. This is because the methodology of accessing COM objects in .Net requires an “interop” which is automatically created with a COM objects type library is reference in a Visual Studio 2008 project.

Microsoft Visual Studio 2008 Intellisense displays the methods of provided by the APLComCsHelper class. These methods correspond to the methods and properties of the APL+Win COM object.

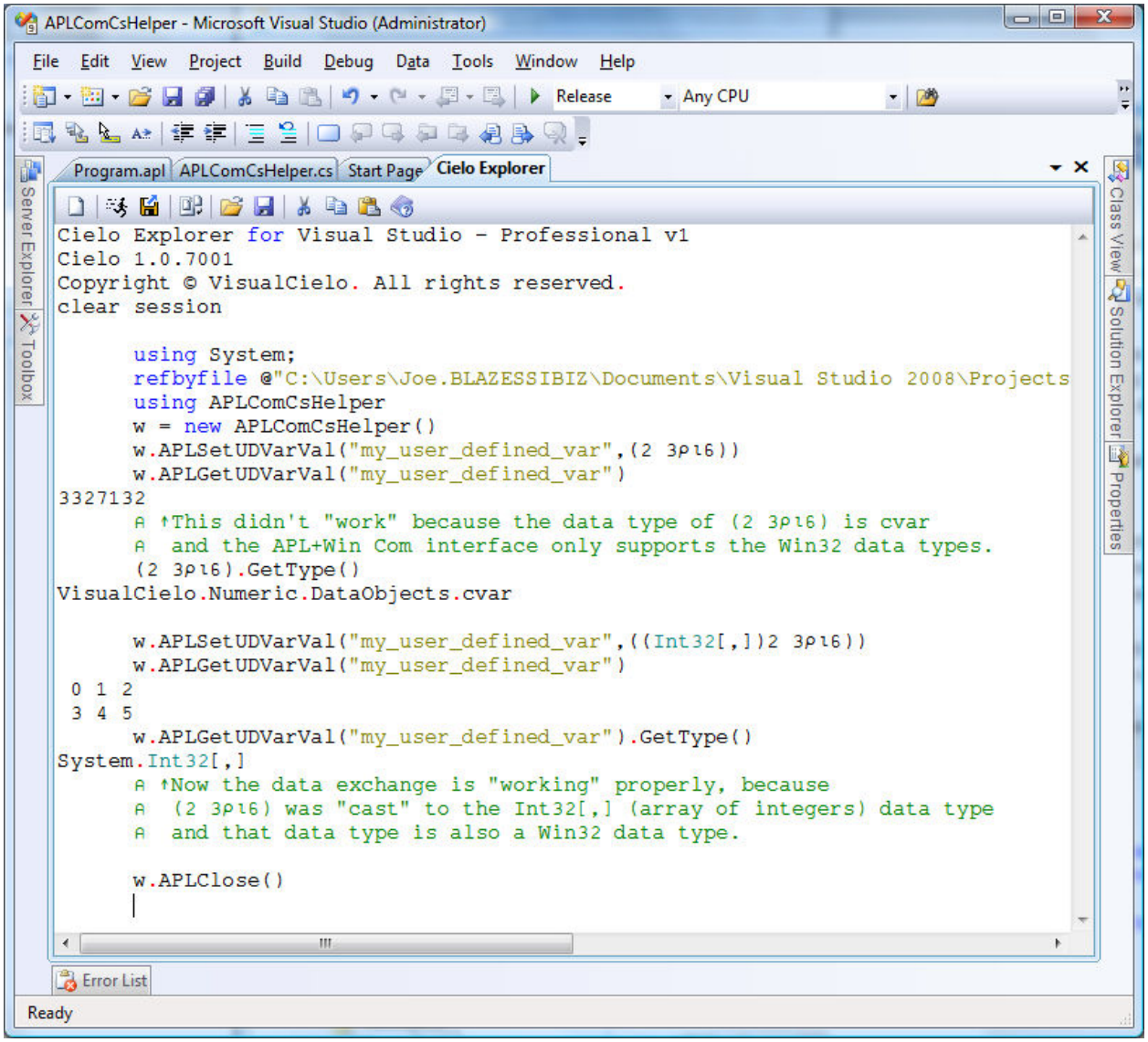
(Use the Adobe Reader magnifier tool to view this detail in the screen capture.)



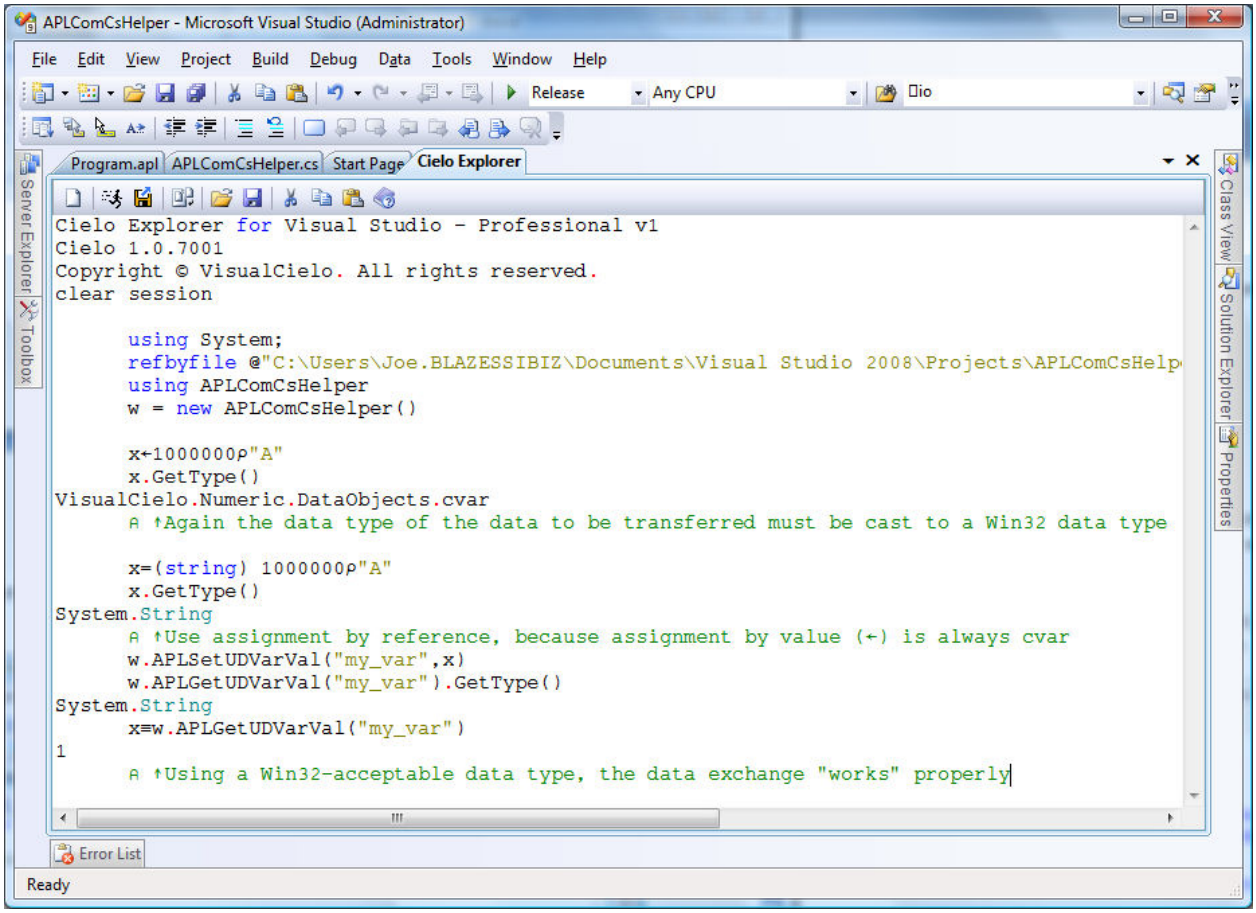
The remainder of the Cielo Explorer session is provided below. Observe how data can be transferred between APL+Win and VisualAPL using this “APLComCsHelper.dll”:



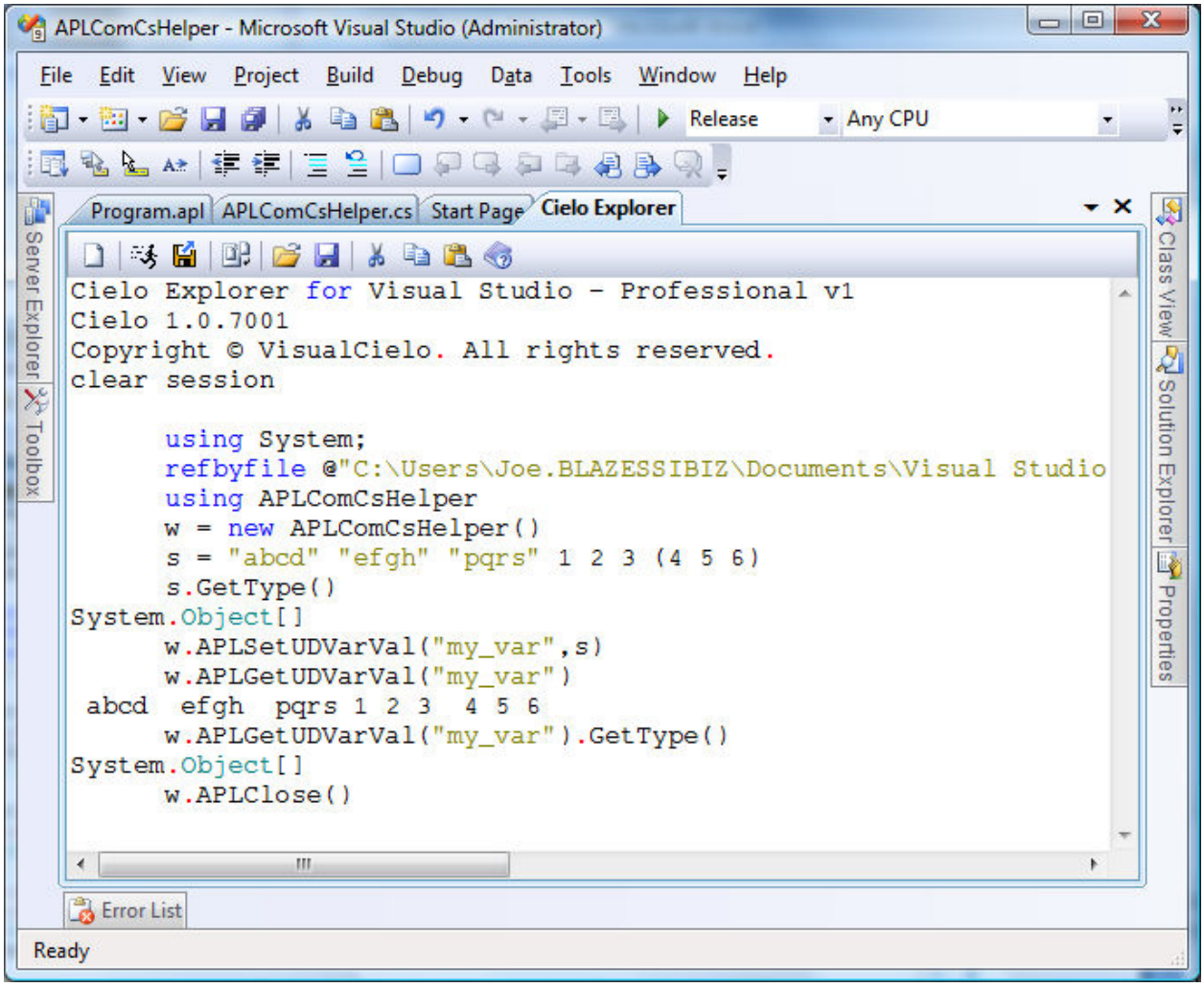
When exchanging data between .Net and the Win32-based COM object, the data type of the data must be carefully configured. Here is an numeric array example:



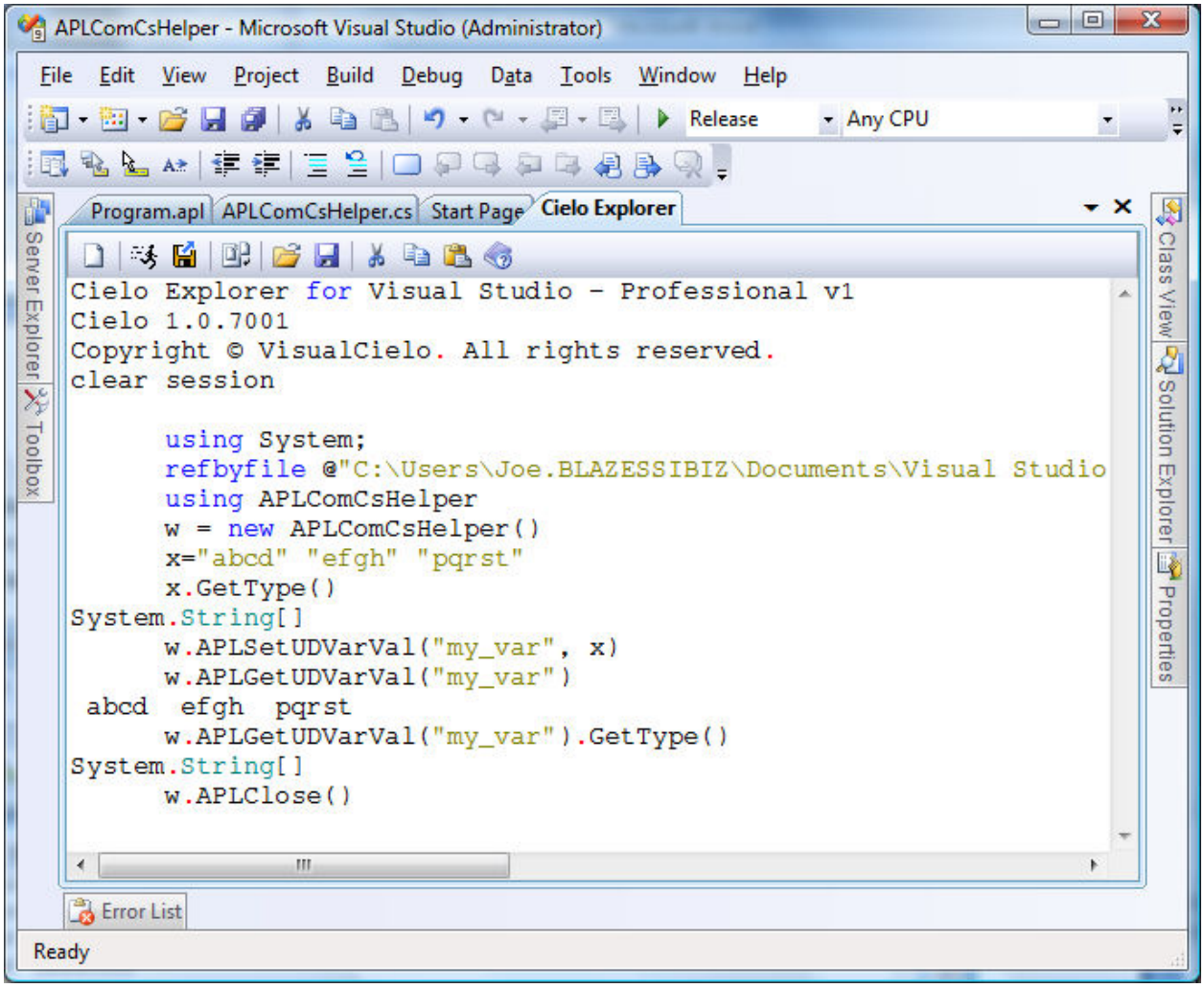
Here is another data exchange example using the string data type:



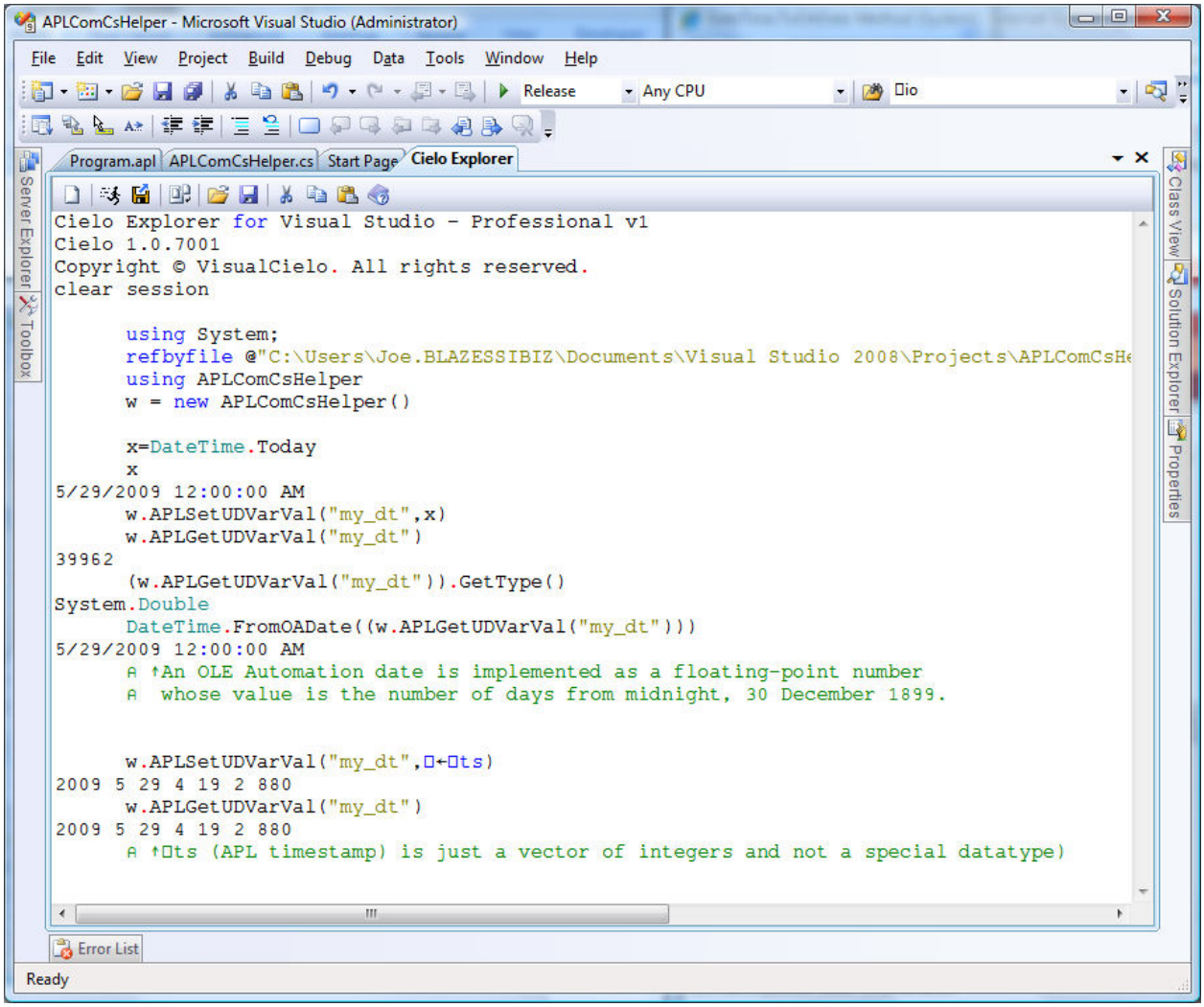
Here is a data exchange example using a mixed data type array. Notice the use of the assignment by reference (=), because using the assignment by value (←) always results in a cvar data type, which is not a Win32 data type.



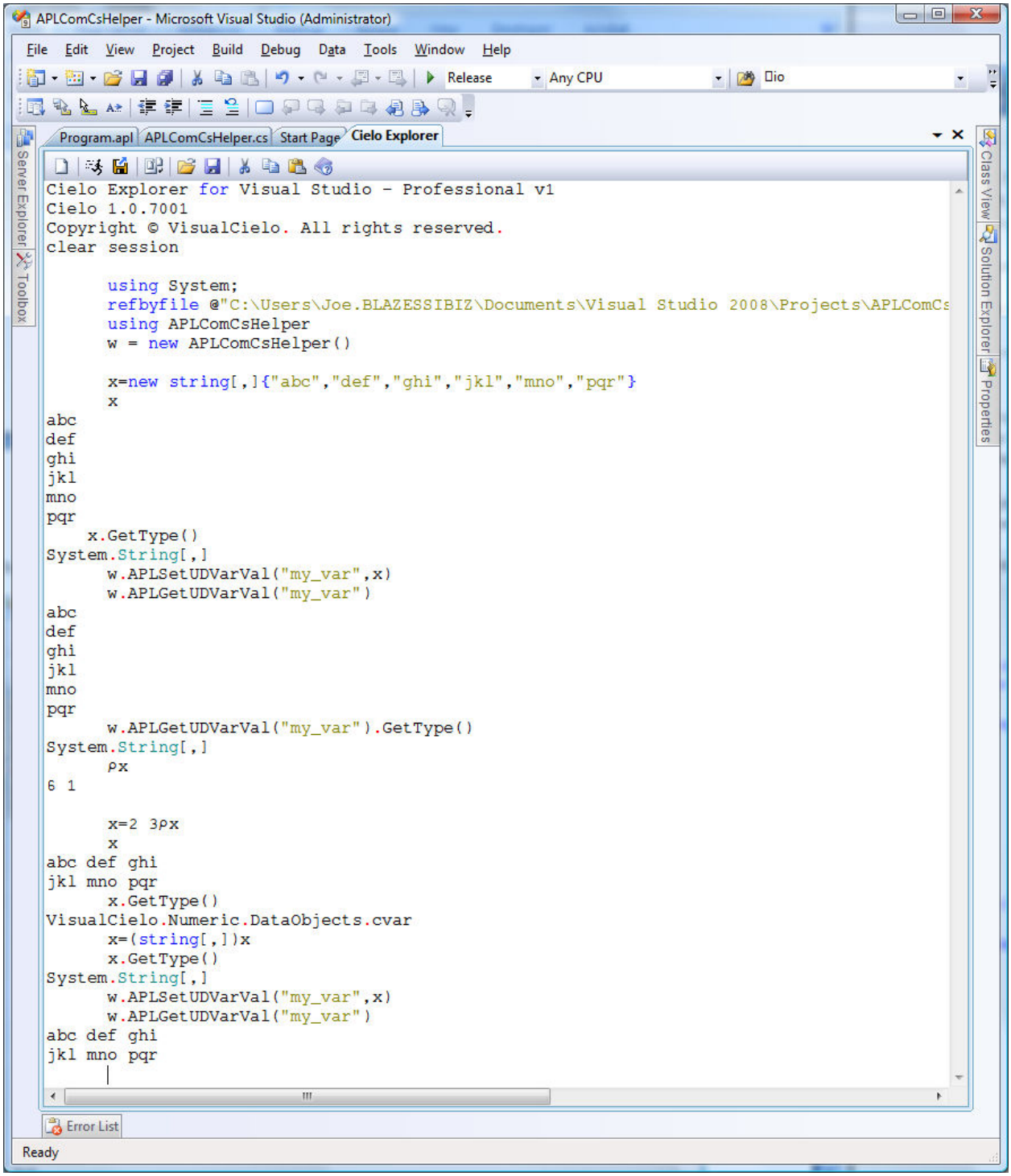
Here is a data exchange example using a string[] data type:



The DateTime data type although named the same in both .Net and Win32 requires careful handling:



Here is an example of data exchange using a 2-dimensional array of strings:



Here is an example of a large string of 50,000,000 ‘characters’: