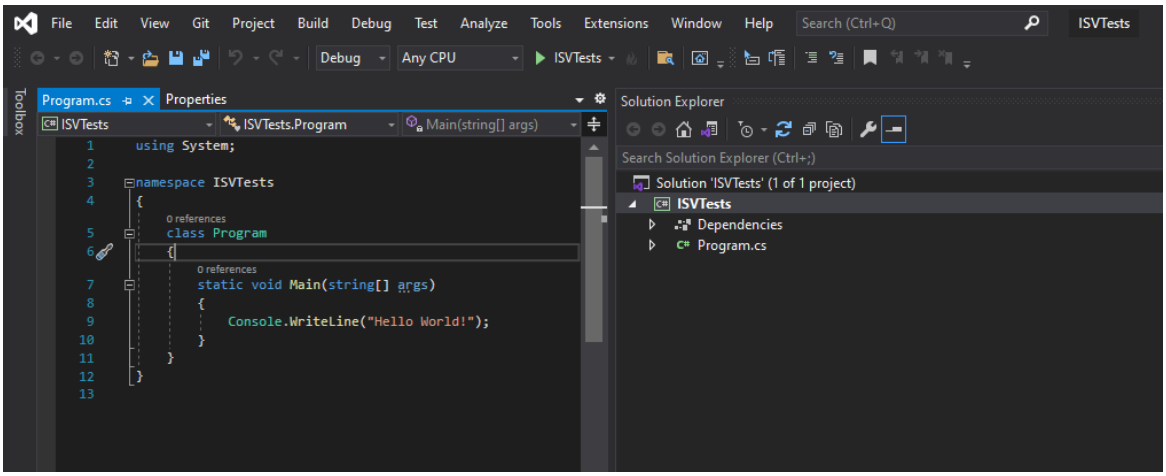
Starting a new Test SDK Project

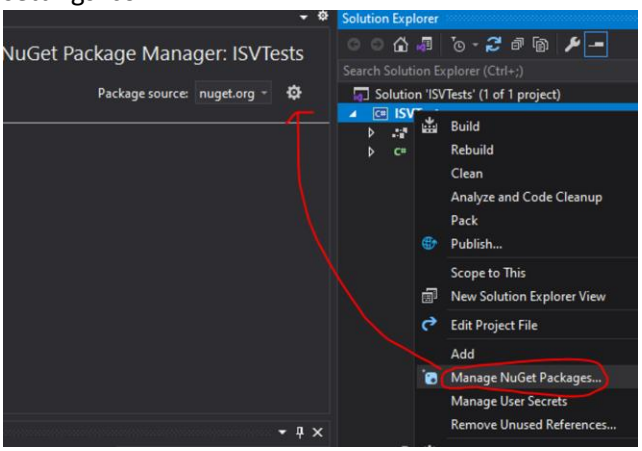
This guide is to assist ISV partners in setting up their initial Test SDK automated test cases to enable a perfect UI driven simulation of their solution's use cases. The Test SDK tests will be able to run against each minor and major release of Acumatica to ensure compatibility with each update and if there is a failure, it will let you know exactly where the failure occurred in the logs.

1. Download all the required files
   a) Download and install the version of Acumatica you are testing on https://builds.acumatica.com/
   b) Download and extract the matching version Test SDK to C:\Acumatica2021R2\TestSDK: http://acumatica-builds.s3.amazonaws.com/index.html?prefix=builds/21.2/21.201.0086/TestSDK/
   c) Download .NET v4.8: https://dotnet.microsoft.com/en-us/download/dotnet-framework/net48

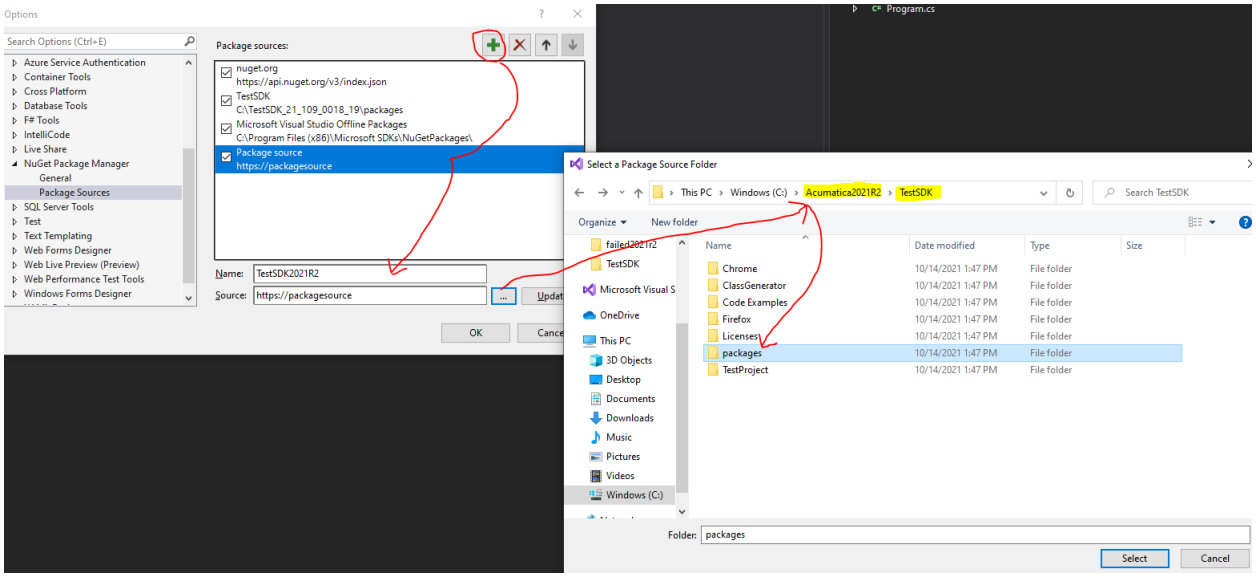
2. Open Visual Studio and create a new project, Using the Console App (.NET Core) C# template.



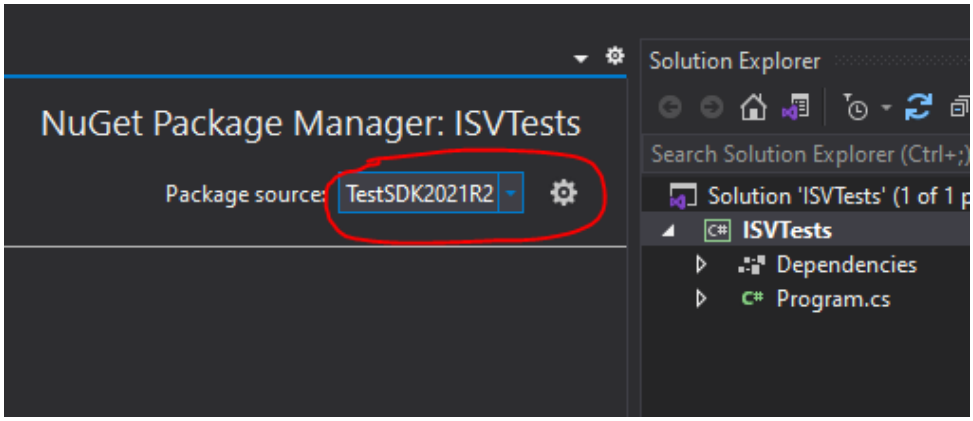
3. Once the project is created, Right click the project and "Manage Nuget Packages", Then click the settings icon.



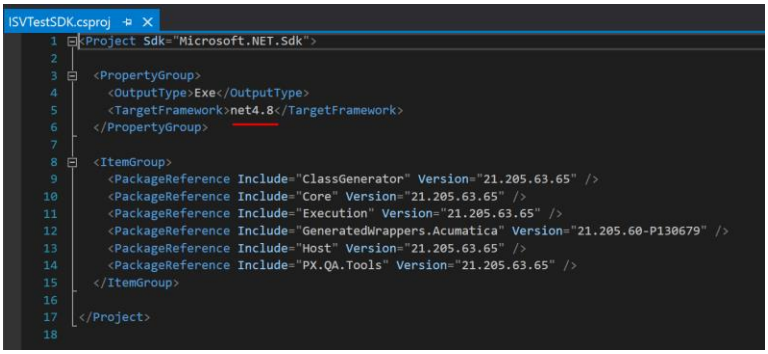
4. Create a new package source, enter a name (recommended to include the Acumatica build number) and use the testSDK/packages folder we extracted earlier from C:\Acumatica2021R2\TestSDK



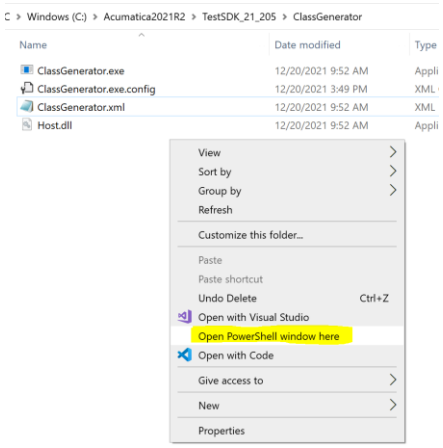
5. Ensure the newly created package source is selected.



6. Install **all** the references from the added package source
7. Your .csproj file should look like this now, you may need to manually type in net4.8 to target the correct framework.



8. Create a new folder named “Wrappers” in the project and then generate the wrappers needed for your tests using the following instructions, you must Generate the wrappers for **every** screen you use for your tests; custom screens and Acumatica screens as well.
   a) Go to C:\Acumatica2021R2\TestSDK_21_205\ClassGenerator folder
   b) Edit ClassGenerator.exe.config
   c) Ensure you include all screens you need to use during your test in the FilenameFilter attribute. As well as fill the other attributes with your sites correct information.
   d) Shift+right click the ClassGenerator folder to “Open with powershell”

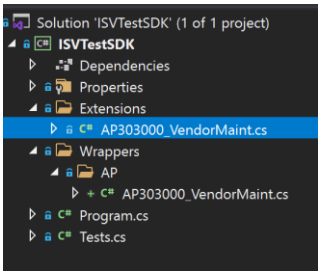


   e) In PowerShell run the following command to generate the wrappers “.\ClassGenerator.exe” The wrappers will be generated into the specified folder you declared in the config file. **Note:** the generate wrappers command prompt will take 2-3 mins to show progress, don’t worry it is not frozen, just keep waiting for the first screen wrapper to generate then they start going quickly.
   f) Copy those folders with wrappers inside to the Wrappers folder of your solution
   g) Congrats, you have wrappers generated for the default Acumatica and Custom screens! Now to create extensions to make them accessible!

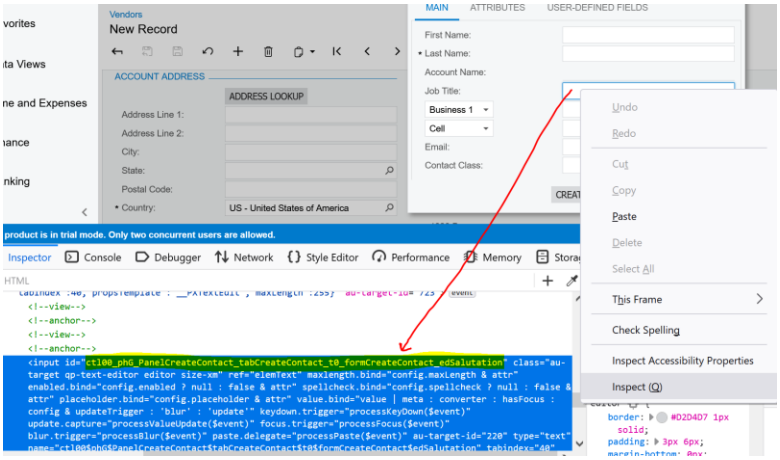
9. Create the Extensions for the Wrappers so they become accessible inside your solution
   a) Create a folder names “Extensions” in the test project.
   b) Make a cs file for each screen, form and action used in your code Follow the below instructions, You can also see examples without explanation from page 10-12 of the ReadMe file in the Test SDK download folder.

How to create Extensions for Custom screens:

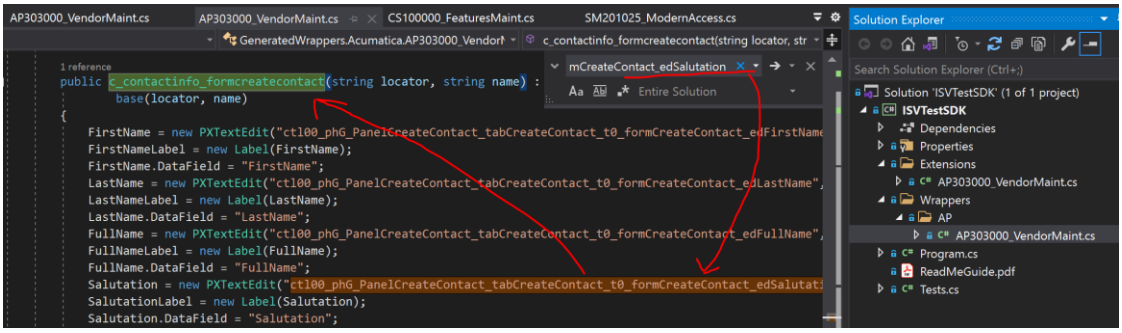
a) Create a .cs file in the extensions folder.



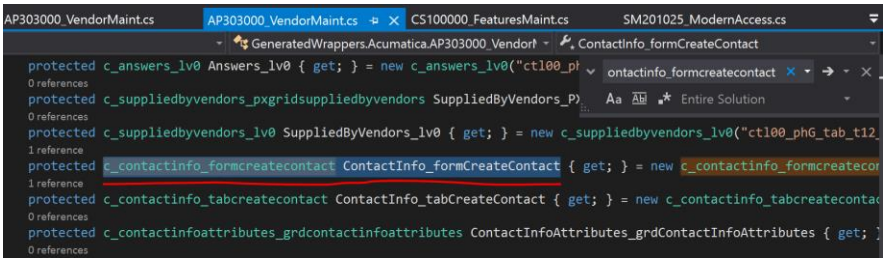
b) Right click and Inspect element on the form/button/field you want to use in your test, Copy the label text or element ID of a unique or easily identifiable field you want to access, then take that field name and search the generated wrapper file for the element id or field's label name



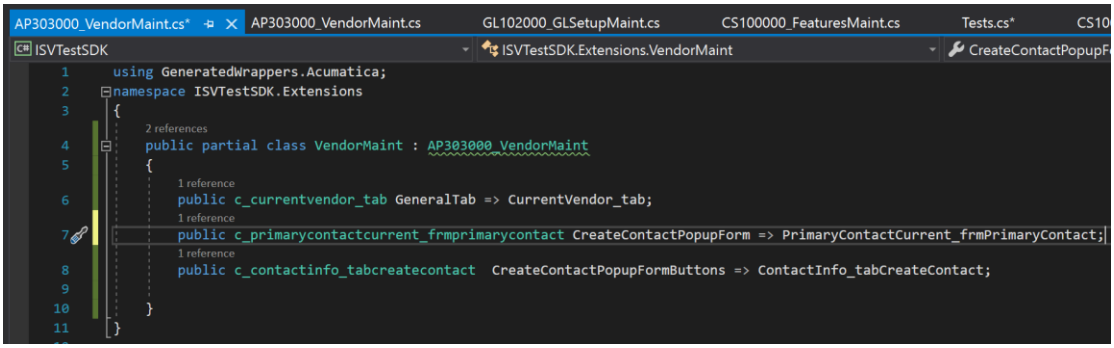
c) Search the name of the element ID or label name in the generated wrapper file



d) Then search for the first use of the class name for that field. Copy the **class <u>and</u> class name** from the wrapper.



e) Paste those 2 values on the extension file in the format shown below to make it accessible in your test. Name it appropriately to use in your test cases, in this case "`CreateContactPopupForm`"



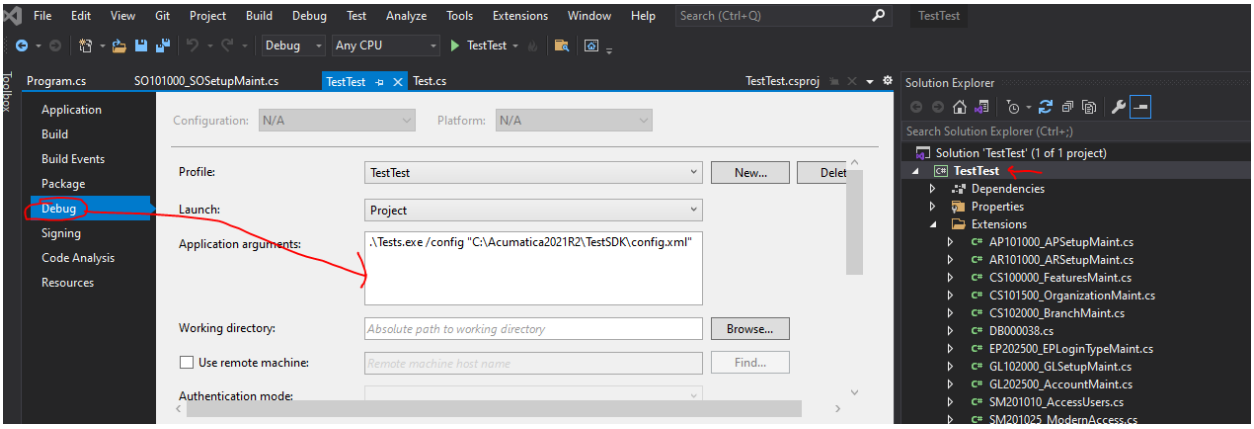
f) Now that you have created the extensions, you can write the tests using those forms/fields, autocomplete will show you all the fields contained in those form elements.

```
VendorMaint.OpenScreen();
VendorMaint.GeneralTab.AcctName.Type("111"); //inserting data into a field of general Tab
VendorMaint.Save();
VendorMaint.CreateContact(); //using an action in the action toolbar menu (...)
VendorMaint.CreateContactPopupForm.LastName.Type("TestLastNameText"); //entering text into the popup form shown by the action above
VendorMaint.CreateContactPopupFormButtons.Create(); //pressing the create button to submit the popup form
```

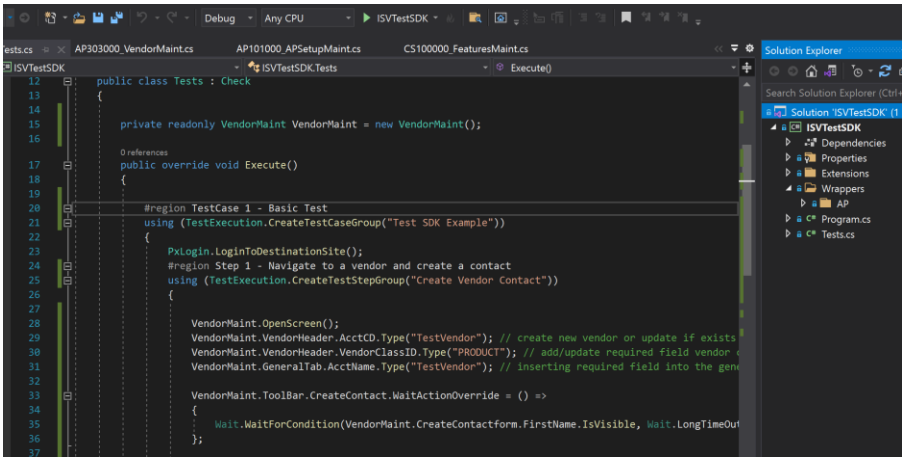
10. Create a basic Test.cs test as seen above. You can make more complex tests following the code snippets on page 15-19 of the SDKReadme.pdf from the TestSDK download .zip.
    Note: the code example test given in the readme only works on a blank copy of Acumatica, I always recommend SalesDemo data as a base dataset as it has items, accounts, customers, contacts, etc pre-configured.

11. Configure the config.xml from C:\Acumatica2021R2\TestSDK modifying as needed for your site.

12. Add the following to the Properties (right click the solution name -> Properties) of the Test Project so you can simply click run(f5) in visual studio to kick off the test.cs.
    .\Tests.exe /config "C:\Acumatica2021R2\TestSDK\config.xml"



13. After setting the config.xml and debug application arguments, Run the project, The Acumatica site will pop up and run through the steps showing the UI as it runs. Then review the output files in the logging folder location you set in the config files..



Github Complete Source Code: https://github.com/AaronBeehoo/Create-Acumatica-Test-SDK-Tests-from-Scratch-for-ISV-s

Common Errors:

1) Acumatica site no longer accessible after generating wrappers or running a test: Something went wrong with the Acumatica site web.config file.
    a) Install a new website, check your TestSDK config files for correct screen ID's, regenerate the wrappers.
    b) Save the orignal web.config and replace it when it breaks
2) Test exited with Error code 0: The test passed successfully.
3) Test exited with Error code 1: Invalid/missing file, likely incorrect folder mapped in the config files somewhere or a missing wrapper/extension