Create a C# Template for AutoCAD using Visual Studio 2017

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 2017, allowing to automatically launch AutoCAD by loading the DLL from Visual Studio in Debug mode.

The example shown uses Visual Studio Community 2017 but it is easily transferable to other versions.

It tutorial will create a model for AutoCAD 2018 but there again, it is possible to target other versions of AutoCAD.

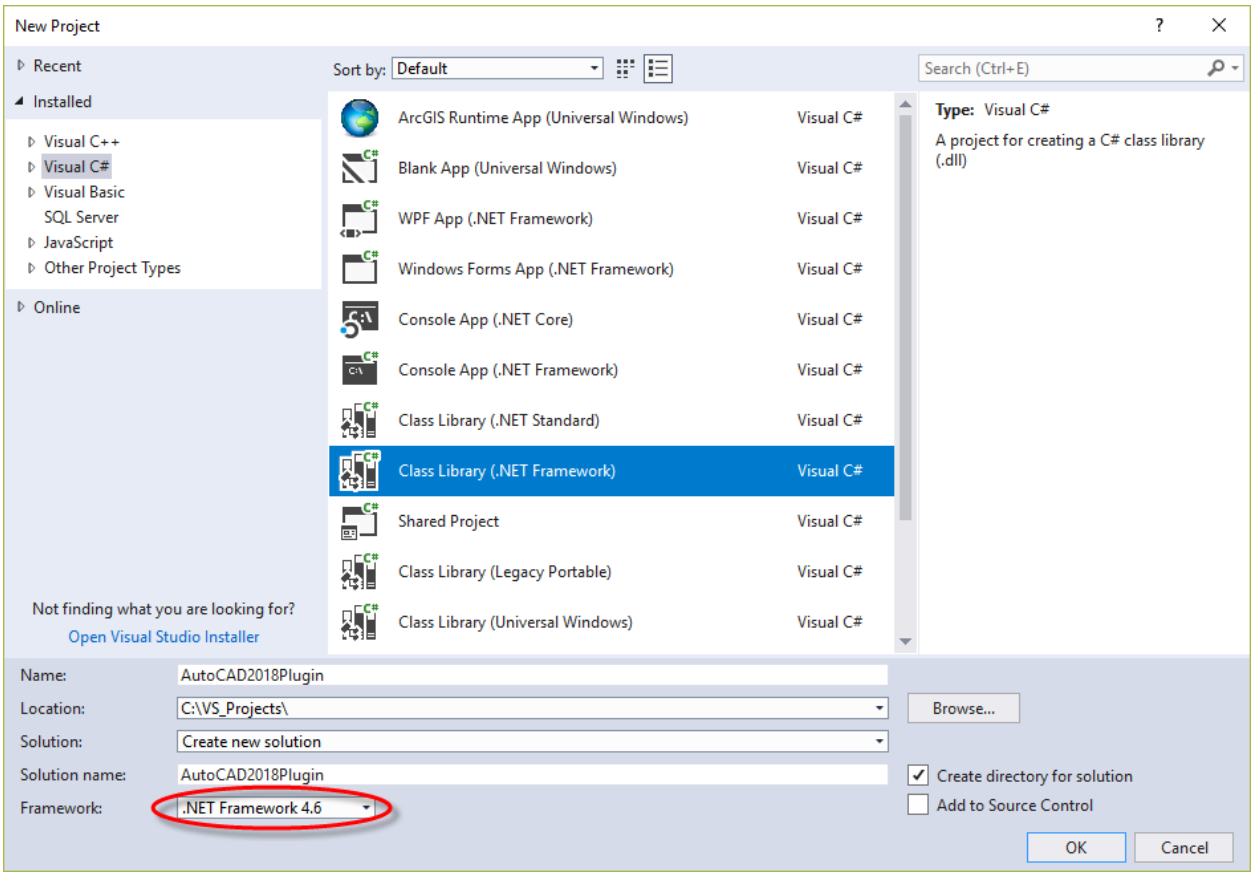
The path to the output directory for the Debug mode used in the example is the default (.\bin\Debug).

Start a new project

In Visual Studio, start a new project (CTRL+Shift+N), select the Visual C# language and the type of project: class library.

Rename the project: Acad2018Plugin for example, and change the path to the location as desired.

Check “Create directory for solution”.

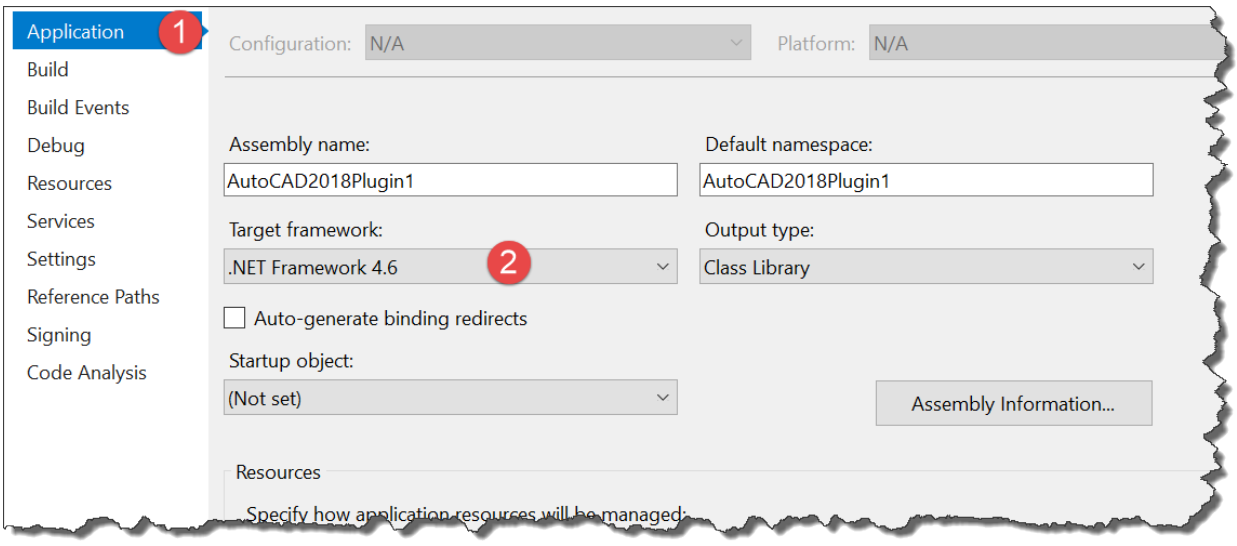


Specify the target Framework

As shown above, in the new project dialog, select the target Framework drop-down list. For 2018, this will be 4.6. For other versions, see the table below.

AutoCAD Version	2013	2014	2015	2016	2017	2018	2019	2020
Release	R19.0	R19.1	R20.0	R20.1	R21.0	R22.0	R23.0	R23.1
DWG Format	AC1027	AC1027	AC1027	AC1027	AC1027	AC1032	AC1032	AC1032
Installed .NET Framework	4.0	4.0	4.5	4.5	4.6	4.6	4.7	4.7
Visual Studio 2010 (10.0)								
Visual Studio 2012 (11.0)								
Visual Studio 2013 (12.0)								
Visual Studio 2015 (14.0)								
Visual Studio 2017 (15.0)								
Visual Studio 2019 (16.0)								

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 add a reference...

In the Reference Manager dialog, click Browse...

References to add are in the version targeted for AutoCAD or, better installation directory, in the ObjectARX20 folder #inc corresponding to the version of AutoCAD targeted (Inc.-win32 or inc - x 64 versions of AutoCAD and platform the station used).

You can download the latest versions of the ObjectARX SDK on this page:

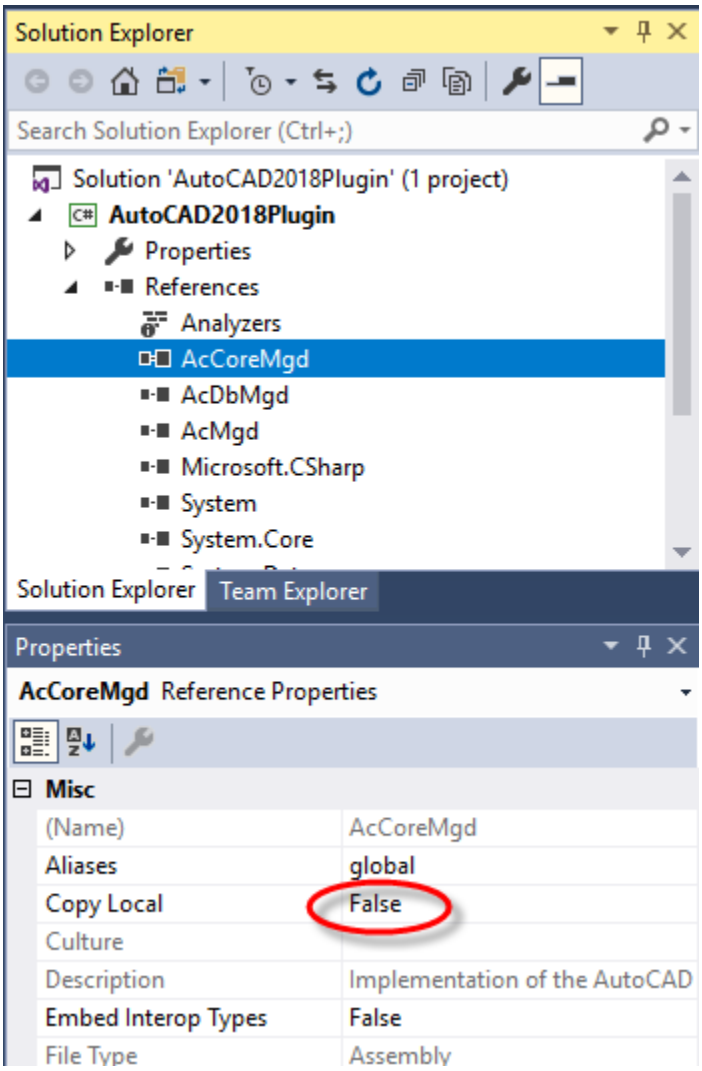
https://www.autodesk.com/developer-network/platform-technologies/autocad/objectarx

http://download.autodesk.com/esd/objectarx/2018/Autodesk_ObjectARX_2018_Win_64_and_32_Bit.sfx.exe

Choose commonly used libraries (it will be easy to add others in the same way if the creation of the new project requires them).

- AutoCAD 2007 to 2012: AcDbMgd.dll and AcMgd.dll
- AutoCAD-2013/2017: AcCoreMgd.dll, AcDbMgd.dll and AcMgd.dll

Change the **Copy Local** property of these dll's to **False**.



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()`.

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.

```
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;

namespace AutoCAD2018Plugin
{
    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();
            }
        }
    }
}
```

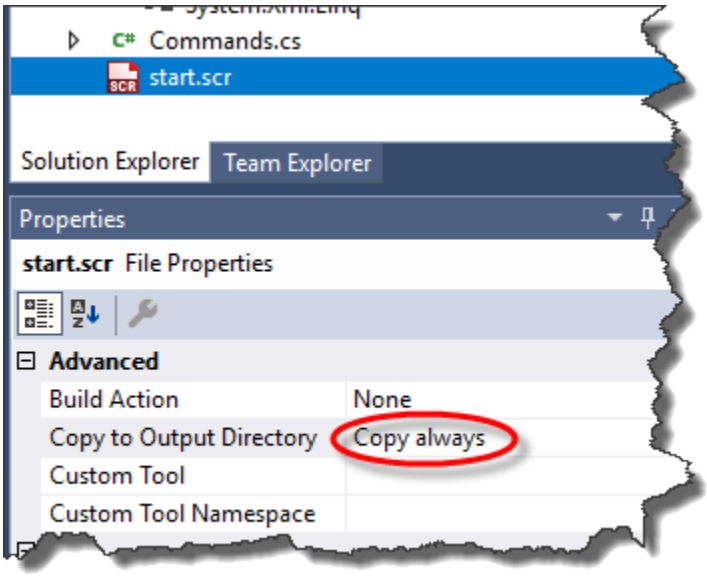
Build the solution, making sure that there are no errors (F6).

Add a script to load the application at startup of AutoCAD

From Solution Explorer, right click on the project Acad2018Plugin 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 "Acad2018Plugin.dll"

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**.



Note: Since AutoCAD 2016, it is imperative that the LEGACYCODESEARCH system variable is 1 for AutoCAD integrates the output directory (.\Acad2018Plugin\bin\Debug) in the search paths and so so it will find the DLL and the script file.

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 Acad2017Plugin.csproj file.
Open the file **.\Acad2018Plugin\Acad2018Plugin\Acad2018Plugin.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 2018\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 2018\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:\ObjectARX1018.

For example, replace the following relative paths:

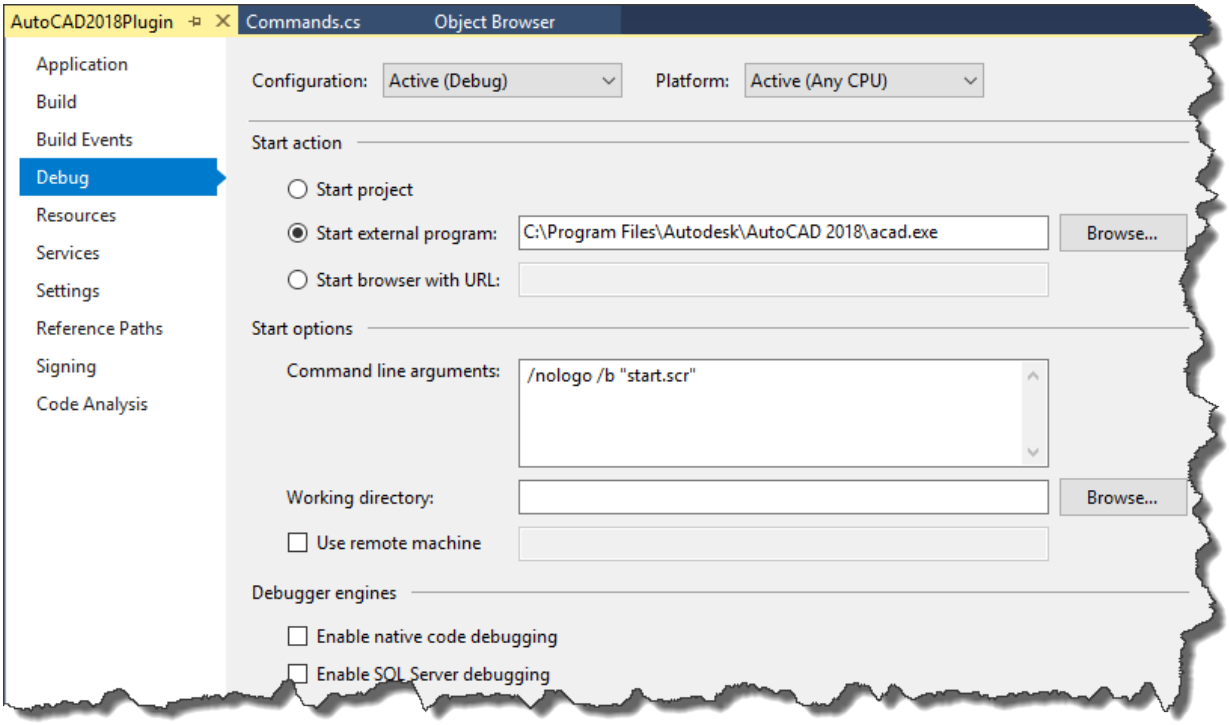
```
<Reference Include="AcCoreMgd">
  <HintPath>..\..\..\..\..\..\..\ObjectARX 2018\inc\AcCoreMgd.dll</HintPath>
  <Private>False</Private>
</Reference>
<Reference Include="AcDbMgd">
  <HintPath>..\..\..\..\..\..\..\ObjectARX 2018\inc\AcDbMgd.dll</HintPath>
  <Private>False</Private>
</Reference>
<Reference Include="AcMgd">
  <HintPath>..\..\..\..\..\..\..\ObjectARX 2018\inc\AcMgd.dll</HintPath>
  <Private>False</Private>
</Reference>
```

With this:

```
<Reference Include="AcCoreMgd">
  <HintPath>C:\ObjectARX 2018\inc\AcCoreMgd.dll</HintPath>
  <Private>False</Private>
</Reference>
<Reference Include="AcDbMgd">
  <HintPath>C:\ObjectARX 2018\inc\AcDbMgd.dll</HintPath>
  <Private>False</Private>
</Reference>
<Reference Include="AcMgd">
  <HintPath>C:\ObjectARX 2018\inc\AcMgd.dll</HintPath>
  <Private>False</Private>
</Reference>
```

Save the Acad2018Plugin.csproj file.

These changes should appear in Visual Studio, in the **Debug** tab of the project properties.



Export the template

Reopen the solution Acad2018Plugin.sln in Visual Studio.

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

Command: netload Assembly file name: "AutoCAD2018Plugin.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: "Acad2018Plugin.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.

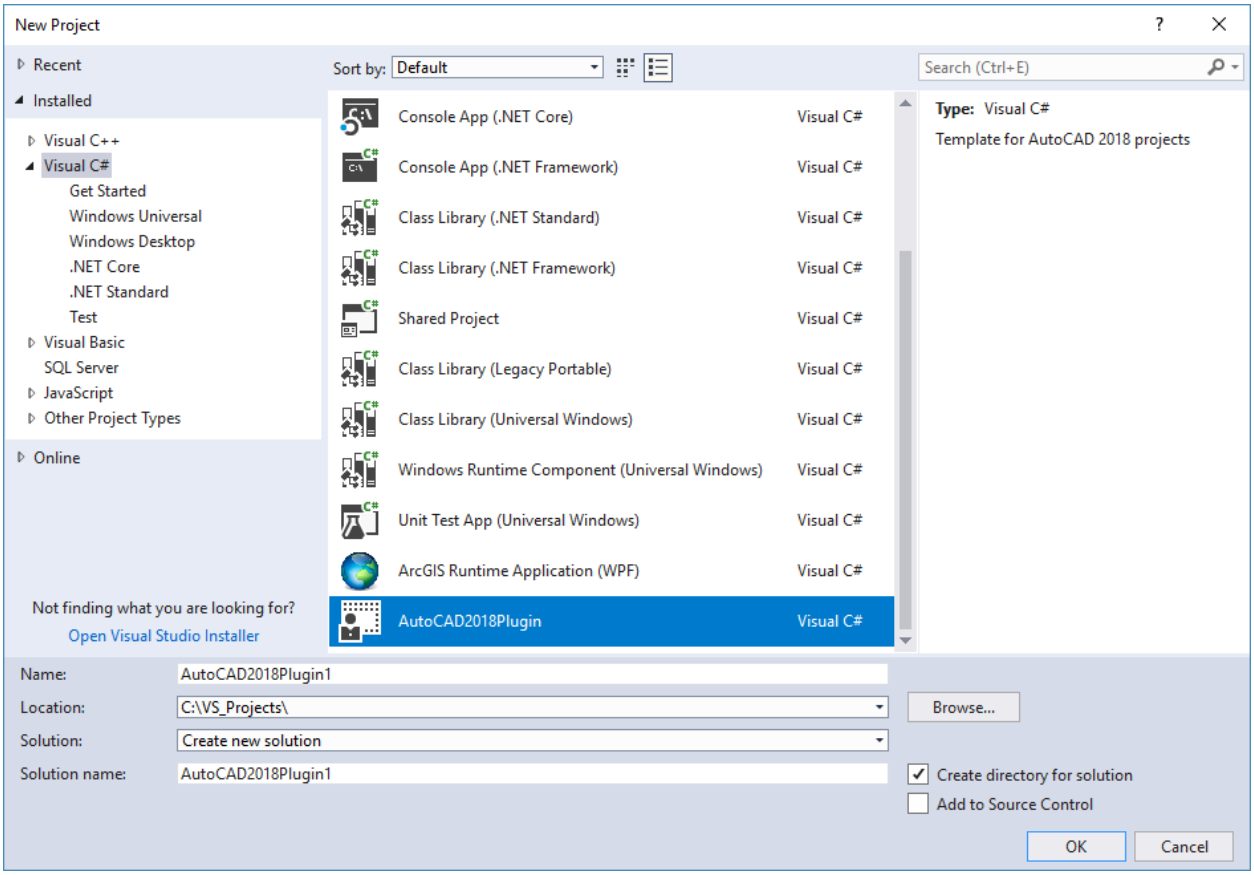
Change the name of the template as desired, AutoCAD 2018 Plugin, for example, and possibly add a description: **Template for AutoCAD 2018**.

Click **Finish**.

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

My Documents\Visual Studio 2017\Templates\ProjectTemplates and **My Documents\Visual Studio 2017\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.



By default, when you 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 2017\Templates\ProjectTemplates\AutoCAD2018Plugin.zip.
Extract the contents of archive AutoCAD 2017 Plugin.zip and open **start.scr** in a text editor.
Replace:
netload "Acad2018Plugin.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) and 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>4.6</RequiredFrameworkVersion>
```

… after the `<Description>` tag.

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>AutoCAD2018Plugin</Name>
    <Description>Template for AutoCAD 2018 C# projects</Description>
    <RequiredFrameworkVersion>4.6</RequiredFrameworkVersion>
    <ProjectType>CSharp</ProjectType>
    <ProjectSubType>
    </ProjectSubType>
    <SortOrder>1000</SortOrder>
    <CreateNewFolder>true</CreateNewFolder>
    <DefaultName>AutoCAD2018Plugin</DefaultName>
    <ProvideDefaultName>true</ProvideDefaultName>
    <LocationField>Enabled</LocationField>
    <EnableLocationBrowseButton>true</EnableLocationBrowseButton>
    <Icon>__TemplateIcon.ico</Icon>
  </TemplateData>
  <TemplateContent>
    <Project TargetFileName="AutoCAD2018Plugin.csproj" File="AutoCAD2018Plugin.csproj"
ReplaceParameters="true">
      <ProjectItem ReplaceParameters="true"
TargetFileName="Commands.cs">Commands.cs</ProjectItem>
      <Folder Name="Properties" TargetFolderName="Properties">
        <ProjectItem ReplaceParameters="true"
TargetFileName="AssemblyInfo.cs">AssemblyInfo.cs</ProjectItem>
      </Folder>
      <ProjectItem ReplaceParameters="true"
TargetFileName="start.scr">start.scr</ProjectItem>
      <ProjectItem ReplaceParameters="true"
TargetFileName="ThisProject.cs">ThisProject.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Save the changes and recreate archive AutoCAD2018Plugin.zip with the modified files.

Replace the old archive in Visual Studio 2017\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"