Create a C# Template for AutoCAD using Visual Studio 2022

The goal of this tutorial is to show how to create a template for starting a new project in C# for AutoCAD in Visual Studio 2022, allowing to automatically launch AutoCAD by loading the DLL from Visual Studio in Debug mode.

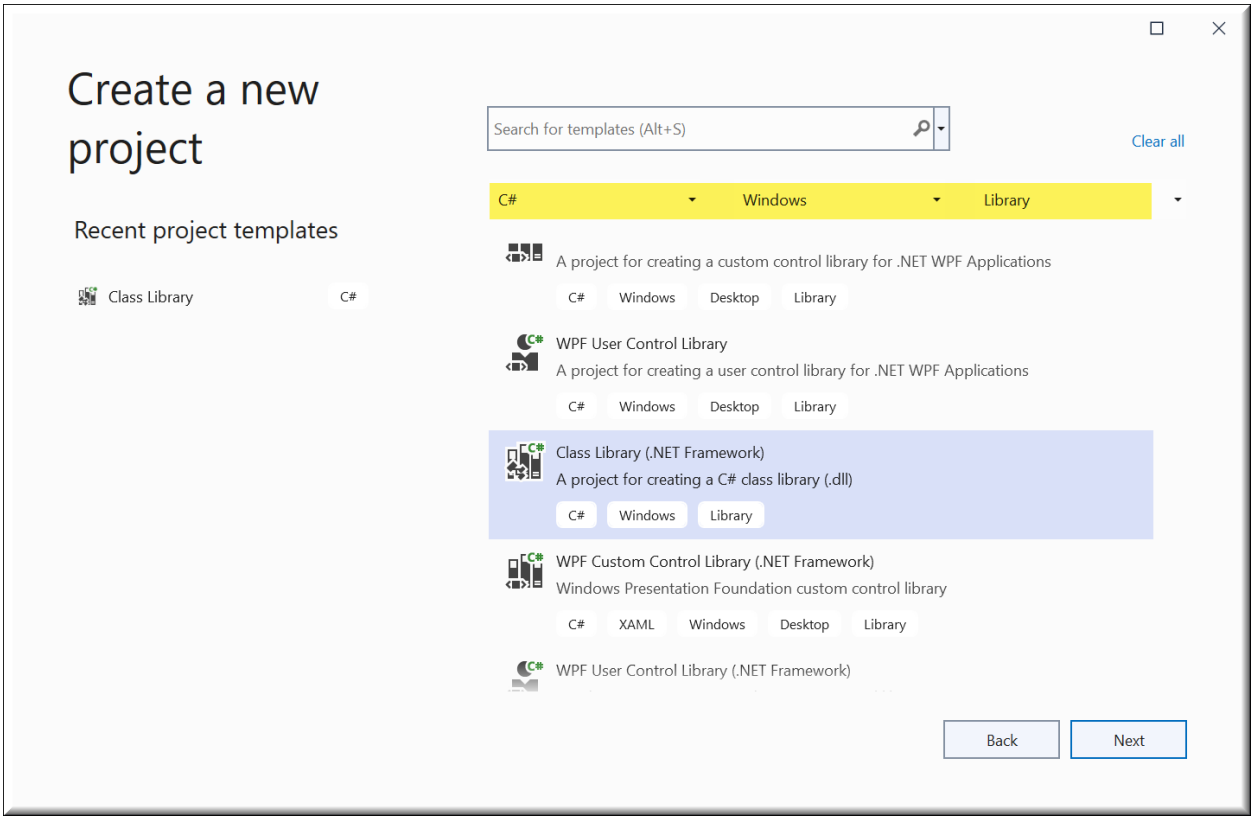
The example shown uses Visual Studio Community 2022 but it is easily transferable to other versions.

This tutorial will create a model for AutoCAD 2024, but there again, it is possible to target other versions of AutoCAD.

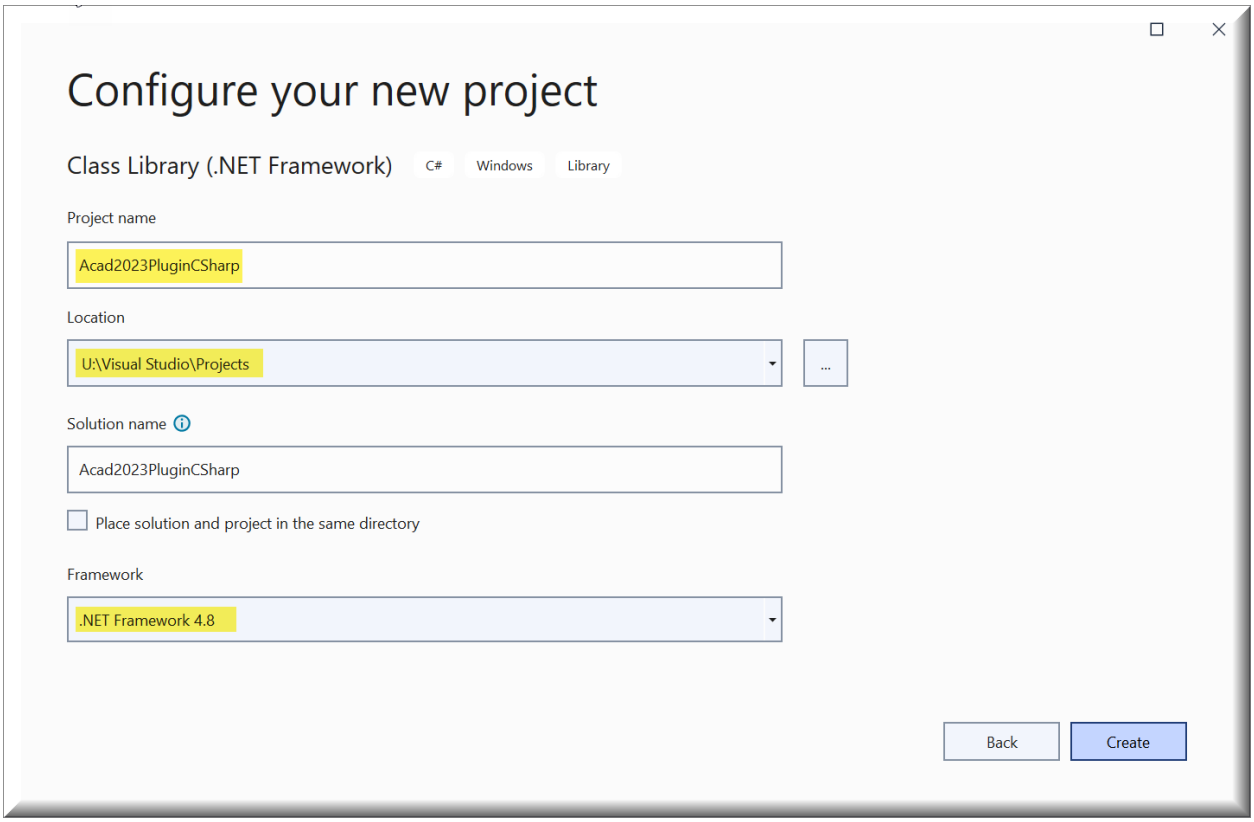
The path to the output directory for the Debug mode used in the example is the default (.\bin\Debug).

Start a new project

In Visual Studio, start a new project (CTRL+Shift+N), select the C# language and the type of project: **Class Library (.NET Framework)**. Be sure not to select just **Class Library**. It will only allow you to target .NET 5.0 and above. Then click **Next**.



Rename the project: Acad2024PluginCSharp for example, and change the path to the location as desired.

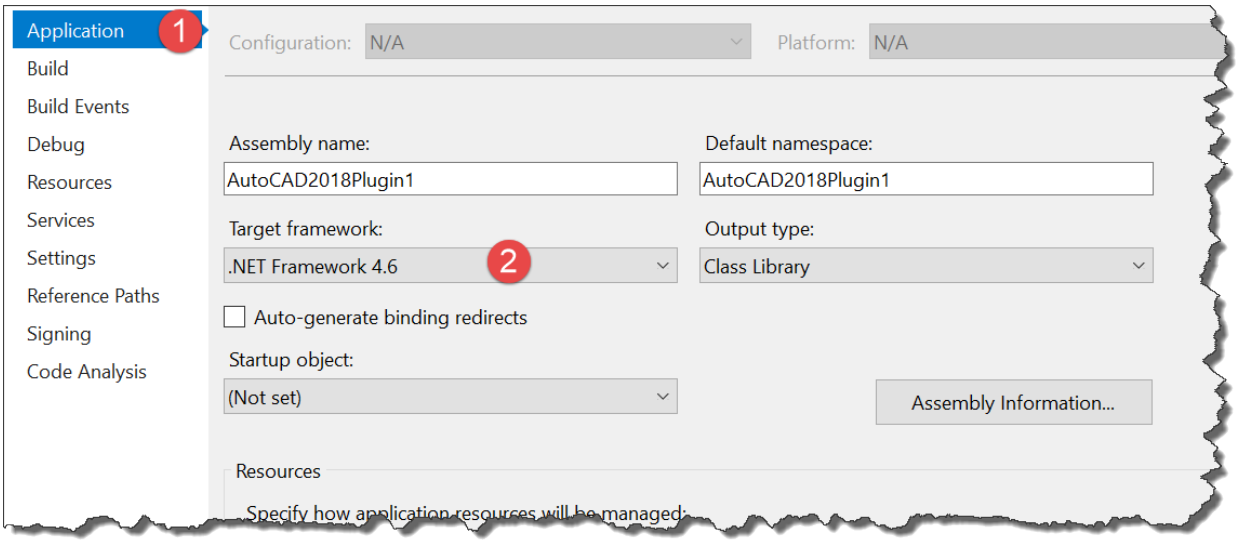


Specify the target Framework

As shown above, in the new project dialog, select the target Framework drop-down list. For 2024, this will be 4.8. For other versions, see the table below.

AutoCAD Version	2019	2020	2021	2022	2023	2024
Release	R23.0	R23.1	R24.0	R24.1	R24.2	R24.3
DWG Format	AC1032	AC1032	AC1032	AC1032	AC1032	AC1032
Installed .NET Framework	4.7	4.7	4.8	4.8	4.8	4.8
Visual Studio 2017 (15.0)						
Visual Studio 2019 (16.0)						
Visual Studio 2022 (17.2)						

To set the target framework after creating the project, go to the Solution Explorer (Ctrl+Alt+L) and right click on the project name. Select Properties. 1. Click on the Application tab of the project's Properties window. 2. Set the Target Framework and save the file.



Add AutoCAD libraries

In Solution Explorer, select then right click References and then select **add a reference...**

In the **Reference Manager** dialog, click **Browse...**

References to add are in the version targeted for AutoCAD installation directory, or better, in the ObjectARX2024 folder \#inc corresponding to the version of AutoCAD targeted (Inc.-win32 or inc - x 64 versions of AutoCAD and platform the station used). When you use the ObjectArx files, added symbol definitions are loaded into memory when you are debugging. This provides information to Visual Studio to assist in debugging.

Choose commonly used libraries (it will be easy to add others in the same way if the creation of the new project requires them). At minimum, you need the following:

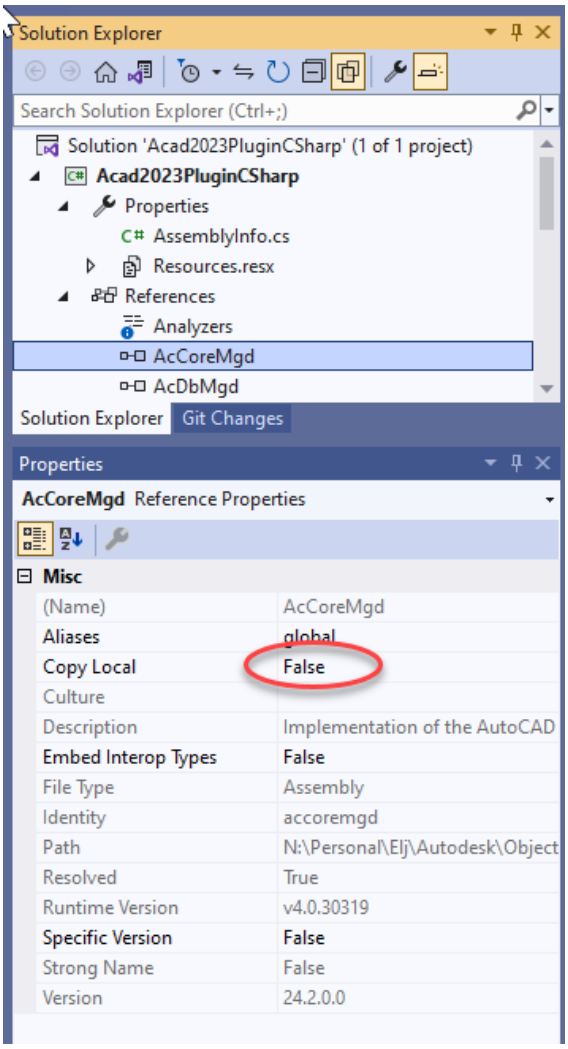
- AutoCAD 2007 to 2012: AcDbMgd.dll and AcMgd.dll
- AutoCAD-2013 to 2024: AcCoreMgd.dll, AcDbMgd.dll and AcMgd.dll

Change the **Copy Local** property of these dll's to **False**. By setting to False, the versions of the dll's that come with AutoCAD are used instead of the debugging versions.

You can download the latest versions of the ObjectARX SDK from the link below. The download link is at the bottom of the page. After you fill out the license form, it will take you to a page where you can select the ObjectArx version corresponding to the version of AutoCAD you are targeting.

https://www.autodesk.com/developer-network/platform-technologies/autocad/objectarx

The default location for the files to extract is "C:\Autodesk\ObjectARX_for_AutoCAD_2024_Win_64bit_dlm\", but you can put them anywhere. Typically, I put them where I store my development files and rename the folder to ".\ObjectArx2024".



Add some draft code

You can add whatever class files (*.cs) you want to your template, depending on how you want to organize your code. The Autodesk AutoCAD .Net Add-in Wizard creates a template that has two class files, MyCommands.cs and MyPlugin.cs. This organizes code into two logical blocks, an area for project specific code and an area for your commands. Some prefer to add a third class for utility functions. As long as you use the same namespace in each class, the plugin will be able to find everything. Let's start with MyPlugin.cs. It just implements IExtensionApplication and two void methods `IExtensionApplication.Initialize()` and `IExtensionApplication.Terminate()`.

Below is a sample class file that can execute code when your plugin loads or terminates. This is the MyPlugin.cs file that the Autodesk® tool creates. The Autodesk® copyright permits distributing the code. I've copied it here because it explains how to use the class.

```
//// (C) Copyright 2002-2009 by Autodesk, Inc.
//
// Permission to use, copy, modify, and distribute this software in
// object code form for any purpose and without fee is hereby granted,
```

```
// provided that the above copyright notice appears in all copies and
// that both that copyright notice and the limited warranty and
// restricted rights notice below appear in all supporting
// documentation.
//
// AUTODESK PROVIDES THIS PROGRAM "AS IS" AND WITH ALL FAULTS.
// AUTODESK SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTY OF
// MERCHANTABILITY OR FITNESS FOR A PARTICULAR USE.  AUTODESK, INC.
// DOES NOT WARRANT THAT THE OPERATION OF THE PROGRAM WILL BE
// UNINTERRUPTED OR ERROR FREE.
//
// Use, duplication, or disclosure by the U.S. Government is subject to
// restrictions set forth in FAR 52.227-19 (Commercial Computer
// Software - Restricted Rights) and DFAR 252.227-7013(c)(1)(ii)
// (Rights in Technical Data and Computer Software), as applicable.
//
using System;
using Autodesk.AutoCAD.ApplicationServices;
using Autodesk.AutoCAD.DatabaseServices;
using Autodesk.AutoCAD.EditorInput;
using Autodesk.AutoCAD.Geometry;
using Autodesk.AutoCAD.Runtime;

// This line is not mandatory, but improves loading performances
[assembly: ExtensionApplication(typeof(Acad2024PluginCSharp.MyPlugin))]

namespace Acad2024PluginCSharp
{

    /// This class is instantiated by AutoCAD once and kept alive for the
    /// duration of the session. If you don't do any one time initialization
    /// then you should remove this class.

    public class MyPlugin : IExtensionApplication
    {

        void IExtensionApplication.Initialize()
        {
            // Add one time initialization here
            // One common scenario is to setup a callback function here that
            // unmanaged code can call.
            // To do this:
            // 1. Export a function from unmanaged code that takes a function
            //    pointer and stores the passed in value in a global variable.
            // 2. Call this exported function in this function passing delegate.
            // 3. When unmanaged code needs the services of this managed module
            //    you simply call acrxLoadApp() and by the time acrxLoadApp
            //    returns  global function pointer is initialized to point to
            //    the C# delegate.
            // For more info see:
            // http://msdn2.microsoft.com/en-US/library/5zwkzwf4(VS.80).aspx
            // http://msdn2.microsoft.com/en-us/library/44ey4b32(VS.80).aspx
            // http://msdn2.microsoft.com/en-US/library/7esfatk4.aspx
            // as well as some of the existing AutoCAD managed apps.

            // Initialize your plug-in application here
        }

        void IExtensionApplication.Terminate()
```

```
        {
            // Do plug-in application clean up here
        }

    }

}
```

To create your commands class, rename the class created by Visual Studio, **Class1,** with a more meaningful name, for example, **Commands**, from the solution Explorer. A dialog box offers to rename all references to **Class1** in the project, answer **Yes**.

Open the class **Commands** in the code editor by double click on **Commands** in Solution Explorer.

Add the `using` statements to import the most widely used AutoCAD namespaces and, possibly, an alias for the Autodesk.AutoCAD.ApplicationServices.Application class. By including an alias, it eliminates ambiguity with System.Windows.Application. Below is a sample class file for your commands.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Autodesk.AutoCAD.ApplicationServices;
using Autodesk.AutoCAD.DatabaseServices;
using Autodesk.AutoCAD.EditorInput;
using Autodesk.AutoCAD.Geometry;
using Autodesk.AutoCAD.Runtime;
using AcAp = Autodesk.AutoCAD.ApplicationServices.Application;

// This line is not mandatory, but improves loading performance
[assembly: CommandClass(typeof(Acad2024PluginCSharp.Commands))]

namespace Acad2024PluginCSharp
{
    public class Commands
    {
        [CommandMethod("TEST")]
        public void Test()
        {
            var doc = AcAp.DocumentManager.MdiActiveDocument;
            var db = doc.Database;
            var ed = doc.Editor;
            using (var tr = db.TransactionManager.StartTransaction())
            {
                tr.Commit();
            }
        }
    }
}
```

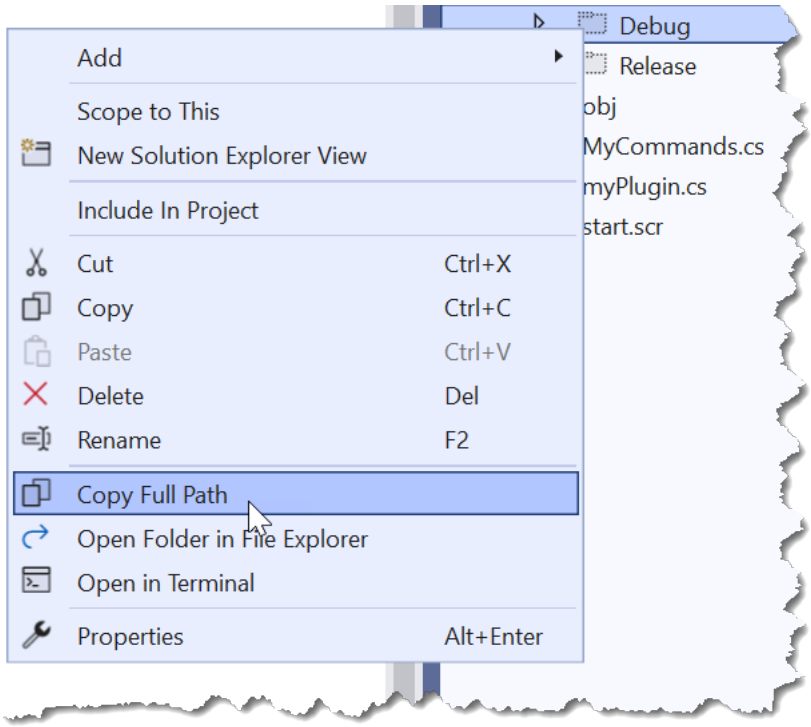
Build the solution, making sure that there are no errors (F6).

Add a script to load the application at startup of AutoCAD

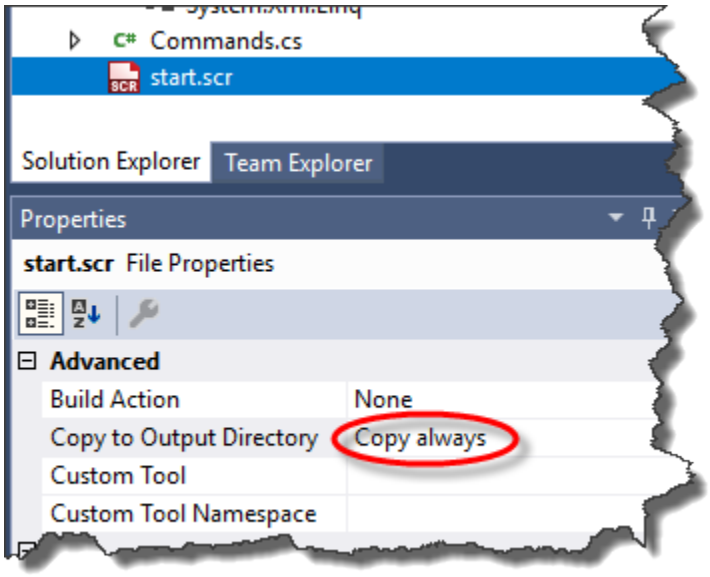
To make debugging easier, we'll add an AutoCAD script file that executes the NETLOAD command for the dll we're debugging. When you distribute your dll, you should include it in a bundle. Kean Walmsley describes it here and here. Don't distribute the script file.

From Solution Explorer, right click on the project Acad2024PluginCSharp and then **Add** and **New Item**... (Ctrl + Shift + A).
In the box to add a new item dialog select: text file. Rename the file: **start.scr** and click Add.
The file opens in the Editor window, add the following statement **including a space** at the end.
netload "Acad2024PluginCSharp.dll"

Note: Since AutoCAD 2016, it is imperative that the LEGACYCODESEARCH system variable is 1 for AutoCAD integrates the output directory (.\Acad2024PluginCSharp \bin\Debug) in the search paths and so it will find the DLL and the script file. If you don't want to set this variable to 1 (ON), then you can manually add the path to the scr file. To do this, right click on **start.scr** in the Solution Explorer and select **Copy Full Path**. In the scr file, paste the full path into your script, in front of the dll name. You will have to do this again when you use the template to create a new project so that it points to the new solution. You will also need to the startup arguments shown on page 9.



Save the **start.scr** file. In the **start.scr** properties (select **start.scr** in Solution Explorer), set the property **Copy to Output Directory** to **Copy Always**.



Change the MSBuild project file to launch AutoCAD for debugging

(.Csproj) MSBuild project files are xml files that describe and control the process of generation of applications. It is in this file that should be added the instructions that allow to launch AutoCAD while Visual Studio is in debug mode and load the DLL to start AutoCAD.
Build the solution and close Visual Studio before you can edit the Acad2024PluginCSharp.csproj file.
Open the file **.\Acad2024PluginCSharp\Acad2024PluginCSharp.csproj** with a text editor (Notepad, notepad ++, etc.).
Add the following nodes at the end of the node PropertyGroup corresponding to the generation in debug (the second node PropertyGroup in the file), after changing, if necessary, the **acad.exe** file path corresponding to the version of AutoCAD, targeted and the file **start.scr** (the output for the Debug mode directory).

```
<StartAction>Program</StartAction>
<StartProgram>C:\Program Files\Autodesk\AutoCAD 2024\acad.exe</StartProgram>
<StartArguments>/nologo /b "start.scr"</StartArguments>
```

StartAction indicates that the generation of the project should start a program (which is usually not the case with a DLL).

StartProgram specifies the program to start. Here it is AutoCAD.

StartArguments contains the arguments that are going to run the script that will load the DLL.

The **PropertyGroup** node for Debug mode should look like this:

```
<PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
  <DebugSymbols>true</DebugSymbols>
  <DebugType>full</DebugType>
  <Optimize>false</Optimize>
  <OutputPath>bin\Debug\</OutputPath>
```

```
    <DefineConstants>DEBUG;TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
    <StartAction>Program</StartAction>
    <StartProgram>C:\Program Files\Autodesk\AutoCAD 2024\acad.exe</StartProgram>
    <StartArguments>/nologo /b "start.scr"</StartArguments>
  </PropertyGroup>
```

The **ItemGroup** nodes are added to the project references.

If the solution directory is on the same drive as the one where the referenced DLLs are (the directory of installation of AutoCAD or ObjectARX 20##), then the registered paths are relative.

As the project will be exported in a different directory from the solution, you need to replace these relative paths with absolute paths which point to where you installed ObjectARX, By default, this will be C:\ObjectARX2024.

For example, replace the following relative paths:

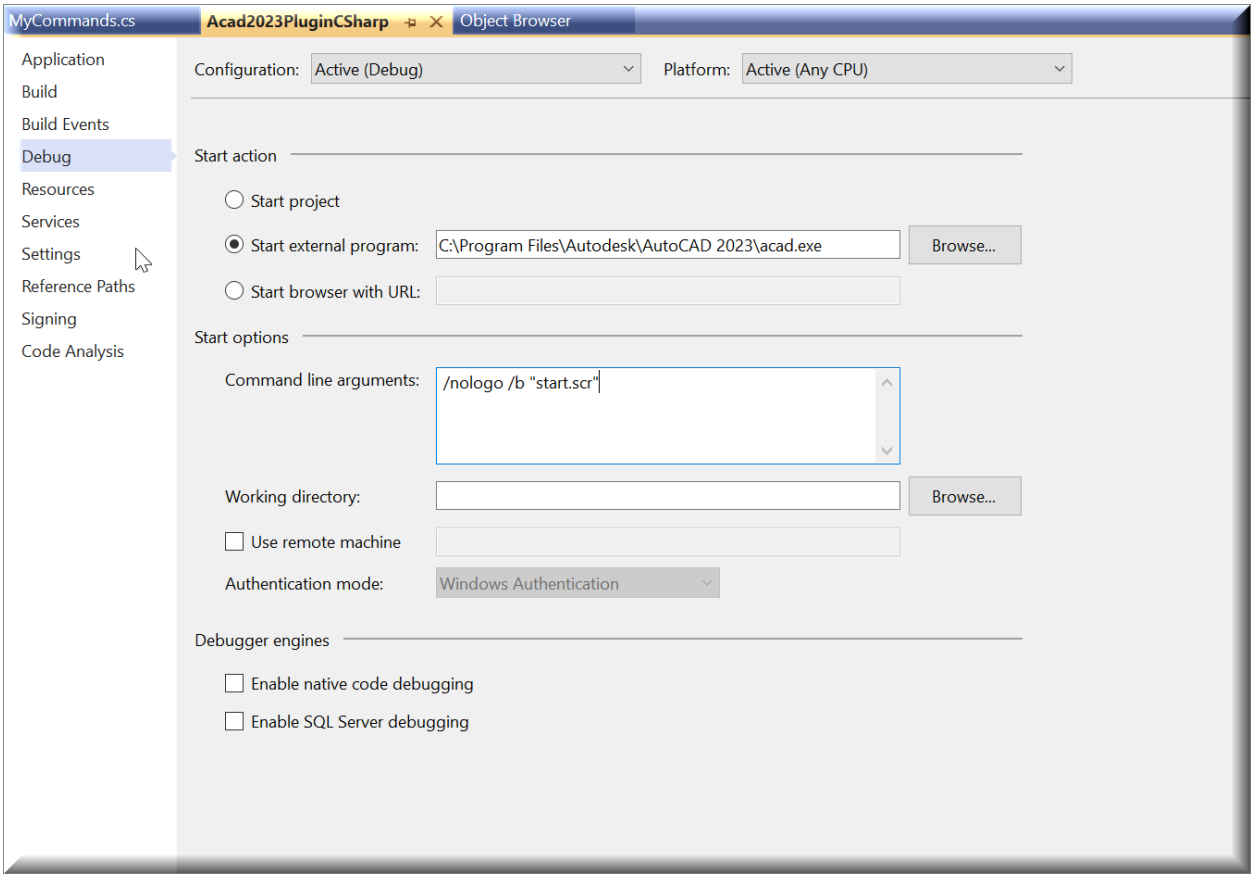
```
    <Reference Include="AcCoreMgd">
      <HintPath>..\..\..\..\..\..\..\ObjectARX 2024\inc\AcCoreMgd.dll</HintPath>
      <Private>False</Private>
    </Reference>
    <Reference Include="AcDbMgd">
      <HintPath>..\..\..\..\..\..\..\ObjectARX 2024\inc\AcDbMgd.dll</HintPath>
      <Private>False</Private>
    </Reference>
    <Reference Include="AcMgd">
      <HintPath>..\..\..\..\..\..\..\ObjectARX 2024\inc\AcMgd.dll</HintPath>
      <Private>False</Private>
    </Reference>
```

With this:

```
    <Reference Include="AcCoreMgd">
      <HintPath>C:\ObjectARX 2024\inc\AcCoreMgd.dll</HintPath>
      <Private>False</Private>
    </Reference>
    <Reference Include="AcDbMgd">
      <HintPath>C:\ObjectARX 2024\inc\AcDbMgd.dll</HintPath>
      <Private>False</Private>
    </Reference>
    <Reference Include="AcMgd">
      <HintPath>C:\ObjectARX 2024\inc\AcMgd.dll</HintPath>
      <Private>False</Private>
    </Reference>
```

Save the Acad2024PluginCSharp.csproj file.

These changes should appear in Visual Studio, in the **Debug** tab of the project properties. Newer versions of Visual Studio should allow you to edit the properties here instead of editing the project file. You may need to include the full path to the script as noted on page 7. But you don't need to do this in the template project. You can just do it in your final project that is built from this template.



Export the template

Reopen the solution Acad2024PluginCSharp.sln in Visual Studio.

Start debugging (F5) to start the operation. AutoCAD should open and display command line:

Command: netload Assembly file name: "Acad2024PluginCSharp.dll"

The below error message indicates either a wrong DLL name in the file **start.scr** or that the **start.scr** file has not been copied into the output directory (see copy in the output directory property).

Command: netload Assembly file name: " Acad2024PluginCSharp.dll" failed to load Assembly. " Error details: System.IO.FileNotFoundException: could not load file or assembly...

If everything works correctly, it's time to generate the template.

On the **Project** menu, choose **Export template...**

In the **Export Template** wizard dialog box, leave checked **Project template** and do next.

Change the name of the template as desired, AutoCAD 2024 Plugin C#, for example, and possibly add a description: **Template for AutoCAD 2024**.

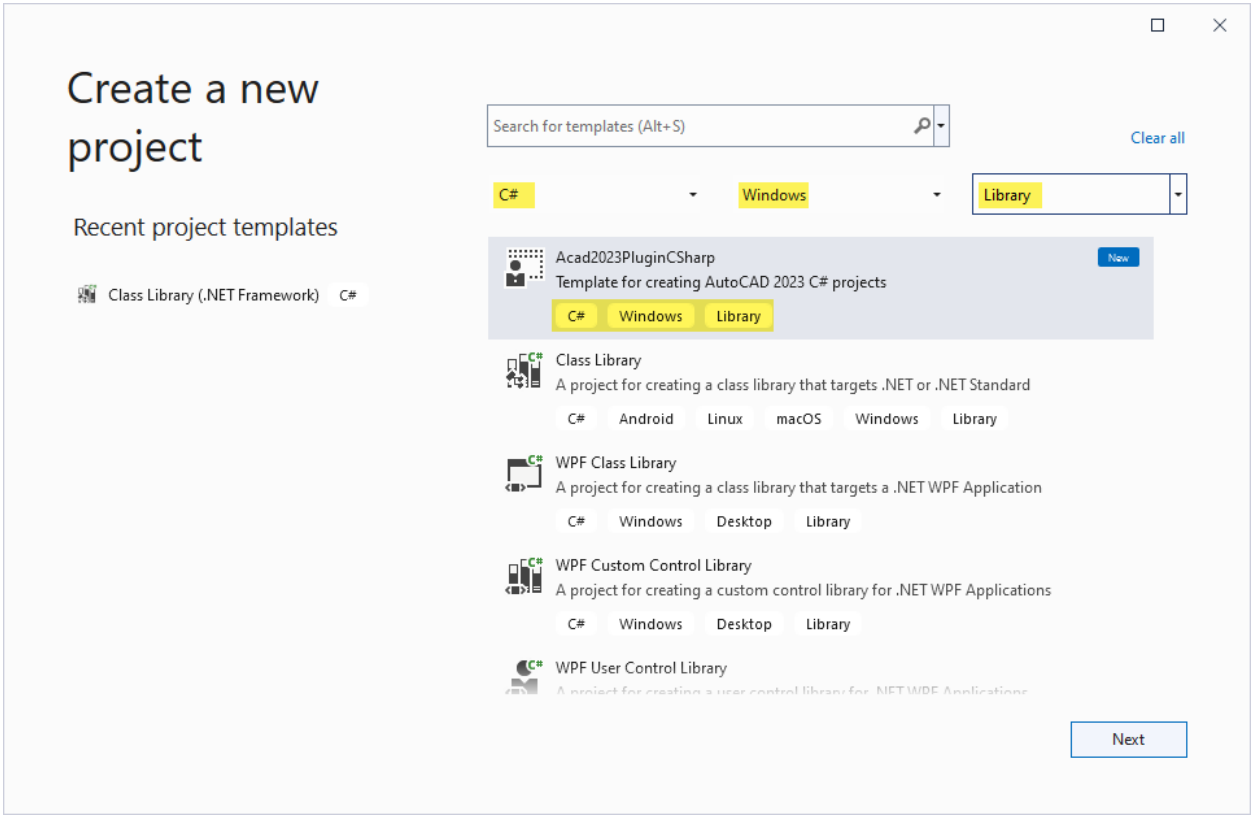
Click **Finish**.

With the default options, the model is exported in the form of a ZIP file in the files:

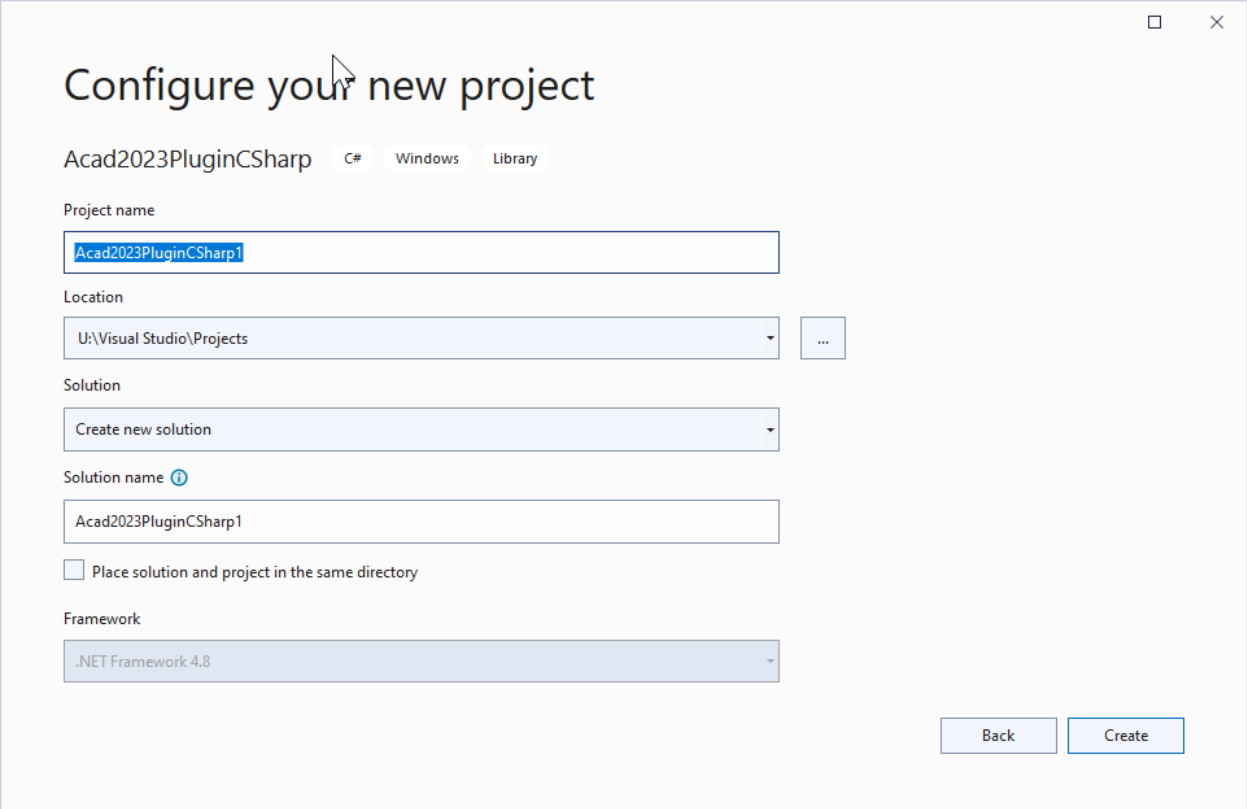
My Documents\Visual Studio 2022\Templates\ProjectTemplates and **My Documents\Visual Studio 2022\My Exported Templates** directory.

Visual Studio uses the templates located in **ProjectTemplates**, **My Exported Templates** serves rather as backup. This model will be now offered by Visual Studio at the start of a new project.

The dialog below shows what we're trying to accomplish. Notice that our project tags match the filter tags and it's showing first in the list.



After clicking on the **Next** button, the dialog below is displayed. Notice that VS has appended a "1" to the end of the project name.



By default, when you create a new project using the dialog above, and rename the project in the dialog box, it will be automatically reflected in the project generated for the default namespace and the name of the project (DLL). On the other hand, the name of the DLL to be loaded in the script will remain unchanged.
So that the name of the project is reflected in the script, we need to edit the file: **start.scr** and the configuration of the template file: **My Template.vstemplate** located in: **Visual Studio 2022\Templates\ProjectTemplates\Acad2024PluginCSharp.zip.**
Extract the contents of archive **Acad2024PluginCSharp.zip** and open **start.scr** in a text editor.
Replace:
netload " Acad2024PluginCSharp.dll"
by:
netload "$projectname$.dll"
Do not forget a space at the end of the line in the script to mimic {Enter} when running in AutoCAD. Save the changes.

Open the **My Template.vstemplate** file in a text editor (it is an XML file). If you open it in VS, you will get Intellisense. Now replace:

```
<ProjectItem ReplaceParameters="false" TargetFileName="start.scr">start.scr

</ProjectItem>
```

With:

```
<ProjectItem ReplaceParameters="true" TargetFileName="start.scr">start.scr

</ProjectItem>
```

You can also add the version of the Framework required to ensure Visual Studio does offer this model when this version of the Framework is selected, add:

```
<RequiredFrameworkVersion>4.8</RequiredFrameworkVersion>
```

… after the `<Description>` tag.

Next, I changed the sort order to 0001 so that my template appears at the top of the list. The default is 1000.

```
<SortOrder>0001</SortOrder>
```

Finally, we will add some tags so that the project shows up when we use filters in the VS New Project dialog. You can add the following tags:

```
<LanguageTag>CSharp</LanguageTag>
<PlatformTag>Windows</PlatformTag>
<ProjectTypeTag>Library</ProjectTypeTag>
```

When you're done, the content of the file should look like this:

```
<VSTemplate Version="3.0.0"
xmlns="http://schemas.microsoft.com/developer/vstemplate/2005" Type="Project">
  <TemplateData>
    <Name>Acad2024PluginCSharp</Name>
    <Description>Template for creating AutoCAD 2024 C# projects</Description>
    <RequiredFrameworkVersion>4.8</RequiredFrameworkVersion>
    <ProjectType>CSharp</ProjectType>
    <ProjectSubType>
    </ProjectSubType>
    <LanguageTag>CSharp</LanguageTag>
    <PlatformTag>Windows</PlatformTag>
    <ProjectTypeTag>Library</ProjectTypeTag>
    <SortOrder>0001</SortOrder>
    <CreateNewFolder>true</CreateNewFolder>
    <DefaultName>Acad2024PluginCSharp</DefaultName>
    <ProvideDefaultName>true</ProvideDefaultName>
    <LocationField>Enabled</LocationField>
    <EnableLocationBrowseButton>true</EnableLocationBrowseButton>
    <Icon>__TemplateIcon.ico</Icon>
  </TemplateData>
  <TemplateContent>
    <Project TargetFileName="Acad2024PluginCSharp.csproj"
File="Acad2024PluginCSharp.csproj" ReplaceParameters="true">
      <ProjectItem ReplaceParameters="true"
TargetFileName="MyCommands.cs">MyCommands.cs</ProjectItem>
      <ProjectItem ReplaceParameters="true"
TargetFileName="myPlugin.cs">myPlugin.cs</ProjectItem>
      <Folder Name="Properties" TargetFolderName="Properties">
        <ProjectItem ReplaceParameters="true"
TargetFileName="AssemblyInfo.cs">AssemblyInfo.cs</ProjectItem>
        <ProjectItem ReplaceParameters="true"
TargetFileName="Resources.resx">Resources.resx</ProjectItem>
```

```
        <ProjectItem ReplaceParameters="true"
TargetFileName="Resources.Designer.cs">Resources.Designer.cs</ProjectItem>
      </Folder>
      <ProjectItem ReplaceParameters="true"
TargetFileName="start.scr">start.scr</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Save the changes and recreate archive Acad2024PluginCSharp.zip with the modified files.

Replace the old archive in Visual Studio 2022\Templates\ProjectTemplates.

Conclusion

The same procedure can be used to create other templates, targeting other versions of AutoCAD and/or framework, or for other types of projects (LISP functions, class libraries, application extensions, ...).

It is possible to add instructions in the file **start.scr**, such as the launch of a command.

It is also possible to specify a DWG file to open in the Arguments of the command line (tab debug the project properties) with a full or relative path (the root being the output directory).

Example:

"Drawing1.dwg" / nologo /b "start.scr"