HOW TO link to Excel

Accessing Excel using Visual Studio Tools for the Microsoft Office system

Using Visual Tools for Office enables the developer to access office documents such as Word or Excel. The example below illustrates the way in which you can export data to Excel by generating an Excel document from within Visual Studio.NET 2003.

As well as Microsoft Visual Studio.NET 2003 and Microsoft Office 2003, you must also install Visual Studio Tools for the Microsoft Office System.

The first step is to create a reference in the project to the Excel x.x Object Library (the version number depends on the installed version of Microsoft Office). Right-click on the References folder in the Solution Explorer and choose Add Reference. Choose the COM tab and pick Microsoft Excel x.x Object Library (Figure 1).

![Figure 1 Adding an Excel reference](image)

In this example, the DataMonitor application is modified so that it generates an MS-Excel document. Taking the DataMonitor (Oatc.OpenMI.Tools.Datamonitor.csproj) as a starting point, you need to change some of the methods:

- Add 'using' statements.

```csharp
using System;
using System.Windows.Forms;
using OpenMI.Standard;
using Oatc.OpenMI.Sdk.Backbone;
using System.Diagnostics;
using Oatc.OpenMI.Sdk.DevelopmentSupport;
using Oatc.OpenMI.Tools.DataMonitor;
using System.Collections;
using System.Threading;
using System.Reflection;
using Excel=Microsoft.Office.Interop.Excel;
```

- Declare workbooks, sheets etc.
namespace Oatc.OpenMI.Tools.DataMonitorPlus
{
    /// <summary>
    /// The DataMonitorPlus class
    /// </summary>
    public class DataMonitorPlus : LinkableComponent, IListener
    {
        private Excel.Application ExcelObj = new Excel.Application();
        private Excel.Workbook _workbook;
        private Excel.Sheets _sheets;
        private Excel.Worksheet _sheet;
        
        Add source to the Initialize method in order to create a workbook and a worksheet.
        
        if (ExcelObj == null)
        {
            throw new Exception("Excel object not loaded");
        }
        // Make Excel object visible and
        // add a workbook to populate
        ExcelObj.Visible = true;
        _workbook = ExcelObj.Workbooks.Add(Type.Missing);
        // Get the sheets collection from the workbook
        _sheets = _workbook.Worksheets;
        // Get the first (and only) sheet
        _sheet = (Excel.Worksheet)_sheets.get_Item(1);
        
        Modify the OnEvent code for the DataMonitor so that it writes the values to the spreadsheet.
    }
}
/// <summary>
/// OnEvent
/// </summary>
/// <param name="Event">Event</param>
public static int rownumber;

public void OnEvent (IEvent Event)
{
    ILink[] links = GetAcceptingLinks();
    Excel.Range range;

    foreach (ILink link in links)
    {
        if (link.SourceComponent==Event.Sender)
        {
            IValueSet values = Event.Sender.GetValues(Event.SimulationTime,link.ID);

            if (values is IScalarSet)
            {
                IScalarSet scalarSet = (IScalarSet)values;
                string[] subitems = new string[4+scalarSet.Count];

                        ToString();
                subitems[3] = link.SourceElementSet.ID;

                rownumber++; // need rownumber in sheet

                for (int i=0;i<scalarSet.Count;i++)
                    subitems[i+4] = scalarSet.GetScalar(i).ToString();

                ListViewItem item = new ListViewItem(subitems);
                _form.listView1.Items.Add(item);

                // write all cells to next row in Excel Worksheet
                for (int i=0;i<scalarSet.Count;i++)
                {
                    char rangeChar = (char)(i+65);
                    string strCell = rangeChar + Convert.ToString(rownumber);
                    range = _sheet.get_Range(strCell,strCell);
                    range.Value2=subitems[i];
                }
            }
        }
    }
}

The difficult part is to determine the cell co-ordinates for the spreadsheet. Excel uses ranges such as (A1:C1). In our example, such a range is being created for each value in the ValueSet (B3:B3) based on the row number and position in the 'subitems' array.
The code shows how to incorporate Excel documents in .NET code. This code needs to be completed by saving the Excel document to a specified filename (either using an argument from the OMI file or by user input) and closing the file.