Installing and Using Visual Studio 2022 Professional Edition

Tim Long

July 9, 2024

Installing and Using Visual Studio 2022 Professional Edition

In order to develop .NET applications, Visual Studio can be used. The current licensed edition is Visual Studio 2022 Professional, hereinafter referred to as *VS2022*.

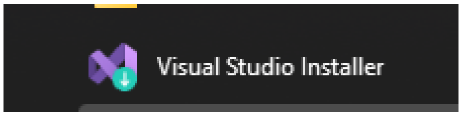
Platform Compatibility

Visual Studio 2022 supports Windows 7 through Windows 11, and .NET Framework 4.6.2 through .NET 8.0.

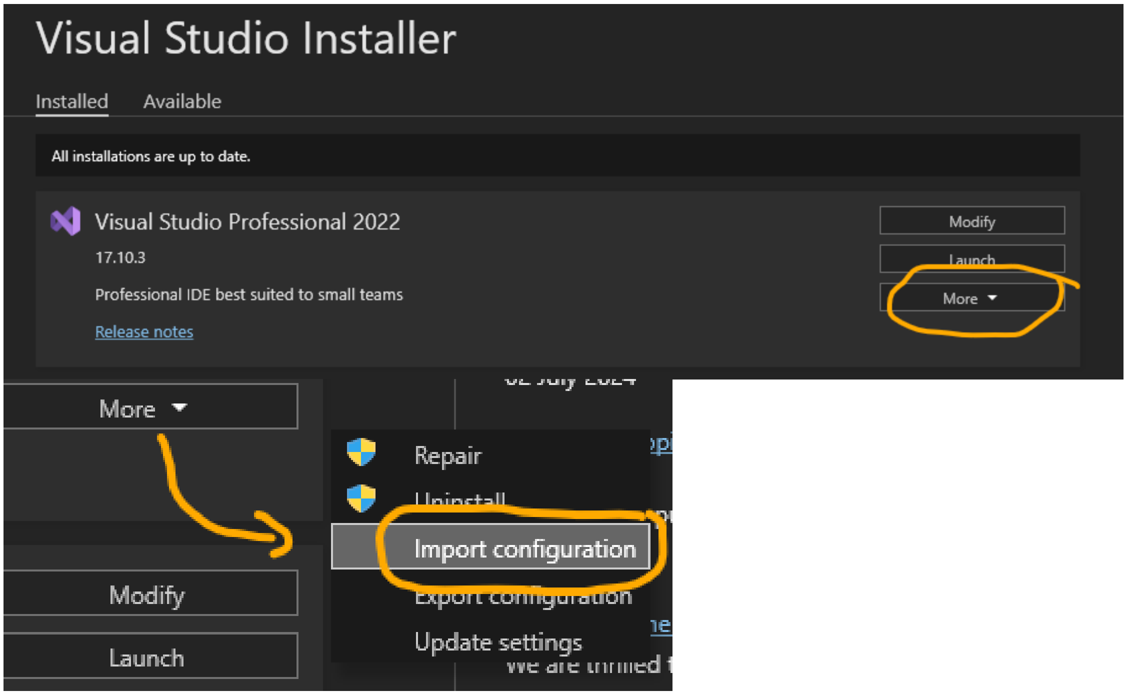
Installation and Configuration

Note: Visual Studio is not the only way to develop .NET applications, but assuming that is the route chosen, then this document details how to install and configure it for use at Ocean Signal.

Proceed as follows:

1. From a DOS or PowerShell command prompt, enter: `winget install Microsoft.VisualStudio.2022.Professional`. This will perform a number of steps:
    - download and install the Visual Studio Installer
    - Install a stock default install of VS2022, which we will customise in the next step.
2. Once installation is complete, re-run the Visual Studio Installer which will now be available on the Windows start menu


3. The installer will show your already-installed instance of VS2022. Pull down the `More` button and select `Import configuration`.



Visual Studio Installer

All installations are up to date.

Visual Studio Professional 2022

17.10.3

Professional IDE best suited to small teams

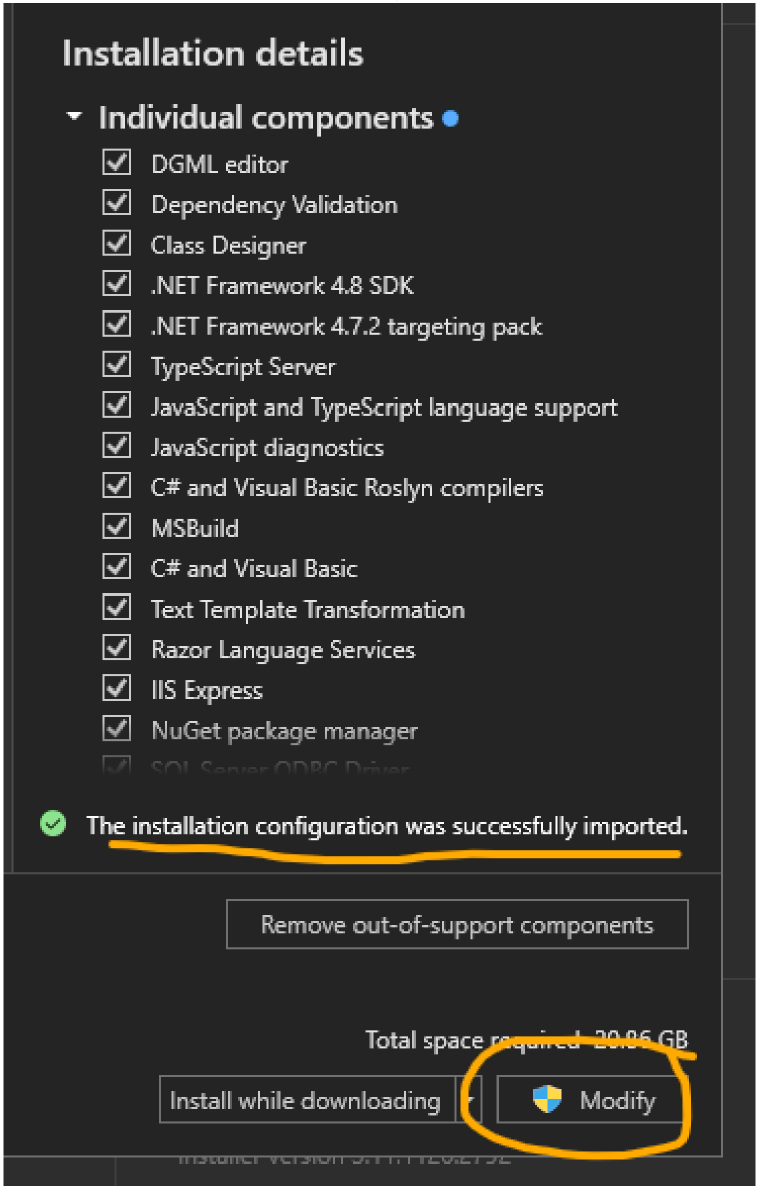
4. Select the file `Ocean Signal Standard Install VS2022.vsconfig` which should be located in the same folder as this document.

5. Click `Review details` and you will have an opportunity to review the selected components and add/remove any as necessary. It is recommended to start with the default set unless there is a compelling reason otherwise. The default selection contains everything necessary to develop C# desktop applications .NET Framework 4.6 through .NET 8.0 and targetting Windows 7 through Windows 11.

It does NOT install tools for mobile development, web development, Windows Azure cloud services, Python, F#, Visual Basic, Game development, Machine Learning tools, etc.

It must be expected that developers will have different needs and will therefore need to install additional components so this should not be regarded as an exclusive configuration, however it is best to start off with the known-good standard configuration. Additional components can be added later on a ‘as-needed’ basis.

When happy, click the `Modify` button to apply the configuration.



Installation details

Individual components ●

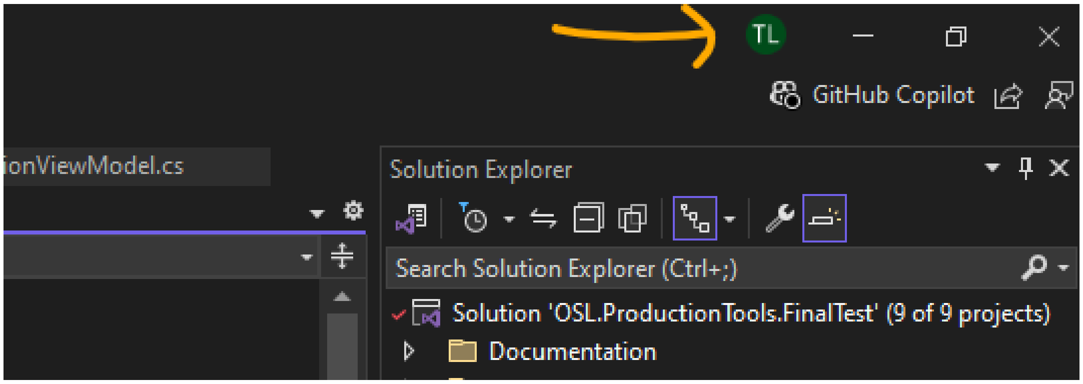
- ☑ DGML editor
- ☑ Dependency Validation
- ☑ Class Designer
- ☑ .NET Framework 4.8 SDK
- ☑ .NET Framework 4.7.2 targeting pack
- ☑ TypeScript Server
- ☑ JavaScript and TypeScript language support
- ☑ JavaScript diagnostics
- ☑ C# and Visual Basic Roslyn compilers
- ☑ MSBuild
- ☑ C# and Visual Basic
- ☑ Text Template Transformation
- ☑ Razor Language Services
- ☑ IIS Express
- ☑ NuGet package manager
- ☑ SQL Server ODBC Driver

✓ The installation configuration was successfully imported.

Total space required 20.86 GB

Settings Sync

For the best experience, make sure to sign in with your Ocean Signal Microsoft 365 credentials. Amongst other things, this will sync your settings across computers and can be very handy if for any reason you need to reinstall or use a different computer.


TL
GitHub Copilot
ionViewModel.cs
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'OSL.ProductionTools.FinalTest' (9 of 9 projects)
Documentation

Targetting Projects for Maximum Longevity

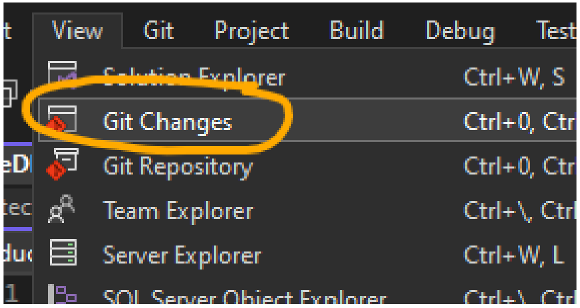
For new projects with no backward-compatibility restrictions, it is recommended to target the latest released long-term support (LTS) version of .NET. At the time of writing that is .NET 8.0. (.NET 9.0 is in late beta but will not be a LTS version so is best avoided).

For projects that must target older versions of .NET (e.g. for driver compatibility), every attempt should be made to use the latest version possible in order to stay within the support lifecycle for as long as possible. FOr .NET Framework projects, at the time of writing, that is `.NET Framework 4.8.1` and that is unlikely to change. For .NET Core projects the oldest supportable version is currently .NET 6.0 but this will alamost always be upgradeable to .NET 8.0

Git Integration

Note: Visual Studio has good Git integration which rather convenient. BitBucket requires use of SSH private/public key pairs so Git may require some additional configuration for this to work. If this is the case, please see this related document. Once Git is thus configured, it should work for *all* git integrations including Visual Studio, VS-Code and from the command line. If you are using SourceTree, then be aware that it uses a non-standard way of managing the SSH keys that will not work with other integrations or the command line.

You may wish to open the Git Changes panel and keep that docked near your Solution Browser panel:


View
Git
Project
Build
Debug
Test
Solution Explorer Ctrl+W, S
Git Changes Ctrl+0, Ctr
Git Repository Ctrl+0, Ctr
Team Explorer Ctrl+\, Ctrl
Server Explorer Ctrl+W, L
SQL Server Object Explorer Ctrl+\, Ctrl