Creating an IADS Custom Export Plugin using C++ VS2005

July 2010

SYMVIONICS Document SSD-IADS-053

SYMVIONICS, Inc.
Telemetry Systems

Created on July 16, 2010

Table of Contents

1. Introduction

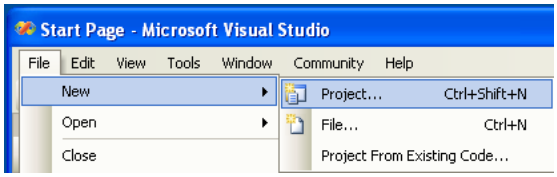
This document will guide you through the process of building an IADS Export Plugin project in Microsoft Visual Studio 2005 using C++ (VS2005). This instruction will cover: creating a new project using the ATL COM Wizard, adding the new export object, adding the self-registration code to integrate into IADS and debugging the new export plugin in IADS. Along with this tutorial is a sample export plugin project (the sample project) that provides the necessary starter code for your new project. Once your plugin is complete and registered on the IADS Client machine it will appear on the Stripchart's Data Export menu with a name provided by you.

2. Overview

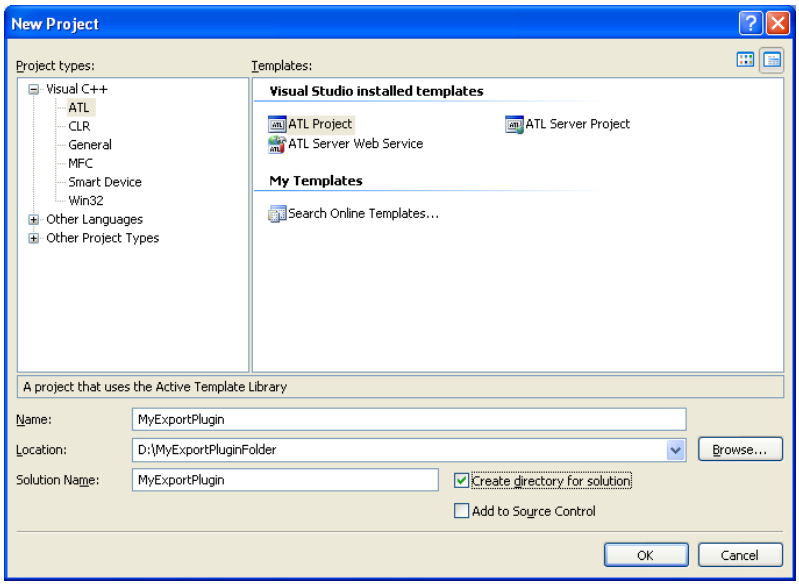
It is important to build a new project rather then simply reusing the sample project developed by SYMVIOINCS. The reason for this is that each export plugin project has its own unique ID called a GUID that is placed in the Windows Registry. If more then one group uses the sample project for their own, they can not register on the same machine. Therefore this tutorial is presented as you the user are creating a new DLL project called MyExportPlugin.dll. Further on we'll show how to copy and past code from the sample project so that you can concentrate solely on your export code and not the interfacing between it and the outer IADS Client shell.

3. Creating Your Export Plugin Project using the ATL COM Wizard

1) Open up VS2005 and Select "File -> New->Project"



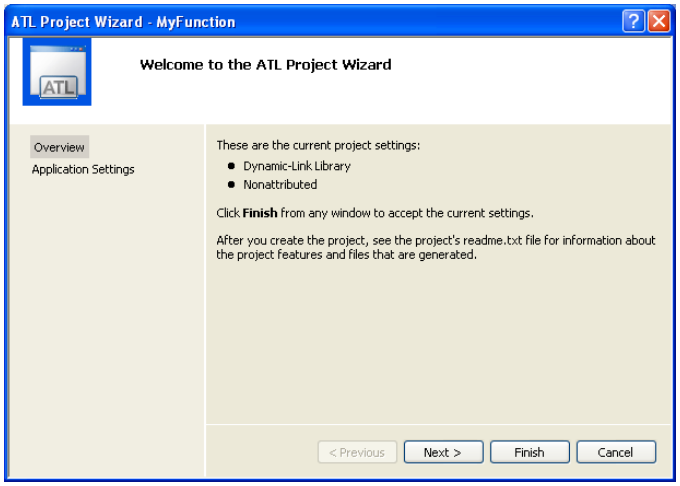
2) In the New Project dialog that appears, choose the “Visual C++->ATL” tier and click the ATL Project” option. At this point, please read the next step before you finish completing the dialog. There are some important considerations when choosing the proper project name.



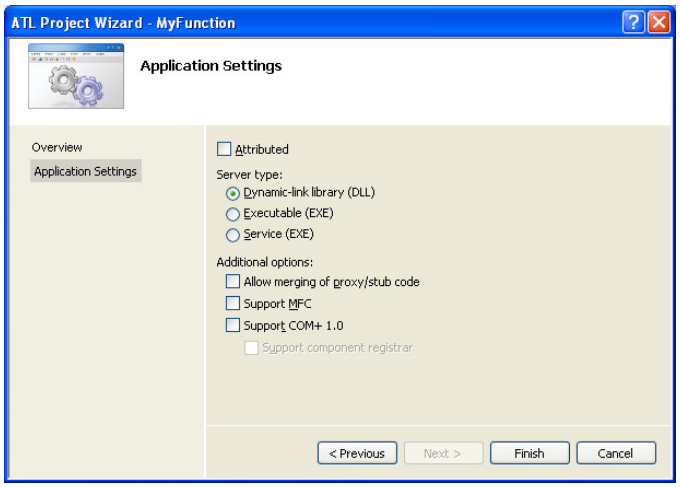
3) The project name you choose will become part of the display identifier name (aka ProgID, see step 10). When it comes time to use your plugin in IADS, users will register your DLL which automatically insert itself into the correct registry position and then be available from the IADS Stripchart’s right click menu. The menu display name will come from your plugin (more on this later). Plan on creating many plugins in one “project” (most common and easier to manage the code). Choose a general project name like “NasaExportPlugins” or “BA609ExportPlugins”. Think of the project name like a library name, and your plugins are the books.

   Now, in the fields at the bottom of the dialog, enter the project name, location, and the solution name.

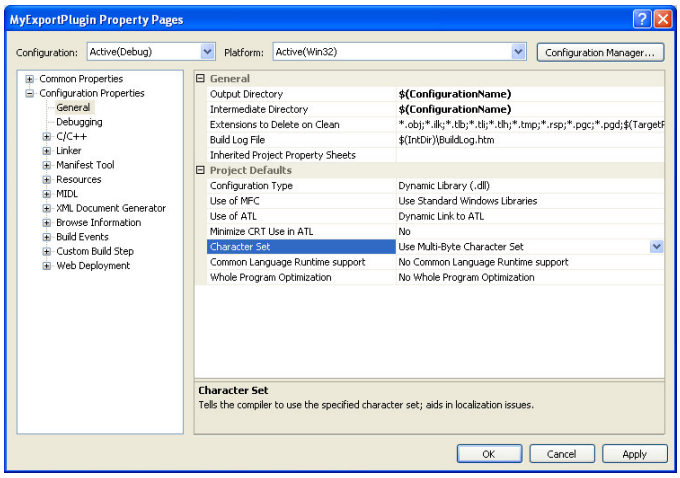
4) After pressing OK, the “ATL Project Wizard” dialog will appear as below.



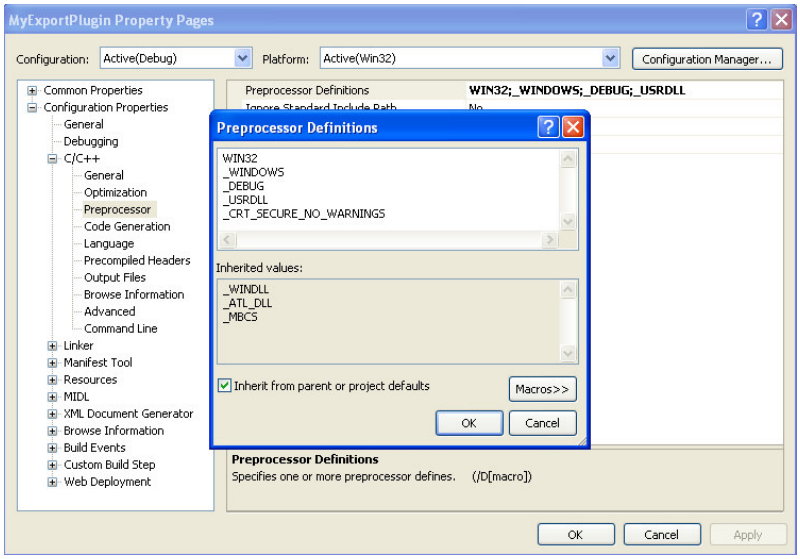
5) Click the Next button in the Wizard. On the new wizard page, ensure that the “Dynamic Link Library (DLL)” is checked. Every plugin that runs in IADS is of type DLL. Press the “Finish” button and the Wizard will create your project.



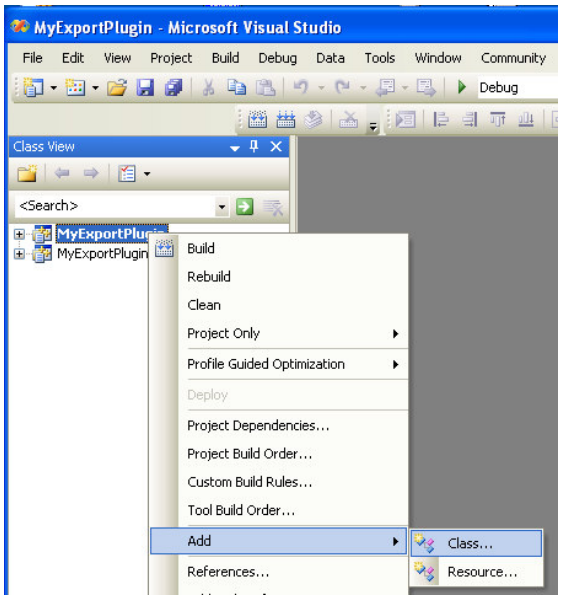
6) In the this step we’ll setup a couple optional project-wide settings that make it easier to work with provided IADS source code and eliminate warnings. Right click on the project and select properties from the menu. Click on the “General” tab and change the “Character Set” option to “Use Multi-Byte Character Set”.



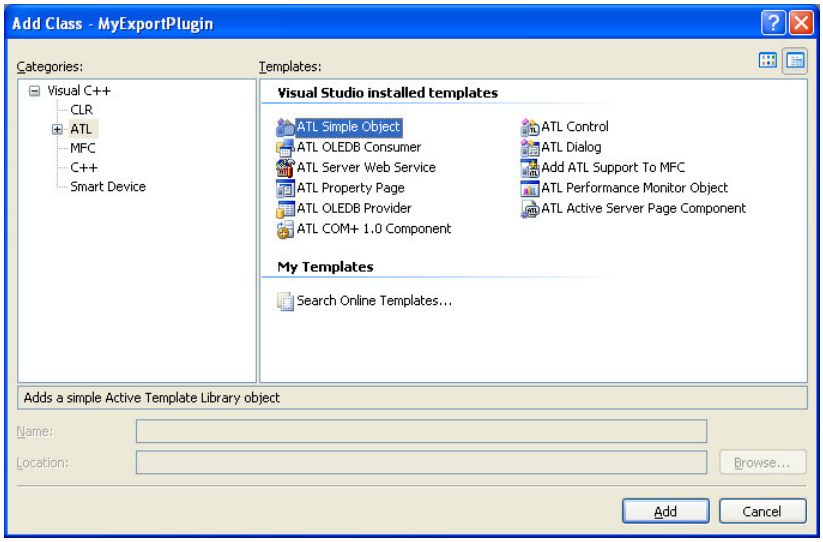
7) Again from the Project/Properties menu select C/C++, Preprocessor and add the following definition "_CRT_SECURE_NO_WARNINGS"



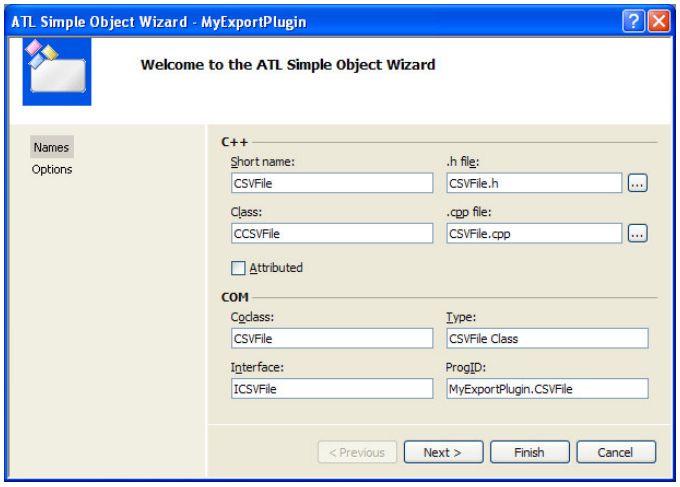
8) Now we'll add our actual export object. Go to the "Class View" tab in Visual Studio's workspace and right-click on the project name. Choose "Add->Class". At this point we are adding our first export plugin



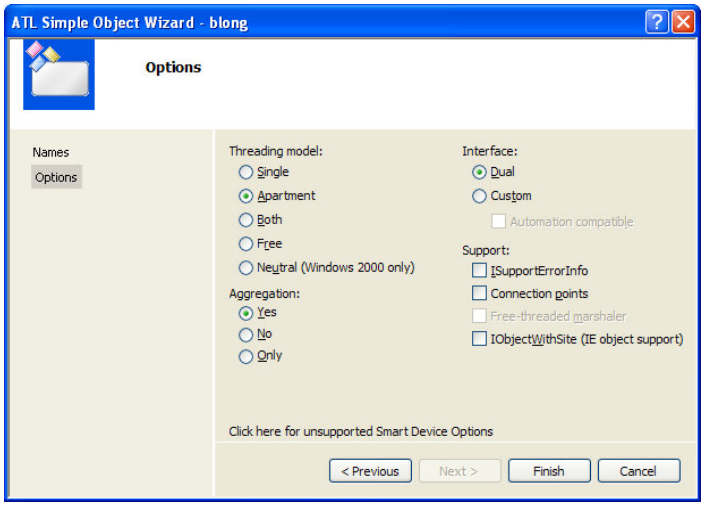
9) Upon adding a new class you will be presented with a dialog. Click the “ATL” tier and “ATL Simple Object” as shown below. When that is complete, press the “Add” button.



10) On the first tab, enter the name of your display in the “Short Name” field. The wizard will fill out the rest of the tab automatically. For this example, I used “CSVFile” as the short name. Notice that CSVFile.h and CSVFile.cpp will be created by the wizard and will be the source code files that you will edit with your own export code. The name entered will be combined with your project name and will form the final “ProgId” as shown. Press “Next” to continue.



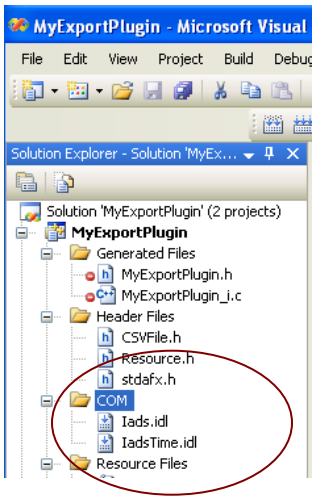
11) On the next tab (“Options”), leave everything as default (Apartment, Dual, Yes, and no other options checked).



The remaining options are basically “COM speak”. More information about these options can be found in the Microsoft documentation. At this point your project can compile and register your DLL successfully, although it doesn’t do anything yet.

3.1. Adding IADS Interface files

The next step is to add the two required IADS interface files, “iads.idl, and “IadsTime.idl”. These files provide the interface between your export plugin and the running IADS Client. They can be added anywhere in your project but I recommend creating a new folder called “COM” to put them into it



Now we need to compile the newly added IDL files to generate the output files needed in the export source code we’ll be editing. IDL files are compiled by a program called “MIDL”

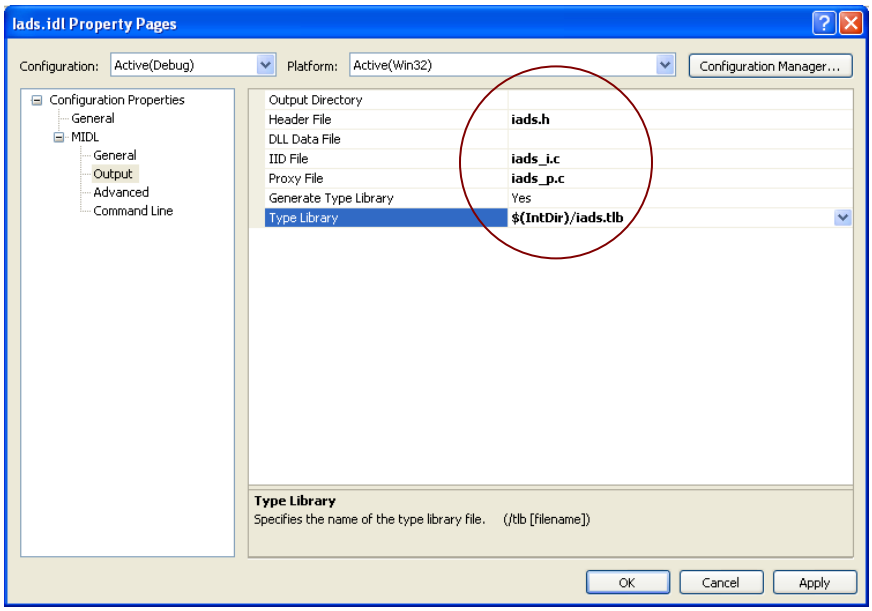
(Microsoft's IDL compiler). This is accomplished by setting up the configuration of each file in the following manner:

1) Right click on the "iads.idl" file and select properties.

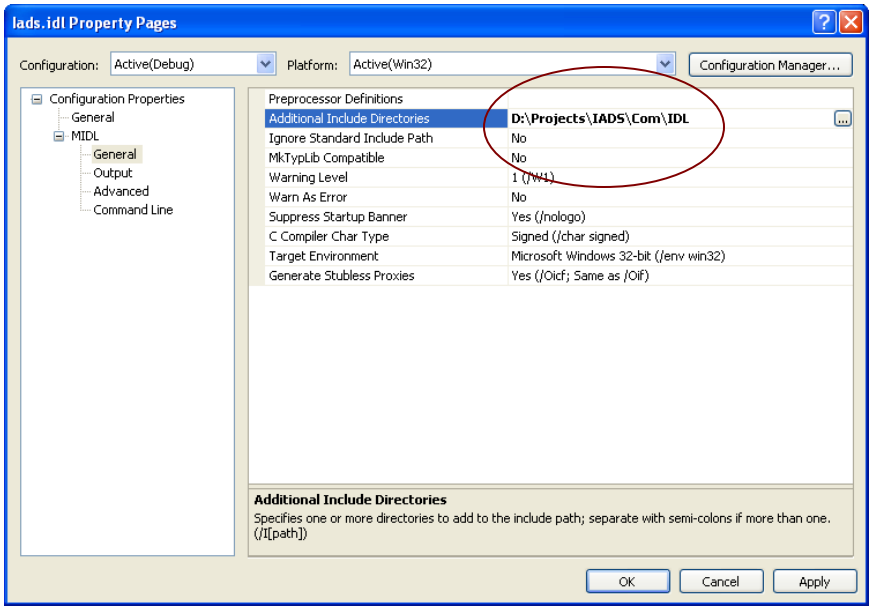
2) Click on the MIDL tier in the dialog and select the Output option.

3) Change the default names from MyExportPlugin to the name of the IDL file as shown in the following example.

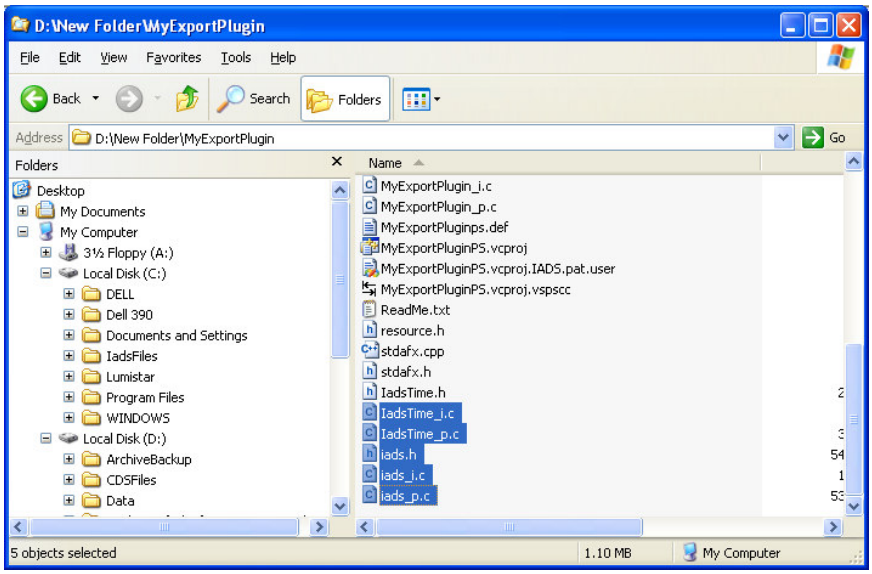
4) Repeat this step for the "IadsTime.idl file".



5) The “iads.idl” file includes the “IadsTime.idl file” so you may need to set the path for the MIDL compiler as shown:

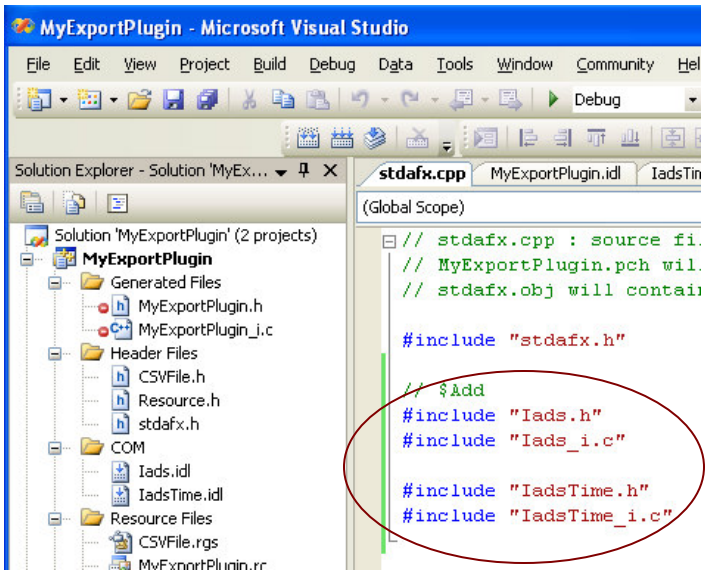


6) Make sure and compile each file individually to run MIDL and create the needed output files:

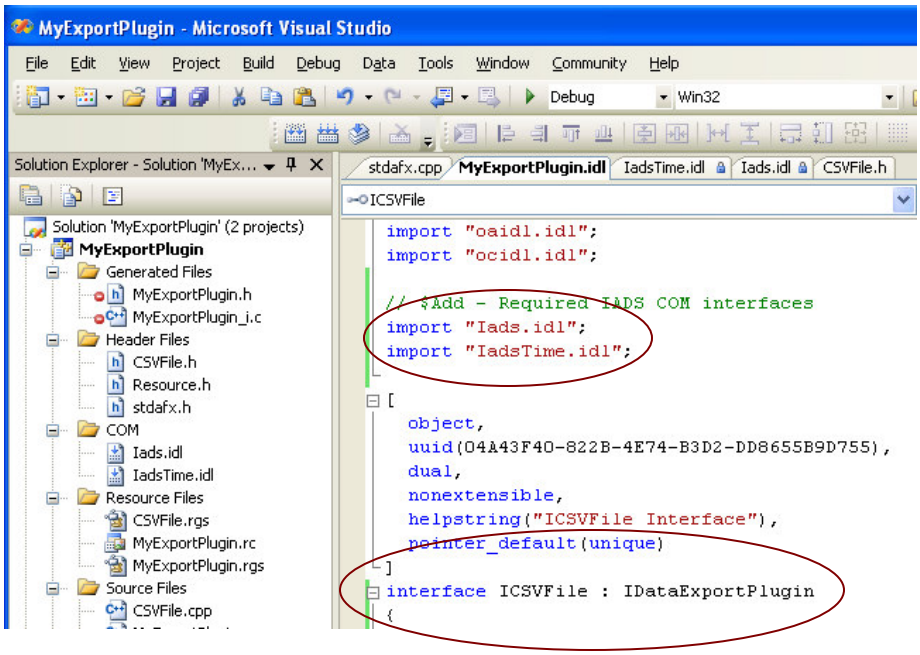


Note: iads_p.c and iadstime_p.c and are created but are not used.

7) Add the MIDL generate files to the “stdafx.c” source code file. Adding these files here will allow access to the needed global variables in your export source code. Once complete, rebuild to ensure no errors.



8) Add the IADS IDL interface file includes into your IDL file and change you export interface to derive from IDataExportPlugin instead of IDispatch that was generated by the wizard.



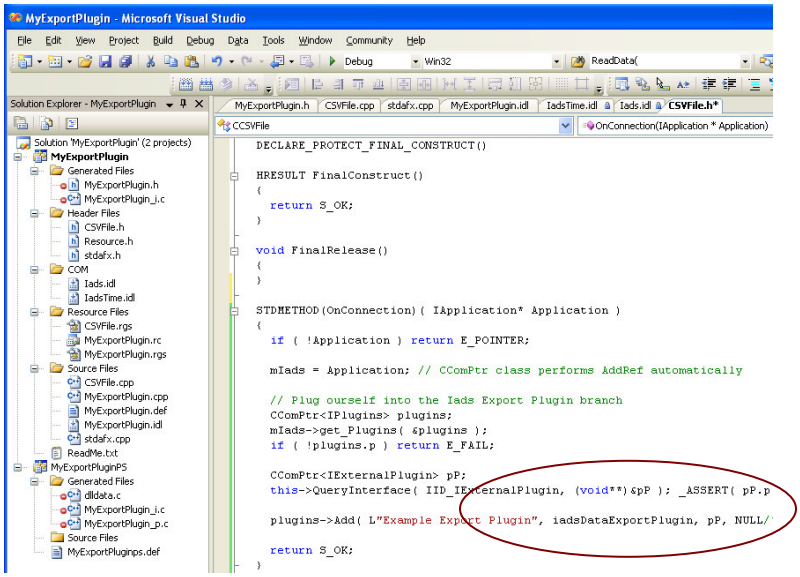
9) Just as in step 4 you may need to set the MIDL path property so that your IDL knows to the location of the “iads.idl” and IadsTime.idl files.

10) At this point all the necessary external files have been added to your project, however it will not compile until we add the routines that are expected by the IDataExportPlugin interface.

3.2. Adding IDataExportPlugin code and your export code

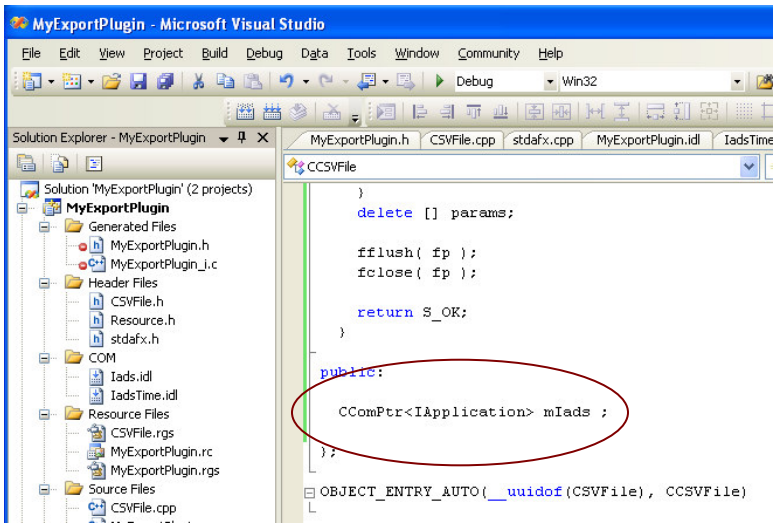
In this section we will add the required routines to that comprise the IDataExportPlugin interface, including the PerformDataExport routine which is where the entirety of your export code will reside.

1) The first step is to cut and paste the IDataExportPlugin interface code from the sample project. The source to copy is in the “CSVFiles.h” file. Cut and Paste the following routines without modification: OnConnection, OnDisconnection, PerformDataExport, ExportSelectedDisplay, ExportDataForSingleParameter, and ExportDataGroup into the CSVFile.cpp. These routines already include most of the source code you‘ll need to access the running IADS Client for necessary parameter information. Al you’ll need to do is replace the actual code that exports to a CSV File with your own output file type, such as HDF.

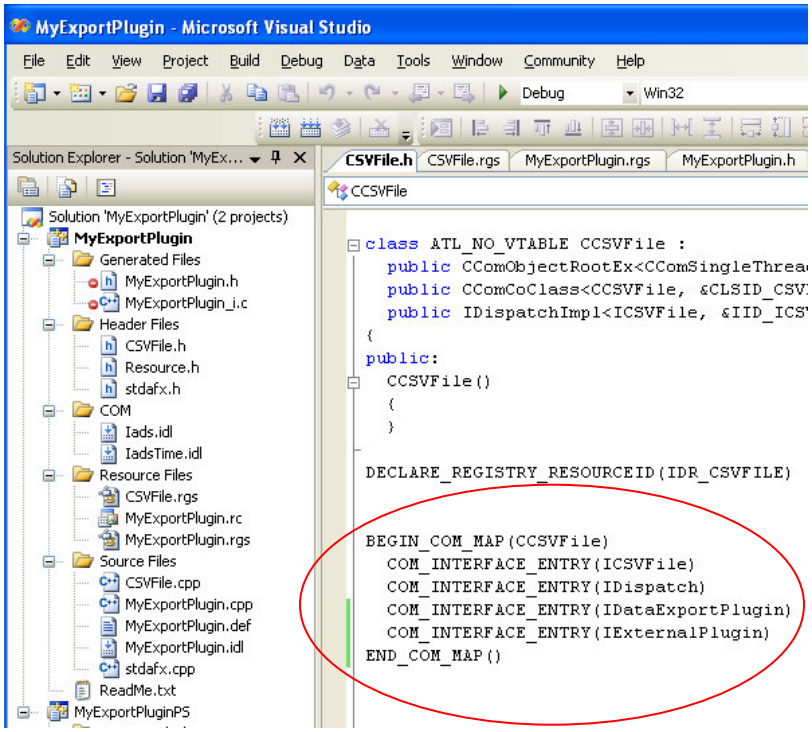


2) Modify the string argument in the plugins-add call. This will become the display name on the right click menu.

3) Next add the “mIads” member variable to the public section of your class.



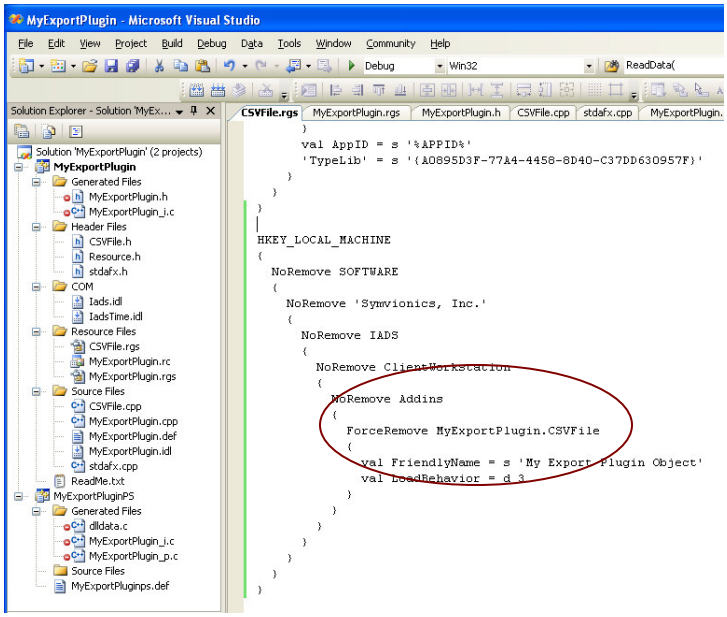
4) Finally add the IDataExportPlugin and the IExternalPlugin COM_INTERFACE_ENTRY entries to the CSVFile.h header file’s COM Map as shown



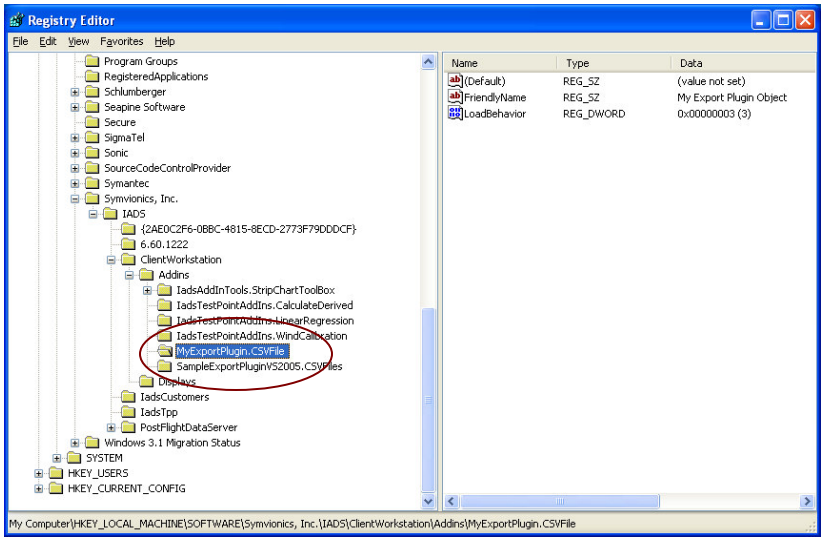
5) At this point your project should compile without errors or warnings.

3.3. Make your DLL self-register for use in IADS

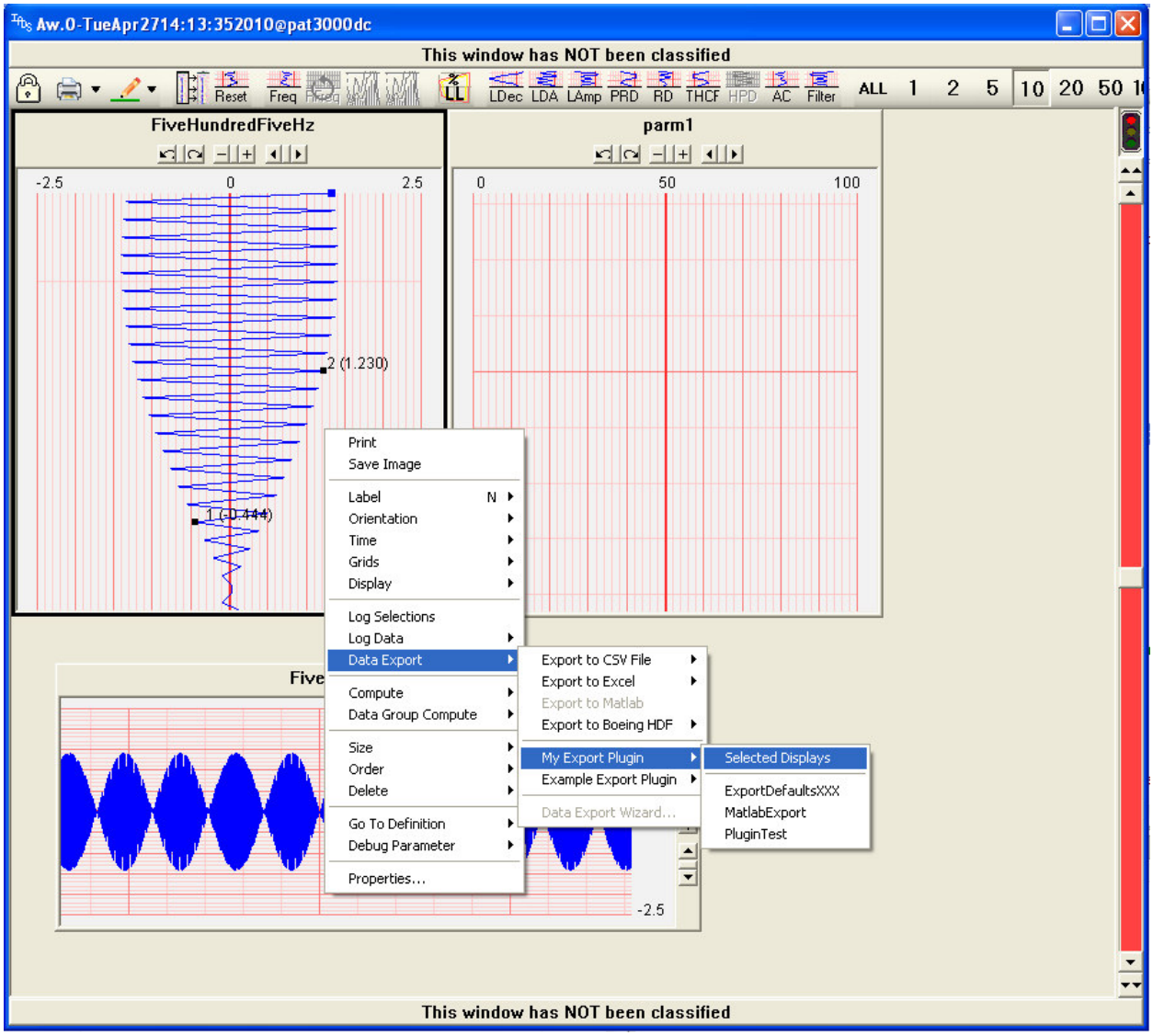
1) In order for your export plugin to be shown in the Strip Chart's Data Export menu, registration code must be added to your objects registration script. The easiest way to do this is to cut and paste the code from the sample project into the CSVFile.rgs file and change the name of the ProgId to the new projects ProgId, as shown here:



2) You can use regedit to verify that the registration code worked properly by putting the ProgId into the HKEY_LOCAL_MACHINE/Symvionics, Inc./IADS/ClientWorkstation/Addins registry hive.

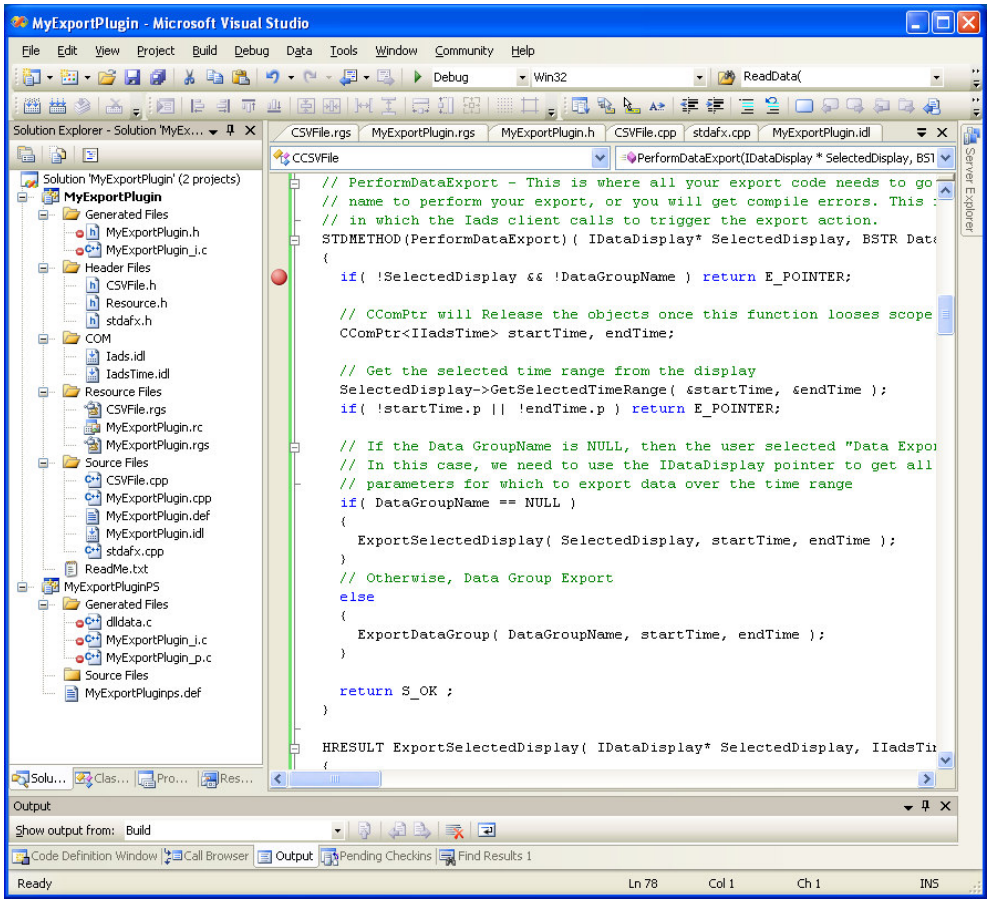


3) Finally you can run IADS (requires version 7.0) and verify that MyExportPlugin was added to the Strip chart Data Export menu as shown here.

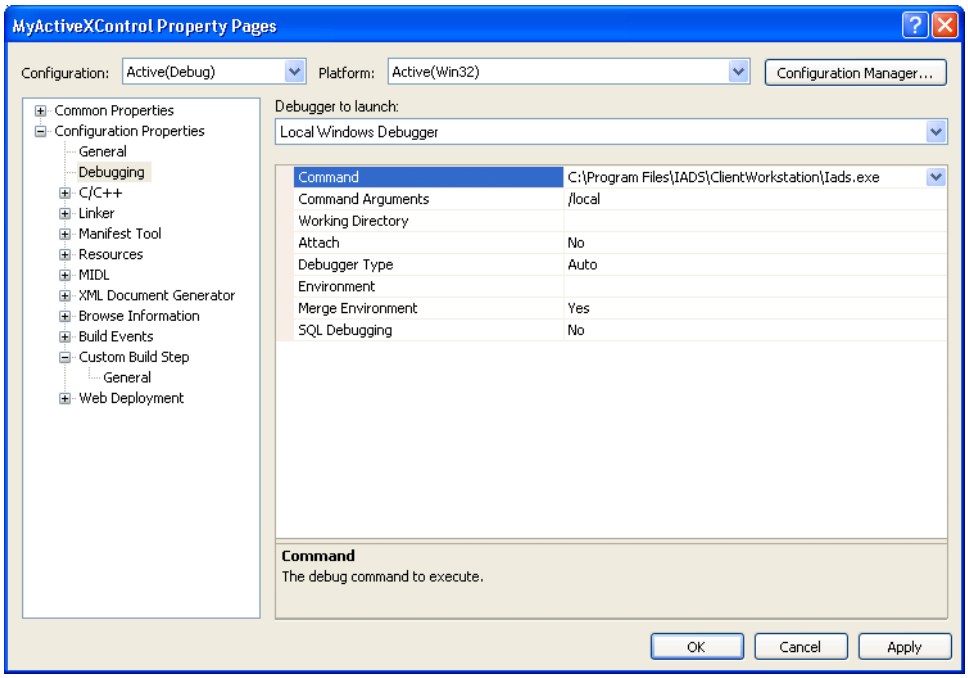


4. Debugging Your New Plugin in IADS

1) Place a break point in your “PerformDataExport” method for testing.



2) In Visual Studio, select the “Project->Properties” drop down menu. Pick “Iads.exe” as your “Command”. It’s in your “C:\Program Files\Iads” directory. Add “/local” to your “Command Arguments”. Build your project and click on the “Go” command. Iads will start.



3) Right Click on the Strip chart and select the Properties option and then the Data Export option. Select “My Export Plugin” to hit your breakpoint in your code in the “PerformDataExport routine.

5. Conclusion

The SampleExportPluginVS2005 demonstration project is available for download from the SYMIONICS web site at the following location:

http://iads.symvionics.com/MainPages/DownloadsPage.htm

If you have any further questions, you can search the IADS Google Group or post a question:

http://groups.google.com/group/iads