.NET Framework Unit Testing Issues

Since we have some libraries that still output to NET Framework targets, some workarounds are required because of some design choices that have been made in the MSTest library and Test Explorer.

MSTest Project Styles (SDK vs Non-SDK)

> NOTE: This only applies to projects that build for NET Framework (NET Core, Standard, and NET are all SDK-style).

Regardless of which csproj style (SDK or non-SDK) a project uses, a corresponding MSTest project must always adhere to the csproj style that is native for that build target.

This issue really comes into effect for MSTest projects that compile for NET Framework.

If an MSTest project that targets a NET Framework version is using the newer SDK-style csproj layout, any tests it contains will be ignored during discovery, and will not execute (remaining blue in Test Explorer).

For clarity, here's a list of what csproj styles must be used for what different build targets:

Compile Target	CSProj Style
NET Framework 4.5.2	non-SDK style
NET Framework 4.8	non-SDK style
NET 5	SDK style
NET 6	SDK style
NET 7	SDK style

The positive side of this restriction is that, even though an MSTest framework project (targeting NET Framework) must be non-SDK style, the underlying project under test (referenced by the unit test project) can be either style.
So, we are still allowed to use the SDK-style csproj layout for a NET Framework project.
This is good, since we've standardized on using all SDK-style csproj files for older NET Framework

libraries and applications.

Coverlet.Collector Usage

This library is only compatible with test projects (csproj files) that use the SDK style.

So, don’t add it to any MSTest projects that target NET Framework.

Choosing Latest MSTest Framework Version

This issue applies to solutions that contain unit testing for multiple NET targets.
Specifically for solutions that include NET Framework targets that are not compatible with newer MSTest framework library versions.

Problem

When the Test Explorer runs in a VS solution, it will look for the latest version of the MSTest framework, as the library version to call.
This causes problems for any solutions that contain test projects that have different versions of the MSTest framework.

Currently, this causes trouble for libraries that target NET4.5.2, as the MSTest framework versions for this target are older than those for NET4.8, 5, 6, and 7.
Here are MSTest framework versions for NET4.5.2:

- Microsoft.NET.Test.Sdk - v17.3.3
- MSTest.TestAdapter - 2.2.10
- MSTest.TestFramework - 2.2.10

Likewise, here are the MSTest framework used by test projects for NET 4.8, NET5, 6, and 7:

- Microsoft.NET.Test.Sdk - v17.6.2
- MSTest.TestAdapter - 3.0.4
- MSTest.TestFramework - 3.0.4

Since the Text Explorer ONLY calls the latest found MSTest framework library version (after scanning all test projects in a solution), any tests for a NET4.5.2 library become incompatible, and their tests remain undiscovered, never execute, and remain blue.

Workaround

The workaround for this is to create a separate solution that only includes the NET4.5.2 MSTest project.

Simply include, in that unit test project, an external dependency to the assembly under test, by adding a Reference and browsing to the bin/debug of the project under test and choosing its project output dll (or exe).

Once this is done, you can run the unit tests on the NET4.5.2 library as normal.
It just has to be done in this isolated solution.

> NOTE: If your unit test includes any SharedProject references, you can add them as an “Existing Project” by pointing back to where they are in the main library solution.

An example of this being done is in the OGA.TCP.Lib project, where it includes build target for NET4.5.2, which requires the above testing isolation, workaround, and some SharedProject references in the unit test project.

Revision #2
Created 11 May 2025 06:36:29 by glwhite
Updated 11 May 2025 07:19:51 by glwhite