ASCOM C# LocalServer Development Framework Checklist

From the **ASCOM Platform 6.1SP1** and **ASCOM Platform Developer Components** documentation, there are two documented ways of developing an ASCOM Local Server:

A. ("Making a Local Server based Driver.pdf" recommendation)
- Start a Visual Studio solution with the local server template
- add 1 or more driver projects using the driver template
- develop and test the driver(s) as in-proc DLLs
- change driver(s) to be LocalServer served
- test drivers served by the local server

B. (LocalServer "ReadMe.htm" recommendation)
- Start a Visual Studio solution with the driver template
- optionally add more driver projects using the driver template
- develop and test the driver(s) as in-proc DLLs
- add a project with the local server template
- change driver(s) to be LocalServer served
- test drivers served by the local server

Type B is much more work to put the local server files in the correct namespace and other naming cleanup from the template wizard (not really recommended here) - it requires "Find In Files" and replacements to correct naming usage.

The following is a checklist and an annotated walk-through example of generating an ASCOM C# LocalServer for a fictitious company/product called Acme serving an ASCOM Focuser driver and a FilterWheel driver assumed to share a single serial port via a custom controller - therefore the need to use a LocalServer. The screen images items associated with each step are highlighted in **red**. The end result is a LocalServer skeleton framework with a tested "working", "non-functional" local server serving two ASCOM drivers.

- - "working" means the served drivers pass the Conformance Checker tool
- - "non-functional" means they control no actual hardware (yet)

Development Environment
- 64-bit Windows 7
- Visual C# 2010 Express Edition
- ASCOM Platform 6.1SP1 installed
- ASCOM Platform Developer Components installed
- ASCOM Driver Conformance Checker installed
- ASCOM 6 Local Server Template (C#) installed (not part of Platform 6.1SP1)

Note: All projects in a multi-project solution in Express Editions of Visual Studio must use the same programming language.

Visual C# 2010 Express ← Run as administrator !!!

Checklist	Image
☐ 1) New Project - **ASCOM 6 Local Server Template (C#)** - Name: **Acme**	NewProject.png
☐ 2) File➢Save All – - Name: **Server** - Location: ...**\CSharp Projects\ASCOM** - Solution Name: **Acme** ← **MAKE SURE**!!! because Visual Studio makes same as Name!!! - [✓] Create directory for solution	SaveProject.png
☐ 3) Set Server Properties➢Application➢Assembly name: **ASCOM.Acme.Server**	ServerAssemblyName.png

☐ 4) Add New Project for Focuser driver — AddFocuser.png

(File➢Add➢New Project... not available in C# Express)

- Right-Click solution name in Solution Explorer➢Add➢New Project...
  (if Solution not shown in Solution Explorer, do Tools➢Options➢Projects and Solutions➢General, [✓] Always show solution)
- **ASCOM Device Driver (C#)**
  - Name: **Focuser**

☐ 5) ASCOM Driver Project Wizard — WizardFocuser.png

- Device Class - **Focuser**
- Device Name/Model - **Acme**
- Create

☐ 6) Set Focuser Properties➢Application➢Target framework: **.NET Framework 3.5** — FocuserNET3.5.png
The driver must use the same .NET Framework as the server.

☐ 7) In Solution Explorer for Focuser, select and delete References➢ASCOM.Utilities.Video since it requires .NET 4.0 and the current configuration is for .NET 3.5

☐ 8) Add New Project for FilterWheel driver — AddFilterWheel.png

(File➢Add➢New Project... not available in C# Express)

- Right-Click solution name in Solution Explorer➢Add➢New Project...
  (if Solution not shown in Solution Explorer, do Tools➢Options➢Projects and Solutions➢General, [✓] Always show solution)
- **ASCOM Device Driver (C#)**
  - Name: **FilterWheel**

☐ 9) ASCOM Driver Project Wizard — WizardFilterWheel.png

- Device Class - **FilterWheel**
- Device Name/Model - **Acme**
- Create

☐ 10) Set FilterWheelProperties➢Application➢Target framework: **.NET Framework 3.5** — FilterWheelNET3.5.png
The driver must use the same .NET Framework as the server.

☐ 11) In Solution Explorer for FilterWheel, select and delete References➢ASCOM.Utilities.Video since it requires .NET 4.0 and the current configuration is for .NET 3.5

☐ 12) Build the solution (F6)

☐ 13) Run the ASCOM **Conform** tool. If it is running in 64 bit mode, change it to run in 32 bit mode with: — Conform64Bits.png Conform32Bits.png

Options➢Conformance Options➢General➢Conform Settings:
[✓] Run as 32bit on a 64bit OS

(this is needed because Visual Studio's *Register for COM interop* only registers the drivers as a 32bit COM driver, but not also as a 64bit COM driver on a 64-bit machine - as would be done by the Inno Setup installer)

☐ 14) Using the ASCOM **Conform** tool, Options➢Check Focuser, Options➢Select Driver, select the *ASCOM Focuser Driver for Acme*. — ConformChooserFocuser.png

☐ 15) Select ASCOM Focuser Chooser➢Properties... to get the Acme Setup dialog for the Focuser — ConformFocuserProperties.png

☐ 16) Run the Check Conformance and verify that no errors, warnings or issues are found and the Focuser driver passes ASCOM validation!! — ConformFocuser.png

☐ 17) Using the ASCOM **Conform** tool, Options➢Check Filter Wheel, Options➢Select Driver, select the *ASCOM FilterWheel Driver for Acme*. — ConformChooserFilterWheel.png

☐ 18) Select ASCOM FilterWheel Chooser➢Properties... to get the Acme Setup dialog for the FilterWheel — ConformFilterWheelProperties.png

☐ 19) Run the Check Conformance and verify that no errors, warnings or issues are found and the FilterWheel driver passes ASCOM validation!! — ConformFilterWheel.png

☐ 20) Add New Project for application for testing the drivers AddTestDrivers.png
- Right-Click solution name in Solution Explorer➢Add➢New Project...
- **ASCOM Driver Test Forms Application (C#)**
- Name: **TestDrivers**

☐ 21) ASCOM Driver Project Wizard WizardFocuser.png
- Device Class - **Focuser**
- Device Name/Model - **Acme**
- Create

☐ 22) Right-Click Solution Explorer➢TestDrivers project➢Set as Startup Project to set the **TestDrivers**project as the startup project

☐ 23) Build the solution (F6)

☐ 24) Run the code (F5), click the test form's Choose button, select the *ASCOM Focuser Driver for Acme*., select ASCOM Focuser Chooser➢Properties... to get the Acme Setup dialog for the Focuser, OK those dialogs and verify the ASCOM.Acme.Focuser is shown on the test form. TestDriversResults.png

At this point, additional code can be added to the separate Focuser and FilterWheel drivers to independently control the Focuser and FilterWheel hardware and additional code and controls can be added to the TestDrivers project to exercise and debug the features of the in-proc DLL Focuser and FilterWheel drivers.

☐ 25) Clean the solution with Build➢Clean Solution so that the driver will be automatically unregistered from COM and ASCOM
(if menu Build➢Clean Solution is not shown, use Tools➢Customize➢Commands➢Menu bar: Build➢Add Command...➢Categories: Build, Commands: Clean Solution, OK, Close to add that menu item)

At this point, the Acme Focuser and FilterWheel should no longer be available in **Conform**'s Select Driver.

Now, make the changes to incorporate the **LocalServer** functionality.

☐ 26) In Focuser Properties➢Build➢ FocuserOutput.png
- Set Configuration: **All Configurations**, Platform: **Active (Any CPU)**
- Set Output➢Output path: **..\Server\bin\Debug**
- Disable Output➢**[ ] Register for COM interop**

☐ 27) Right-Click Focuser project➢Add Reference...➢Projects➢Server to add a reference to the Server project to the Focuser Project

☐ 28) Add the following class to the Focuser project's Driver.cs file just before the Focuser class definition: FocuserDriverMods.png

```
internal class FocuserLocalServerConstants
{
    internal const string DRIVER_ID = "ASCOM.Acme.Focuser";
    internal const string DRIVER_DESCRIPTION = "Acme Focuser";
}
```

☐ 29) Add the following attribute declarations to the Focuser project's Driver.cs Focuser class definition: FocuserDriverMods.png

```
[ProgId(FocuserLocalServerConstants.DRIVER_ID)]
[ServedClassName(FocuserLocalServerConstants.DRIVER_DESCRIPTION)]
```

☐ 30) Change the Focuser project's Driver.cs Focuser class definition to inherit *ReferenceCountedObjectBase*: FocuserDriverMods.png

```
public class Focuser : ReferenceCountedObjectBase, IFocuserV2
```

☐ 31) Change the Focuser project's Driver.cs *driverID* definition to: FocuserDriverMods.png

```
internal static string driverID =
                    FocuserLocalServerConstants.DRIVER_ID;
```

☐ 32) Change the Focuser project's Driver.cs *driverDescription* definition to: FocuserDriverMods.png

```
private static string driverDescription =
                    FocuserLocalServerConstants.DRIVER_DESCRIPTION;
```

☐ 33) Remove the Focuser project's Driver.cs ASCOM registration region code

- ☐ 34) In FilterWheel Properties➢Build➢ FilterWheelOutput.png
  - Set Configuration: **All Configurations**, Platform: **Active (Any CPU)**
  - Set Output➢Output path: **..\Server\bin\Debug**
  - Disable Output➢**[ ] Register for COM interop**
- ☐ 35) Right-Click FilterWheel Project➢Add Reference...➢Projects➢Server to add a reference to the Server project to the FilterWheel project
- ☐ 36) Add the following class to the FilterWheel project's Driver.cs file just before the FilterWheel class definition: FilterWheelDriverMods.png

```
internal class FilterWheelLocalServerConstants
{
    internal const string DRIVER_ID = "ASCOM.Acme.FilterWheel";
    internal const string DRIVER_DESCRIPTION = "Acme FilterWheel";
}
```

- ☐ 37) Add the following attribute declarations to the FilterWheel project's Driver.cs FilterWheel class definition: FilterWheelDriverMods.png

```
[ProgId(FilterWheelLocalServerConstants.DRIVER_ID)]
[ServedClassName(FilterWheelLocalServerConstants.DRIVER_DESCRIPTION)]
```

- ☐ 38) Change the FilterWheel project's Driver.cs FilterWheel class definition to inherit *ReferenceCountedObjectBase*: FilterWheelDriverMods.png

```
public class FilterWheel: ReferenceCountedObjectBase,
                                                IFilterWheelV2
```

- ☐ 39) Change the FilterWheel project's Driver.cs *driverID* definition to: FilterWheelDriverMods.png

```
internal static string driverID =
                FilterWheelLocalServerConstants.DRIVER_ID;
```

- ☐ 40) Change the FilterWheel project's Driver.cs *driverDescription* definition to: FilterWheelDriverMods.png

```
private static string driverDescription =
                FilterWheelLocalServerConstants.DRIVER_DESCRIPTION;
```

- ☐ 41) Remove the FilterWheel project's Driver.cs ASCOM registration region code
- ☐ 42) Right-Click Solution Explorer➢Server project➢Set as Startup Project to set the local server as the startup project ServerStartup.png
- ☐ 43) Build the solution (ignore 2 mismatch warnings for now, see NOTES)
- ☐ 44) Add Server Properties➢Debug➢Start Options➢Command line arguments: **/register** ServerRegister.png
- ☐ 45) Run the project (to have the local server register the drivers with COM and ASCOM) (this registers the drivers for both 32 bit and 64 bit {on a 64-bit machine}, so the ASCOM Conform tool can now be run as 64 bits without problems)
- ☐ 46) Using the ASCOM Conform tool, Options➢Check Focuser, Options➢Select Driver, select the Acme Focuser
- ☐ 47) Select ASCOM Focuser Chooser➢Properties... to get the Acme Setup dialog for the Focuser
- ☐ 48) Run the Check Conformance and verify that no errors, warnings or issues are found and the ASCOM.Acme.Focuser driver passes ASCOM validation!!
- ☐ 49) Using the ASCOM Conform tool, Options➢Check Filter Wheel, Options➢Select Driver, select the Acme FilterWheel
- ☐ 50) Select ASCOM FilterWheel Chooser➢Properties... to get the Acme Setup dialog for the FilterWheel
- ☐ 51) Run the Check Conformance and verify that no errors, warnings or issues are found and the ASCOM.Acme.FilterWheel driver passes ASCOM validation!!
- ☐ 52) Change Server Properties➢Debug➢Start Options➢Command line arguments: **/unregister**
- ☐ 53) Run the project (to have the local server unregister the drivers with COM and ASCOM)

At this point, code can be changed in the Focuser and FilterWheel drivers to appropriately work with the common Focuser and FilterWheel hardware and additional code and controls can be added to the TestDrivers project to exercise and debug the features of the LocalServer-served Focuser and FilterWheel drivers.

When the ASCOM local server and drivers development is complete, the ASCOM Driver Install Script Generator an be used to generate an Inno Setup script to generate a Windows setup executable that can be used to distribute the server and drivers just developed.
Note: The server/driver solution should be closed in the IDE before running the Inno Setup compiler.

NOTES:

- The following warning occurs for both the Focuser driver and the FilterWheel driver when building the solution: BuildWarnings.png

  "There was a mismatch between the processor architecture of the project being built "MSIL" and the processor architecture of the reference "...\Server\bin\Debug\ASCOM.Acme.Server.exe", "x86"."

  These warnings occur because the drivers are built by default from the templates for "**AnyCPU**" while the server is built by default from the template for "**x86**". The server ***must*** be built for "**x86**" (a served driver fails to load when the server is built for "**AnyCPU**"), so the drivers need to also be built for "**x86**". A stand-alone in-proc driver will not work on a 64-bit OS unless it is built for "**AnyCPU**", but a LocalServer-served driver will work on a 64-bit (and 32-bit) OS when built for "**x86**").

  To resolve these warnings: (the easier way described, Configuration Manager can also be used)

  - Enable only 32-bit code generation for the Focuser driver by modifying (outside of the Visual Basic IDE environment) the <PlatformTarget> tag in the Focuser driver's Focuser.csproj file to be **x86** located under the following tags: (it was "AnyCPU")

    <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
    <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
    (i.e. in Focuser.vbprog: <PlatformTarget>**x86**</PlatformTarget>)

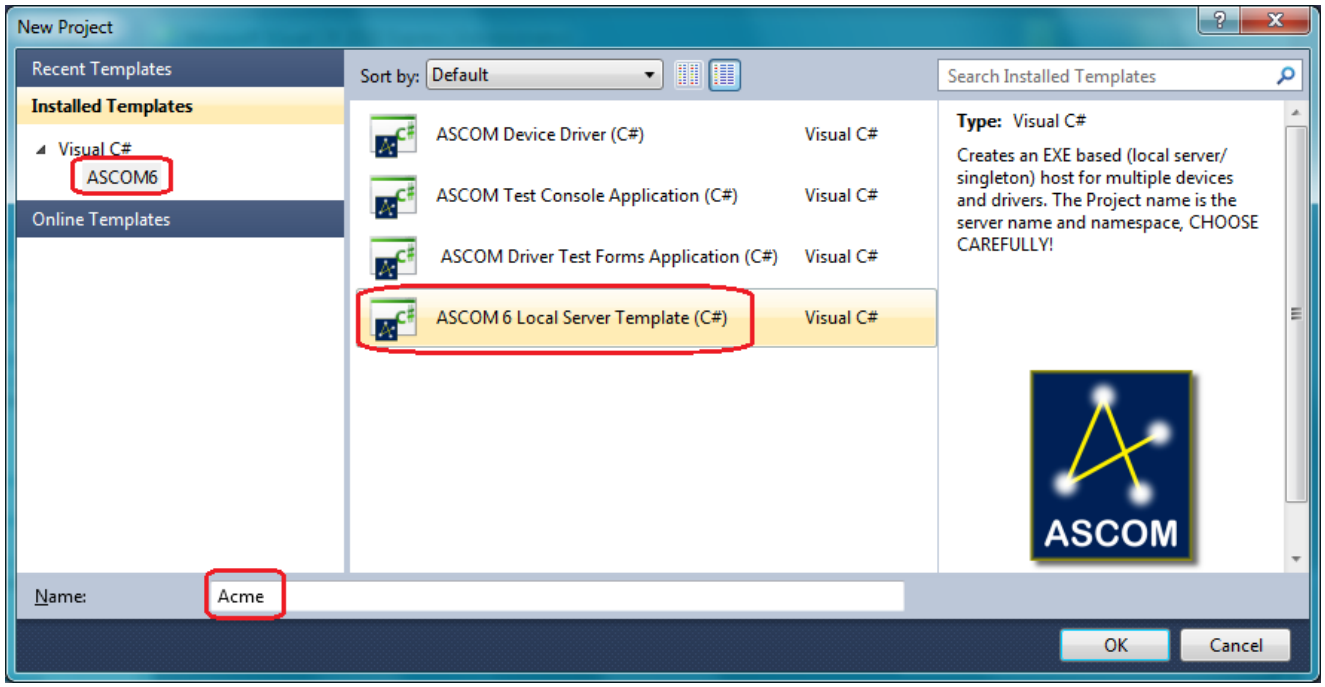
  - Enable only 32-bit code generation for the FilterWheel driver by modifying (outside of the Visual Basic IDE environment) the <PlatformTarget> tag in the FilterWheel driver's FilterWheel.csproj file to be **x86** located under the following tags: (it was "AnyCPU")

    <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
    <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
    (i.e. in FilterWheel.csproj: <PlatformTarget>**x86**</PlatformTarget>)

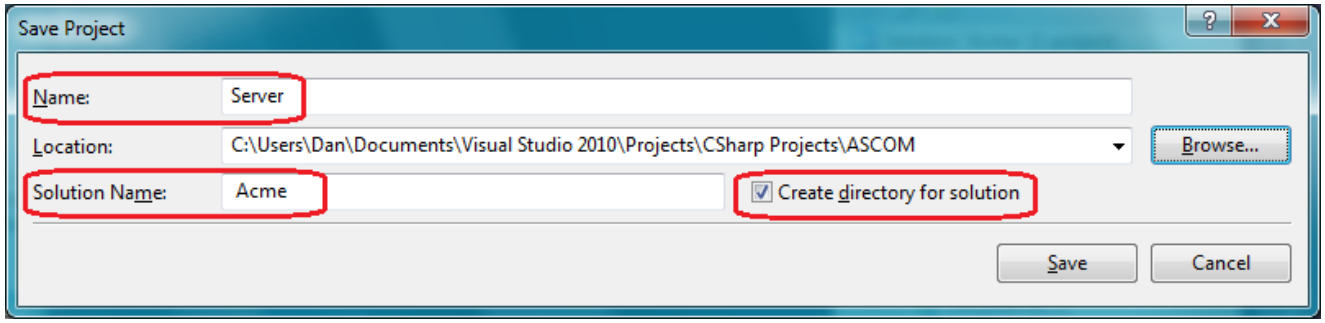
ASCOM C# LocalServer Development Framework Walk-Through

Visual C# 2010 Express ← Run as administrator !!!

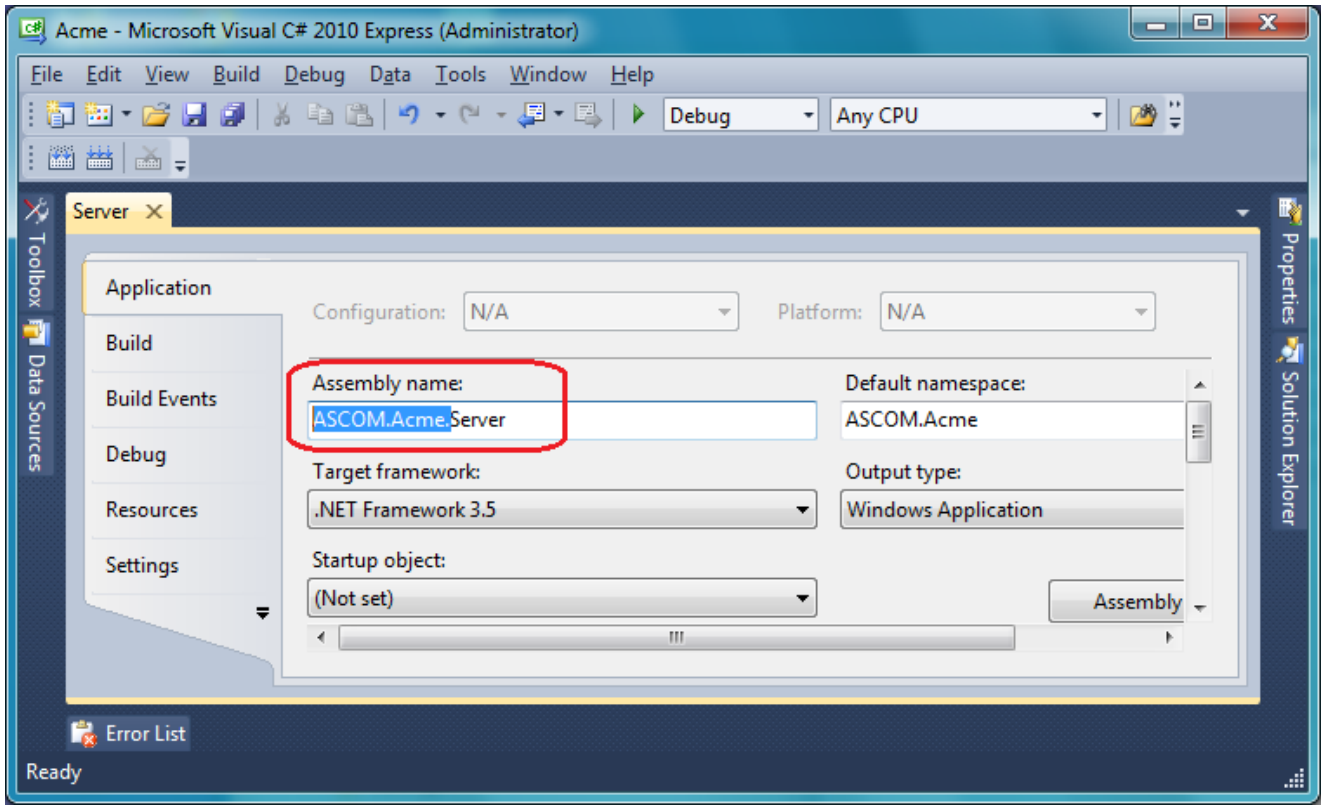
1) Start with New Project - **ASCOM 6 Local Server Template (C#)**
   - Name: **Acme**



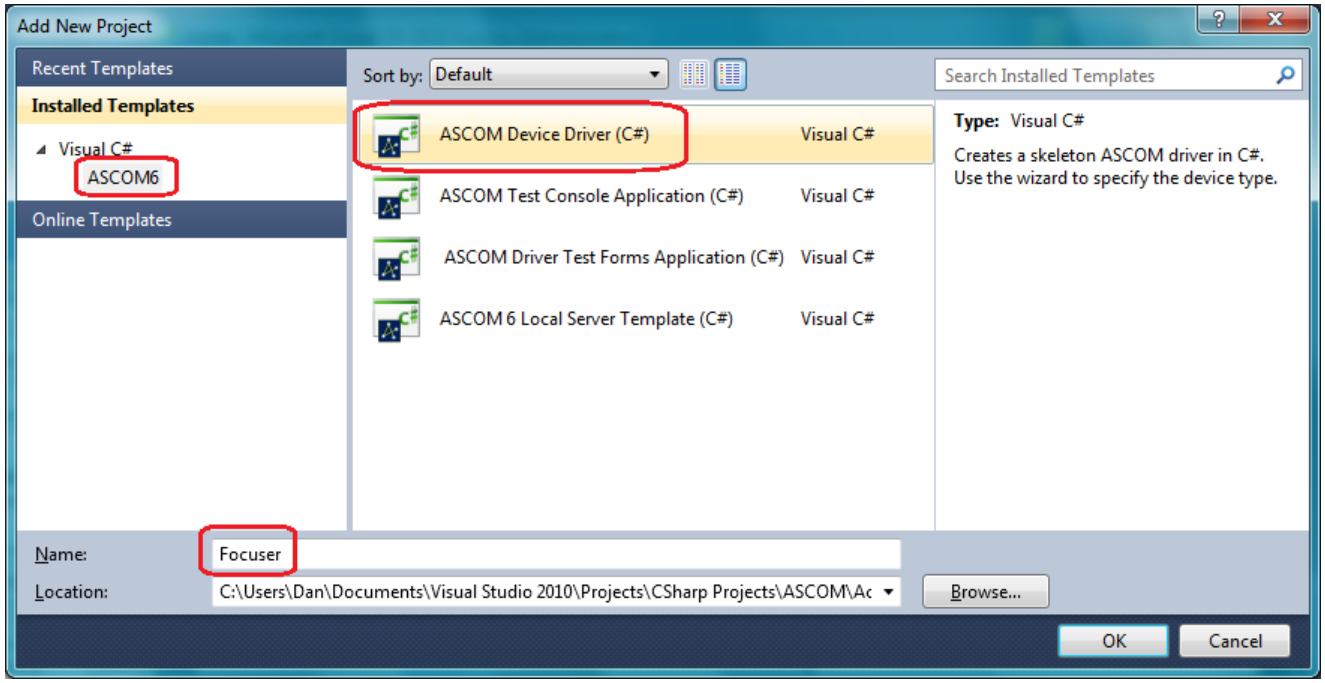
2) File➢Save All –
   - Name: **Server**
   - Location: ...**\CSharp Projects\ASCOM**
   - Solution Name: **Acme** ← **MAKE SURE**!!! because Visual Studio makes same as Name!!!
   - [✓] Create directory for solution



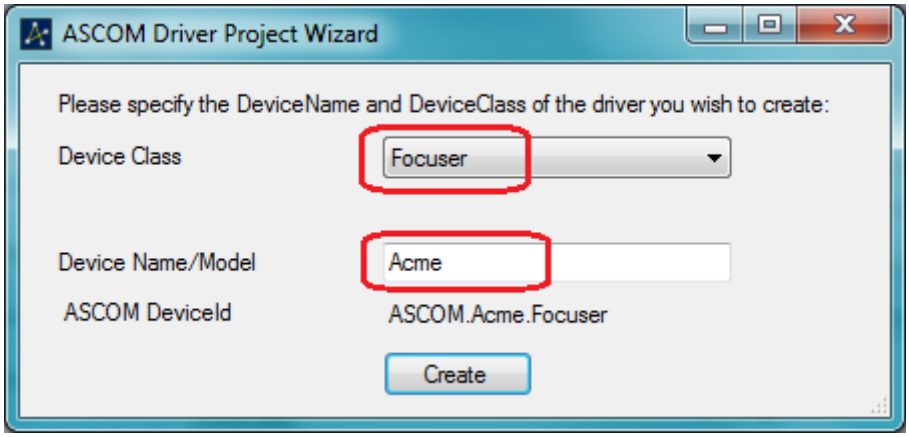
3) Set Server Properties➢Application➢Assembly name: **ASCOM.Acme.Server**
(The template wizard does not include full assembly name - done here for consistency.)



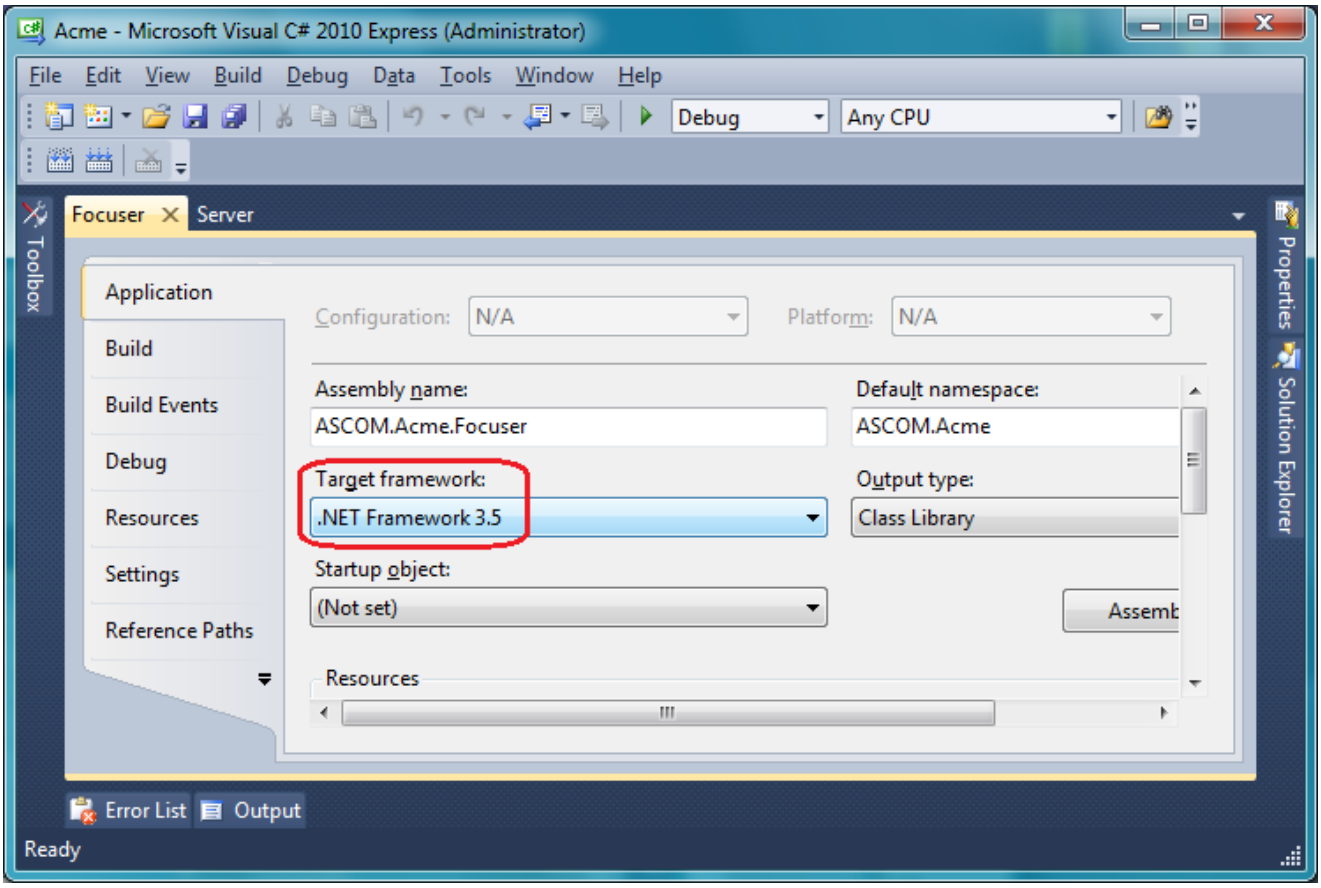
4) Add New Project for Focuser driver (File➢Add➢New Project... not available in C# Express)
- Right-Click solution name in Solution Explorer➢Add➢New Project...
(if Solution not shown in Solution Explorer, do Tools➢Options➢Projects and Solutions➢General, [✓] Always show solution)
- **ASCOM Device Driver (C#)**
  - Name: **Focuser**



5) ASCOM Driver Project Wizard
- Device Class - **Focuser**
- Device Name/Model - **Acme**
- Create

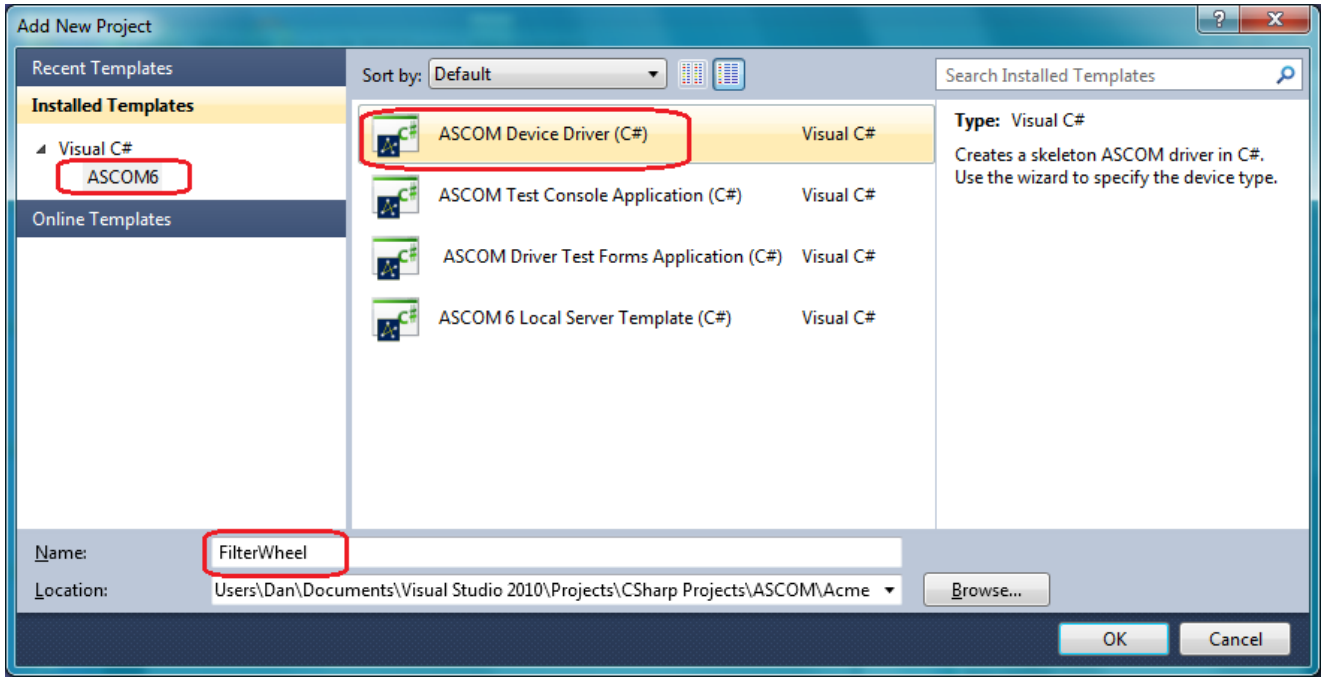


6) Set Focuser Properties➢Application➢Target framework: **.NET Framework 3.5**
(The driver must use the same .NET Framework as the server.)

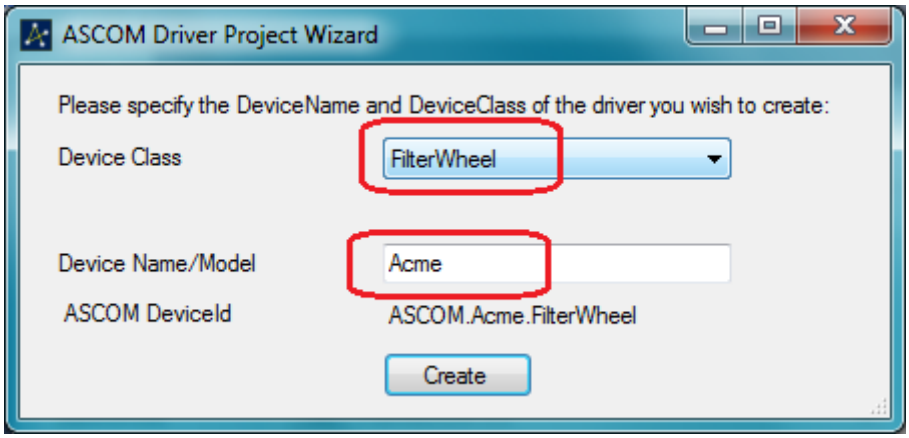


7) In Solution Explorer for Focuser, select and delete References➢ASCOM.Utilities.Video since it requires .NET 4.0 and the current configuration is for .NET 3.5

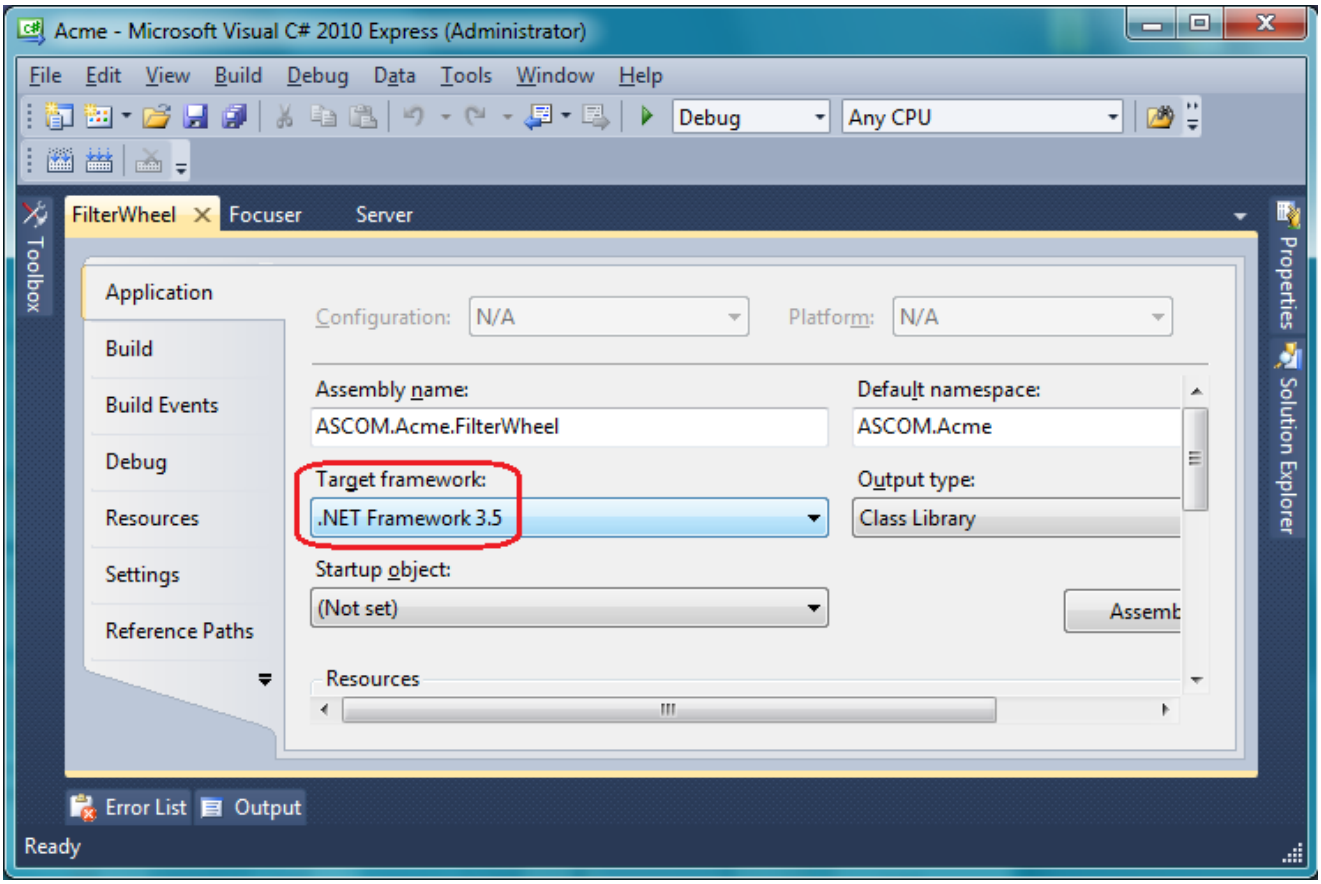
8) Add New Project for FilterWheel driver (File➢Add➢New Project... not available in C# Express)
- Right-Click solution name in Solution Explorer➢Add➢New Project...
(if Solution not shown in Solution Explorer, do Tools➢Options➢Projects and Solutions➢General, [✓] Always show solution)
- **ASCOM Device Driver (C#)**
  - Name: **FilterWheel**



9) ASCOM Driver Project Wizard
- Device Class - **FilterWheel**
- Device Name/Model - **Acme**
- Create



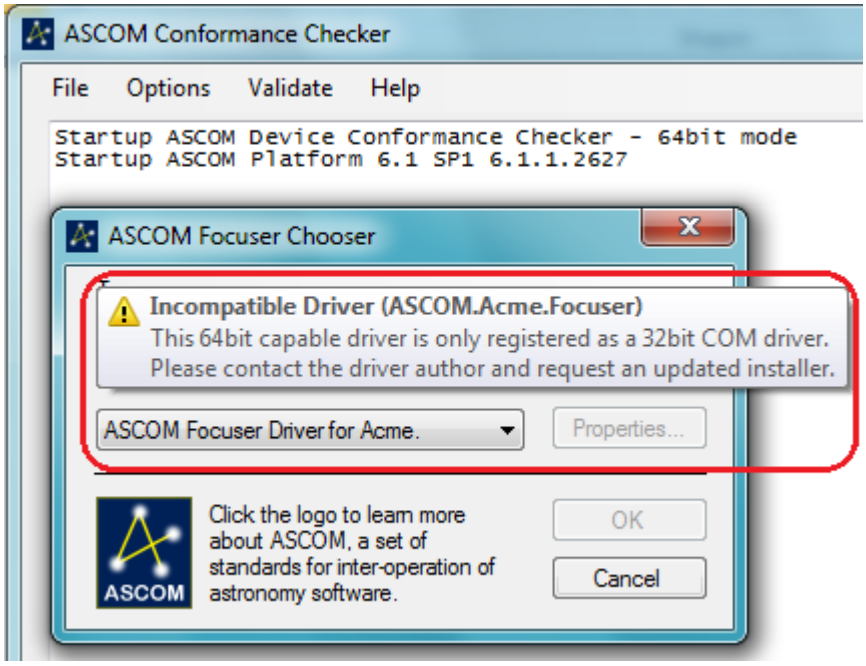
10) Set FilterWheelProperties➢Application➢Target framework: **.NET Framework 3.5**
(The driver must use the same .NET Framework as the server.)

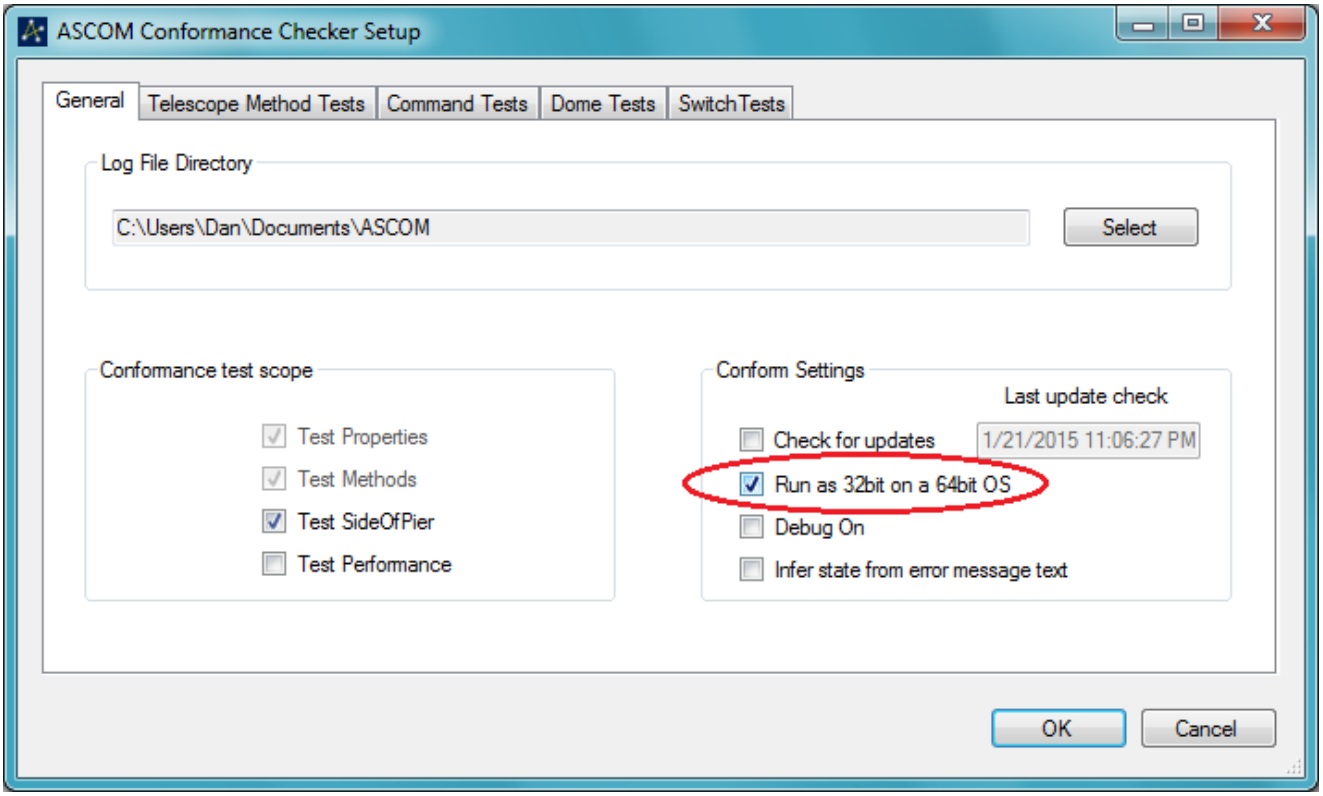


11) In Solution Explorer for FilterWheel, select and delete References➢ASCOM.Utilities.Video since it requires .NET 4.0 and the current configuration is for .NET 3.5

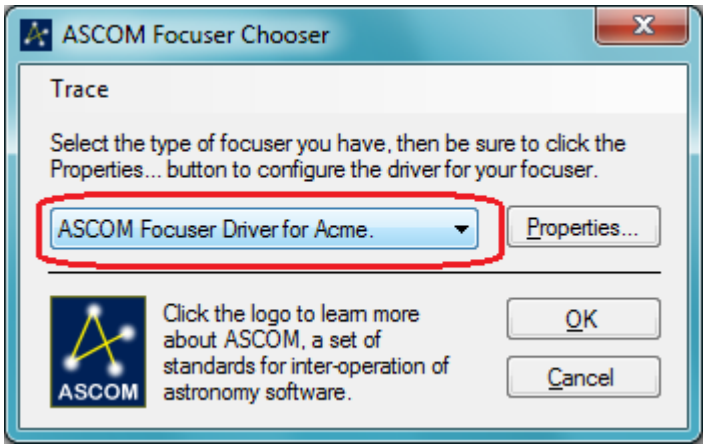
12) Build the solution (F6)

13) Run the ASCOM **Conform** tool. If it is running in 64 bit mode, change it to run in 32 bit mode with:
Options➢Conformance Options➢General➢Conform Settings: [✓] Run as 32bit on a 64bit OS
(this is needed because Visual Studio's *Register for COM interop* only registers the drivers as a 32bit COM driver, but not also as a 64bit COM driver on a 64-bit machine - as would be done by the Inno Setup installer)

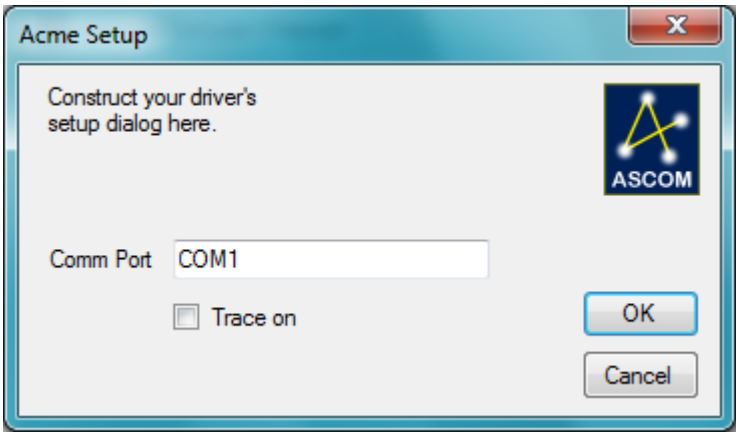




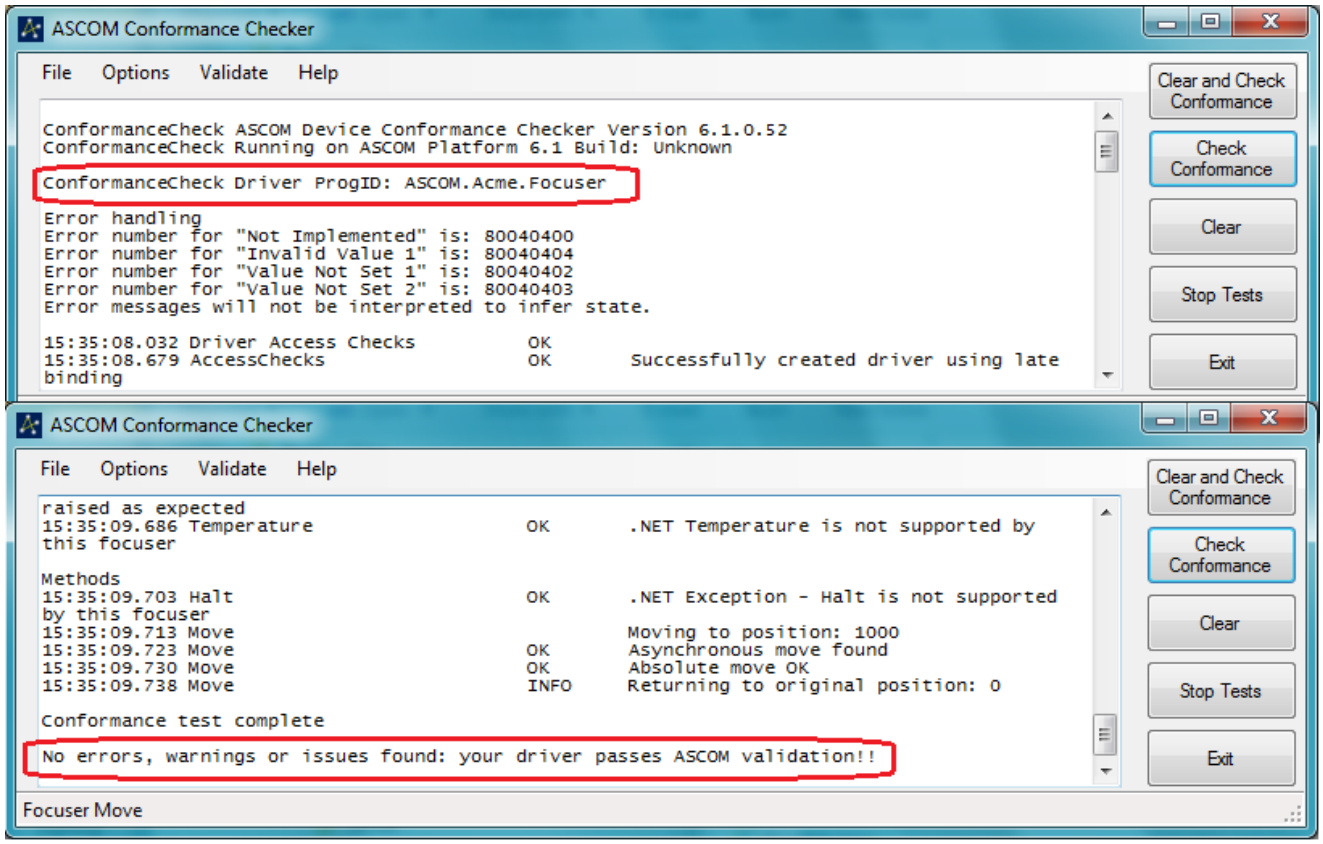
14) Using the ASCOM **Conform** tool, Options➢Check Focuser, Options➢Select Driver, select the *ASCOM Focuser Driver for Acme*.



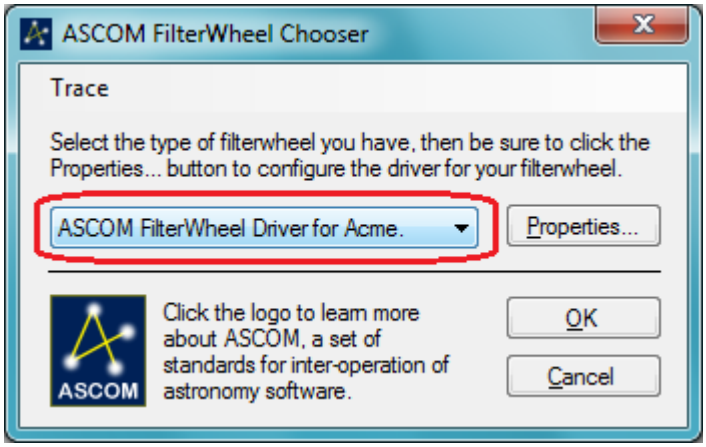
15) Select ASCOM Focuser Chooser➢Properties... to get the Acme Setup dialog for the Focuser



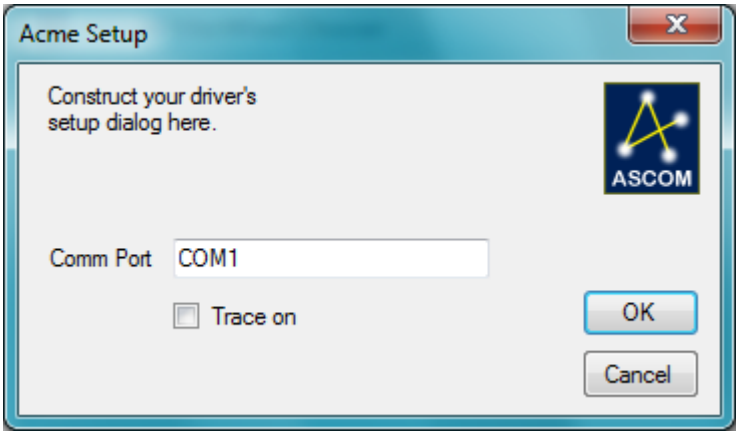
16) Run the Check Conformance and verify that no errors, warnings or issues are found and the Focuser driver passes ASCOM validation!!



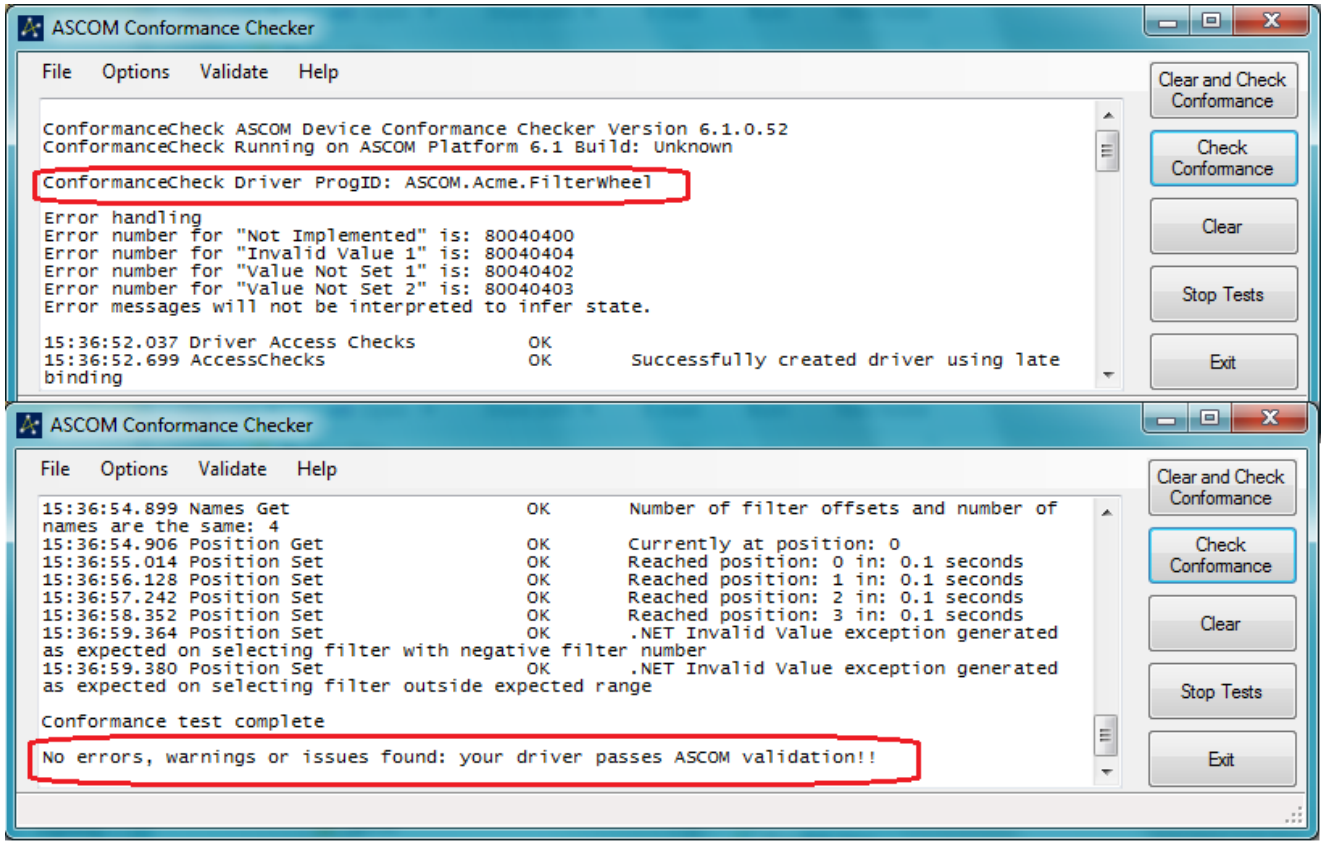
17) Using the ASCOM **Conform** tool, Options➢Check Filter Wheel, Options➢Select Driver, select the *ASCOM FilterWheel Driver for Acme*.



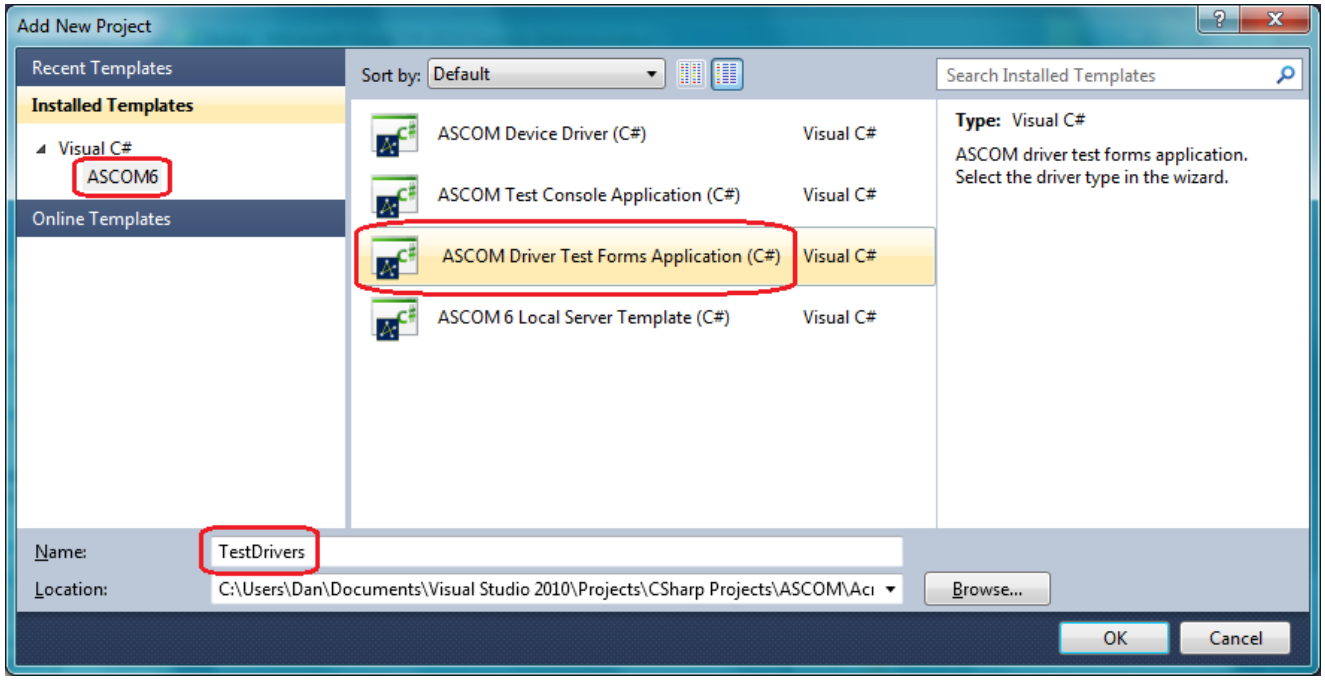
18) Select ASCOM FilterWheel Chooser➢Properties... to get the Acme Setup dialog for the FilterWheel



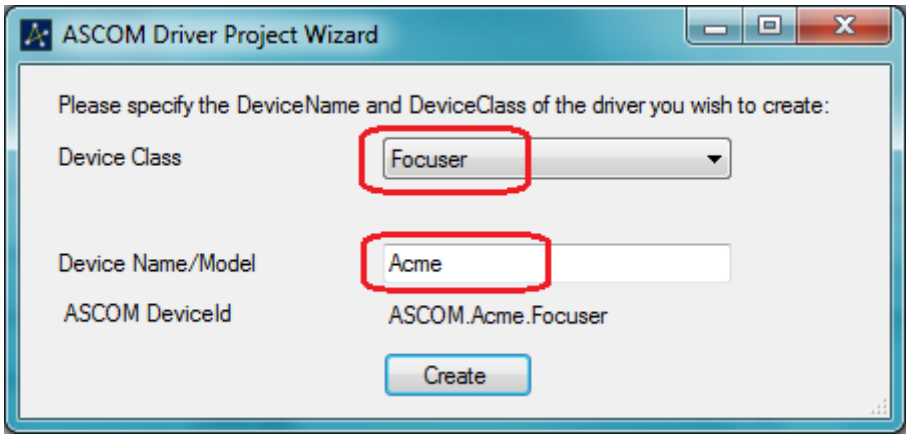
19) Run the Check Conformance and verify that no errors, warnings or issues are found and the FilterWheel driver passes ASCOM validation!!



20) Add New Project for application for testing the drivers
   - Right-Click solution name in Solution Explorer➢Add➢New Project...
   - **ASCOM Driver Test Forms Application (C#)**
   - Name: **TestDrivers**



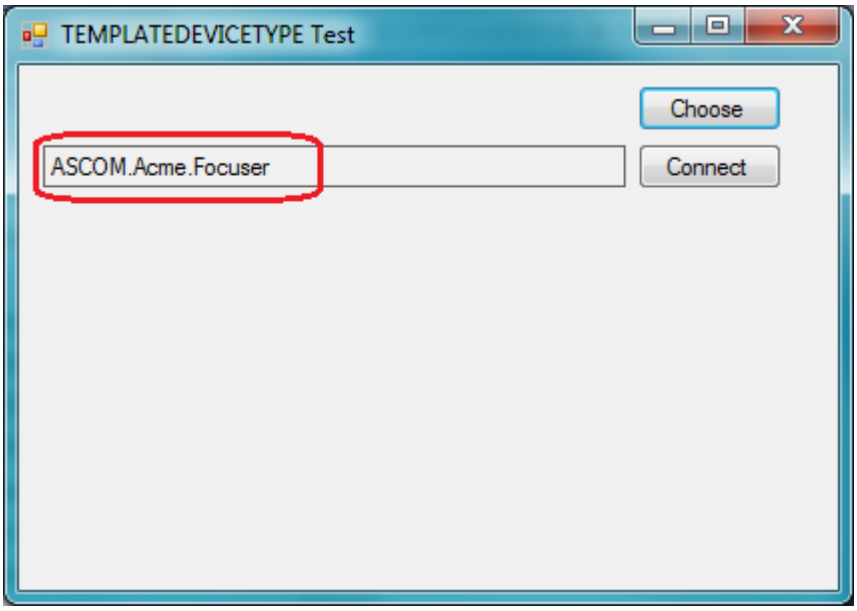
21) ASCOM Driver Project Wizard
   - Device Class - **Focuser**
   - Device Name/Model - **Acme**
   - Create



22) Right-Click Solution Explorer➢TestDrivers project➢Set as Startup Project to set the **TestDrivers** project as the startup project

23) Build the solution (F6)

24) Run the code (F5), click the test form's Choose button, select the *ASCOM Focuser Driver for Acme*., select ASCOM Focuser Chooser➢Properties... to get the Acme Setup dialog for the Focuser, OK those dialogs and verify the ASCOM.Acme.Focuser is shown on the test form.



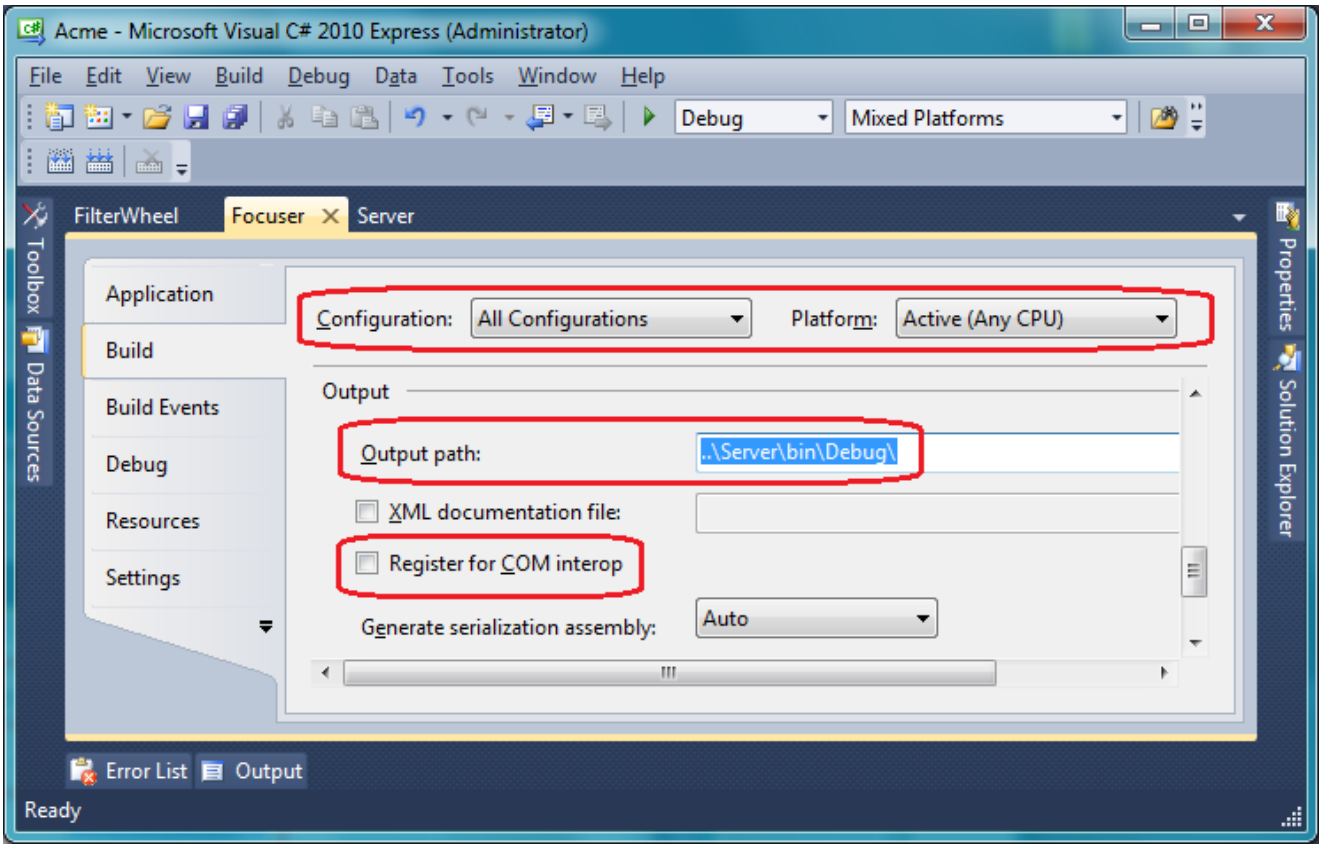
At this point, additional code can be added to the separate Focuser and FilterWheel drivers to independently control the Focuser and FilterWheel hardware and additional code and controls can be added to the TestDrivers project to exercise and debug the features of the in-proc DLL Focuser and FilterWheel drivers.

25) Clean the solution with Build➢Clean Solution so that the driver will be automatically unregistered from COM and ASCOM
(if menu Build➢Clean Solution is not shown, use Tools➢Customize➢Commands➢Menu bar: Build➢Add Command...➢Categories: Build, Commands: Clean Solution, OK, Close to add that menu item)

At this point, the Acme Focuser and FilterWheel should no longer be available in **Conform**'s Select Driver.

Now, make the changes to incorporate the **LocalServer** functionality.

26) In Focuser Properties➢Build➢
    - Set Configuration: **All Configurations**, Platform: **Active (Any CPU)**
    - Set Output➢Output path: **..\Server\bin\Debug** or use **..\Server\bin\Debug**
    - Disable Output➢**[ ] Register for COM interop**



27) Right-Click Focuser project➢Add Reference...➢Projects➢Server to add a reference to the Server project to the Focuser Project

28) Add the following class to the Focuser project's Driver.cs file just before the Focuser class definition:

```
internal class FocuserLocalServerConstants
{
    internal const string DRIVER_ID = "ASCOM.Acme.Focuser";
    internal const string DRIVER_DESCRIPTION = "Acme Focuser";
}
```

This provides a single instance of Focuser constants to decorate the Focuser class and for use inside the Focuser class following the DRY principle (Don't Repeat Yourself - ref. Tim Long).

29) Add the following attribute declarations to the Focuser project's Driver.cs Focuser class definition:

```
[ProgId(FocuserLocalServerConstants.DRIVER_ID)]
[ServedClassName(FocuserLocalServerConstants.DRIVER_DESCRIPTION)]
```

(The server uses this to identify this driver as a driver to be served.)

30) Change the Focuser project's Driver.cs Focuser class definition to inherit *ReferenceCountedObjectBase*:

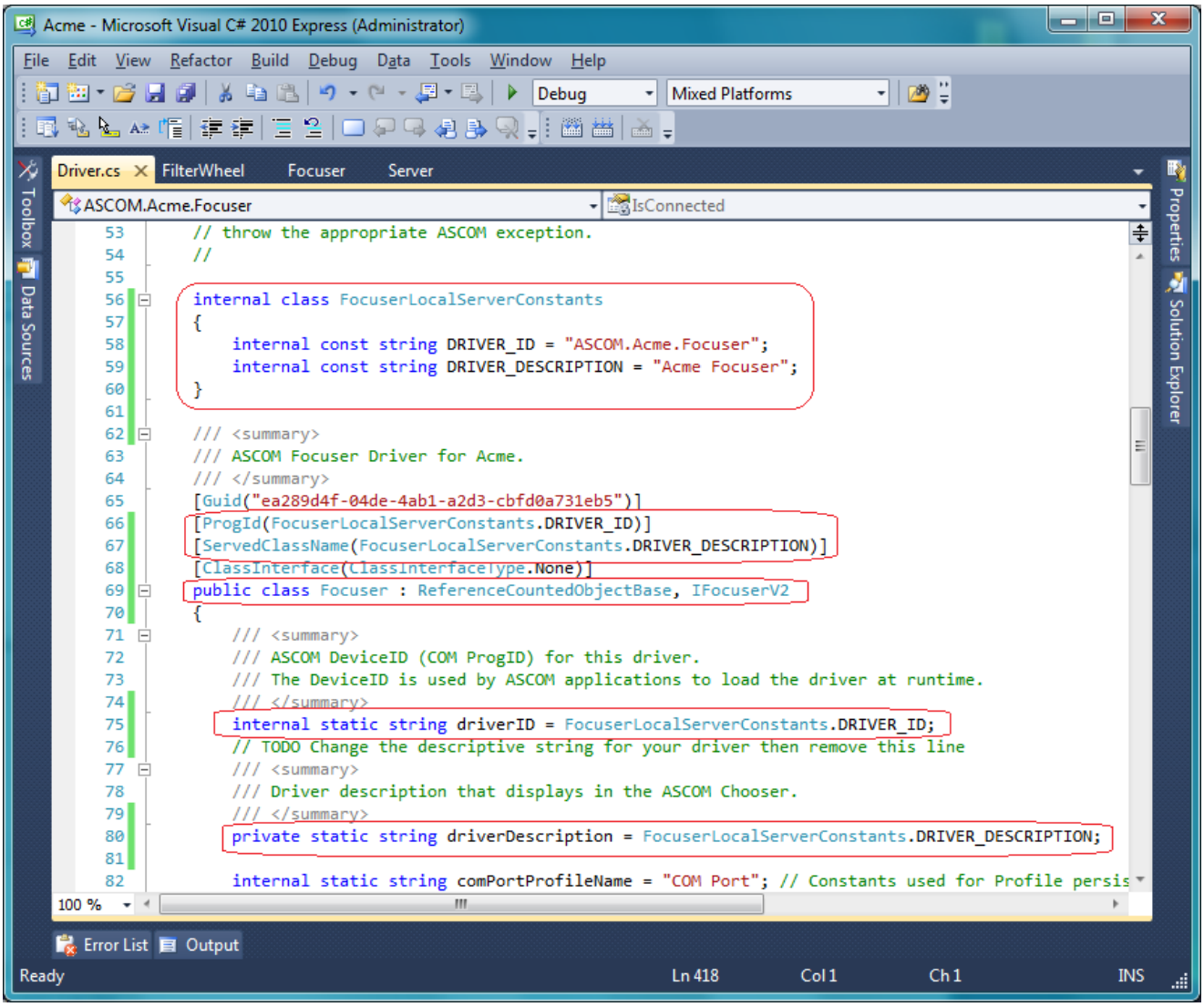
```
public class Focuser : ReferenceCountedObjectBase, IFocuserV2
```

31) Change the Focuser project's Driver.cs *driverID* definition to:

```
internal static string driverID = FocuserLocalServerConstants.DRIVER_ID;
```

32) Change the Focuser project's Driver.cs *driverDescription* definition to:

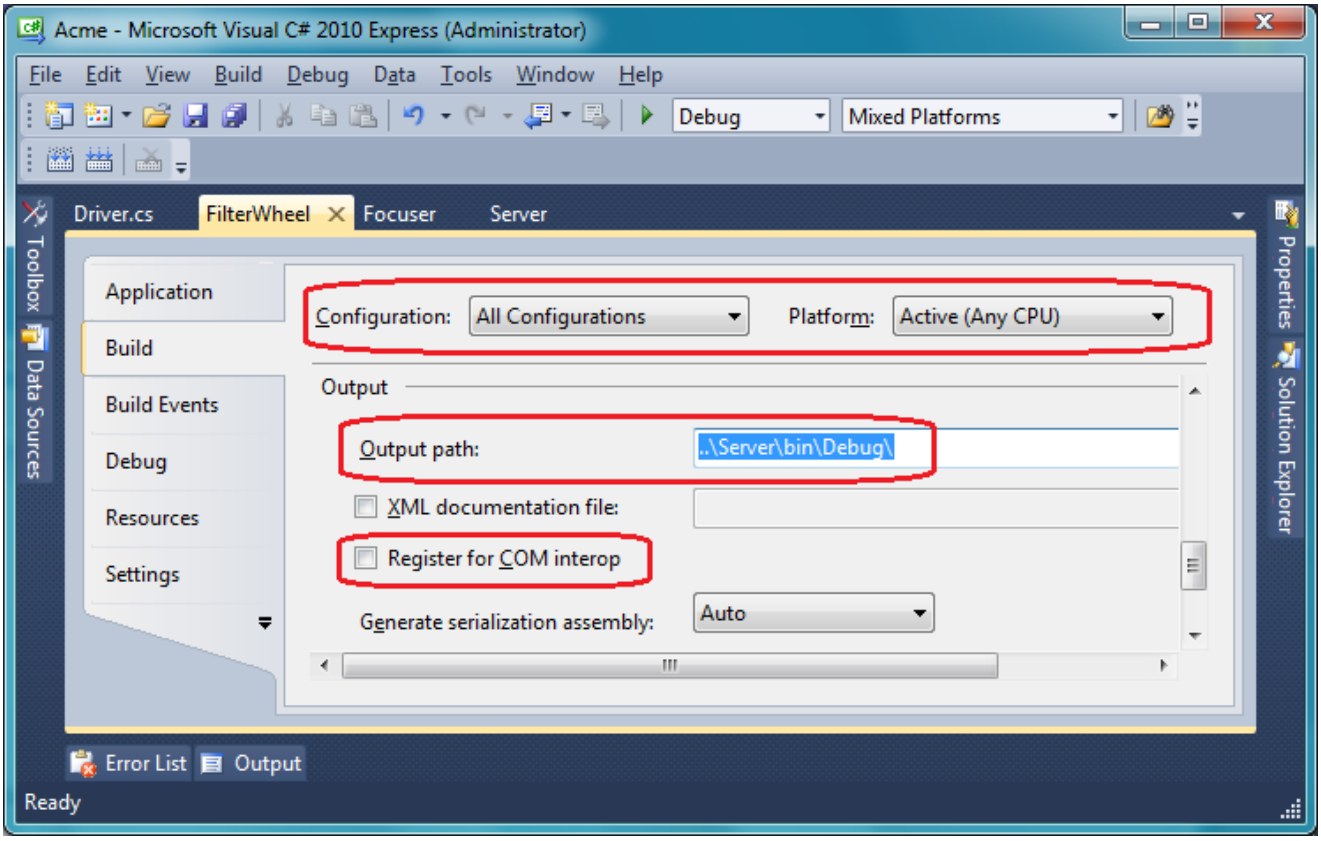
```
private static string driverDescription = FocuserLocalServerConstants.DRIVER_DESCRIPTION;
```



33) Remove the Focuser project's Driver.cs ASCOM registration region code

This completes the basic changes to the Focuser driver to be able to be served by an ASCOM LocalServer.

34) In FilterWheel Properties➢Build➢
   - Set Configuration: **All Configurations**, Platform: **Active (Any CPU)**
   - Set Output➢Output path: **..\Server\bin\Debug** or use **..\Server\bin\Debug**
   - Disable Output➢**[ ] Register for COM interop**



35) Right-Click FilterWheel Project➢Add Reference...➢Projects➢Server to add a reference to the Server project to the FilterWheel project

36) Add the following class to the FilterWheel project's Driver.cs file just before the FilterWheel class definition:

```
internal class FilterWheelLocalServerConstants
{
    internal const string DRIVER_ID = "ASCOM.Acme.FilterWheel";
    internal const string DRIVER_DESCRIPTION = "Acme FilterWheel";
}
```

This provides a single instance of FilterWheel constants to decorate the FilterWheel class and for use inside the FilterWheel class following the DRY principle (Don't Repeat Yourself - ref. Tim Long).

37) Add the following attribute declarations to the FilterWheel project's Driver.cs FilterWheel class definition:

```
[ProgId(FilterWheelLocalServerConstants.DRIVER_ID)]
[ServedClassName(FilterWheelLocalServerConstants.DRIVER_DESCRIPTION)]
```

(The server uses this to identify this driver as a driver to be served.)

38) Change the FilterWheel project's Driver.cs FilterWheel class definition to inherit *ReferenceCountedObjectBase*:

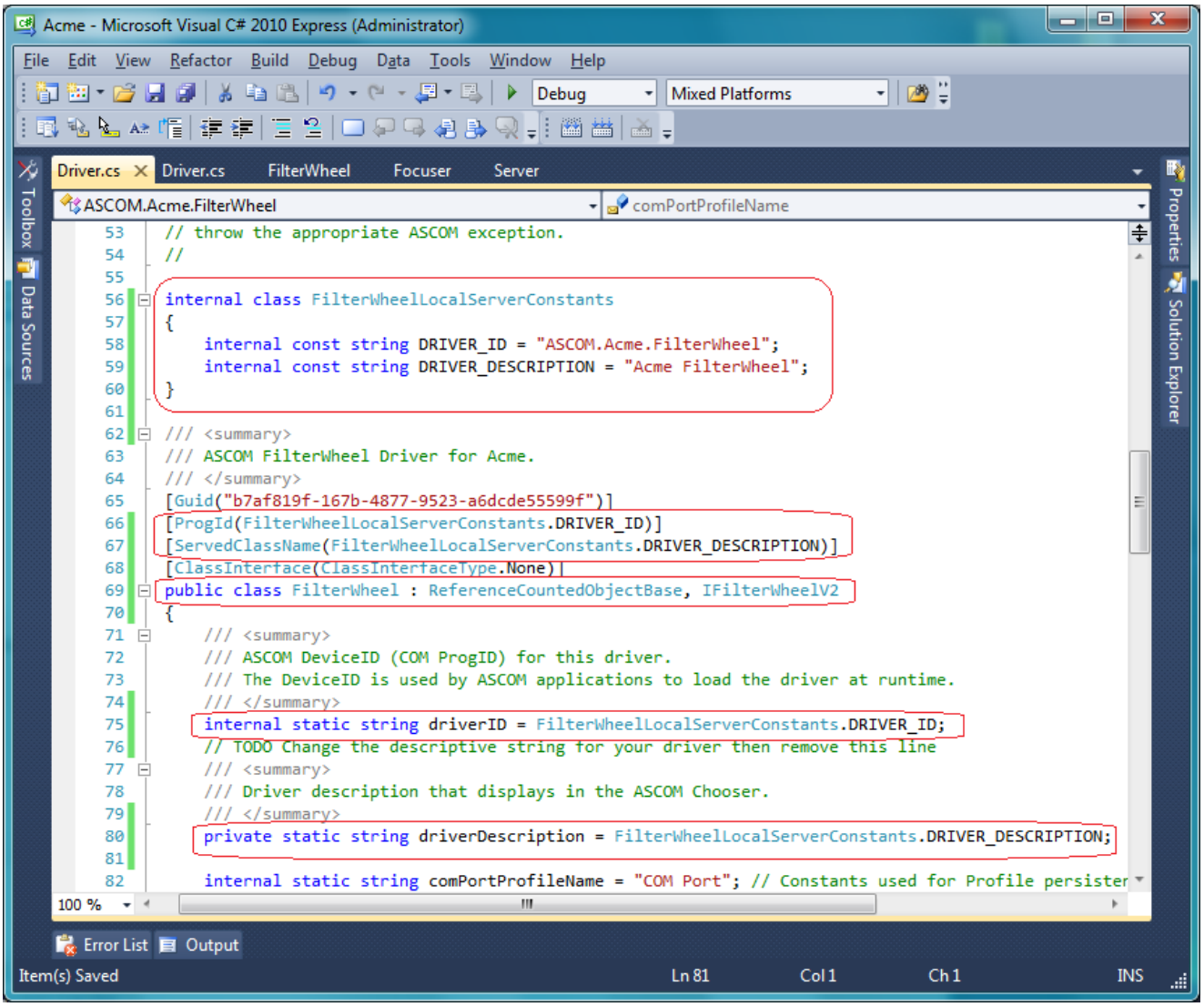
```
public class FilterWheel: ReferenceCountedObjectBase, IFilterWheelV2
```

39) Change the FilterWheel project's Driver.cs *driverID* definition to:

```
internal static string driverID = FilterWheelLocalServerConstants.DRIVER_ID;
```

40) Change the FilterWheel project's Driver.cs *driverDescription* definition to:

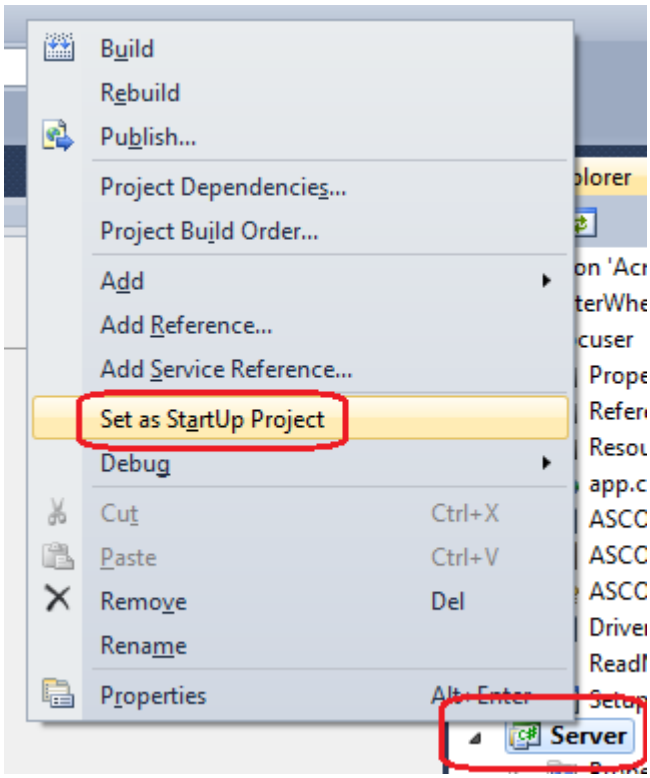
```
private static string driverDescription =
                                        FilterWheelLocalServerConstants.DRIVER_DESCRIPTION;
```



41) Remove the FilterWheel project's Driver.cs ASCOM registration region code

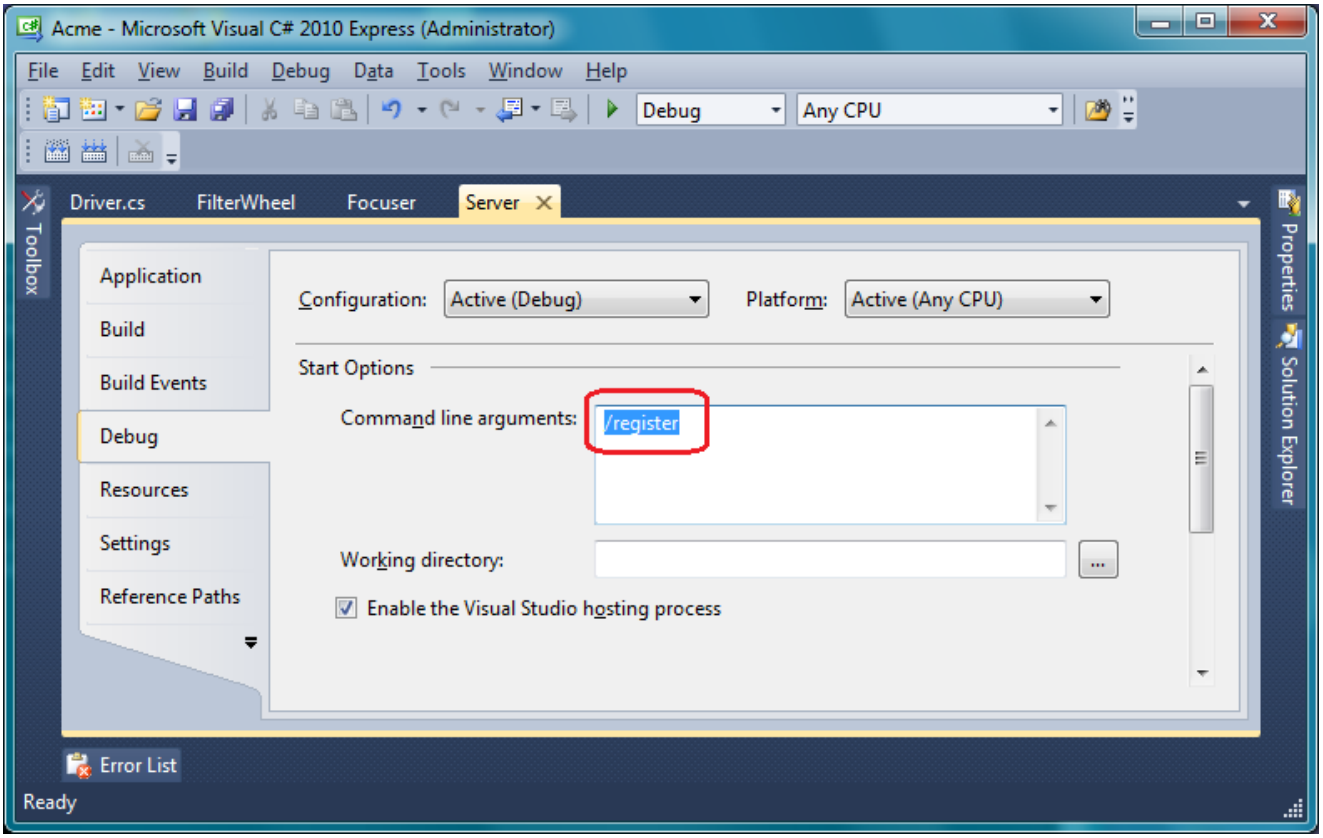
This completes the basic changes to the FilterWheel driver to be able to be served by an ASCOM LocalServer.

42) Right-Click Solution Explorer➢Server project➢Set as Startup Project to set the local server as the startup project



43) Build the solution (ignore 2 mismatch warnings for now, see NOTES)

44) Add Server Properties➢Debug➢Start Options➢Command line arguments: **/register**



45) Run the project (to have the local server register the drivers with COM and ASCOM)
(this registers the drivers for both 32 bit and 64 bit {on a 64-bit machine}, so the ASCOM Conform tool can now be run as 64 bits without problems)

46) Using the ASCOM Conform tool, Options➢Check Focuser, Options➢Select Driver, select the Acme Focuser

47) Select ASCOM Focuser Chooser➢Properties... to get the Acme Setup dialog for the Focuser

48) Run the Check Conformance and verify that no errors, warnings or issues are found and the ASCOM.Acme.Focuser driver passes ASCOM validation!!

49) Using the ASCOM Conform tool, Options➢Check Filter Wheel, Options➢Select Driver, select the Acme FilterWheel

50) Select ASCOM FilterWheel Chooser➢Properties... to get the Acme Setup dialog for the FilterWheel

51) Run the Check Conformance and verify that no errors, warnings or issues are found and the ASCOM.Acme.FilterWheel driver passes ASCOM validation!!

52) Change Server Properties➢Debug➢Start Options➢Command line arguments: **/unregister**

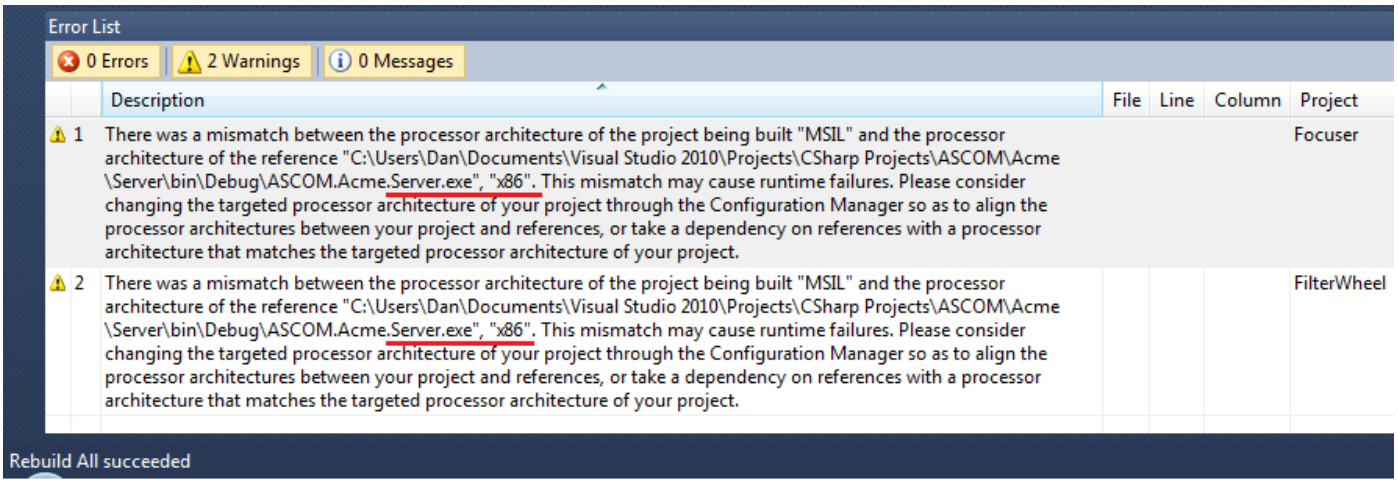
53) Run the project (to have the local server unregister the drivers with COM and ASCOM)

At this point, code can be changed in the Focuser and FilterWheel drivers to appropriately work with the common Focuser and FilterWheel hardware and additional code and controls can be added to the TestDrivers project to exercise and debug the features of the LocalServer-served Focuser and FilterWheel drivers.

When the ASCOM local server and drivers development is complete, the ASCOM Driver Install Script Generator an be used to generate an Inno Setup script to generate a Windows setup executable that can be used to distribute the server and drivers just developed.
Note: The server/driver solution should be closed in the IDE before running the Inno Setup compiler.

NOTES:

- The following warning occurs for both the Focuser driver and the FilterWheel driver when building the solution:
  "There was a mismatch between the processor architecture of the project being built "MSIL" and the processor architecture of the reference "...\Server\bin\Debug\ASCOM.Acme.Server.exe", "x86"."



These warnings occur because the drivers are built by default from the templates for "**AnyCPU**" while the server is built by default from the template for "**x86**". The server ***must*** be built for "**x86**" (a served driver fails to load when the server is built for "**AnyCPU**"), so the drivers need to also be built for "**x86**". A stand-alone in-proc driver will not work on a 64-bit OS unless it is built for "**AnyCPU**", but a LocalServer-served driver will work on a 64-bit (and 32-bit) OS when built for "**x86**").

To resolve these warnings: (the easier way described, Configuration Manager can also be used)

- Enable only 32-bit code generation for the Focuser driver by modifying (outside of the Visual Basic IDE environment) the <PlatformTarget> tag in the Focuser driver's Focuser.csproj file to be **x86** located under the following tags: (it was "AnyCPU")
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
  (i.e. in Focuser.csprog: <PlatformTarget>**x86**</PlatformTarget>)
- Enable only 32-bit code generation for the FilterWheel driver by modifying (outside of the Visual Basic IDE environment) the <PlatformTarget> tag in the FilterWheel driver's FilterWheel.csproj file to be **x86** located under the following tags: (it was "AnyCPU")
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
  (i.e. in FilterWheel.csproj: <PlatformTarget>**x86**</PlatformTarget>)