The instructions for using the Microsoft Visual Studio Test facility to run your TDS test methods, as described in the _TDS User Guide_ section 4.5.2, are not usable with Visual Studio Community 2017. Until those instructions are revised, the following instructions give a brief, but I hope adequate, summary of the work needed to add TDS unit tests to a Visual Studio Solution. In my testing they work properly with either Visual Studio Community 2015 or Visual Studio Community 2017; I have successfully used these instructions with both versions.

Since TDS is never shipped as a part of Visual Studio, some setup will be required. (Sorry.) The good news is, much of what follows needs to be done only once for a Visual Studio Solution, after which you may be able to create and run a working unit test in only a few minutes (depending on the level of detail you desire and the complexity of your function member). In the trivial example shown later in these instructions, this can be done in under two minutes, though that's kind of unrealistic for real-life work.

Having created some TDS test methods, you will be able to run tests with them using either the TDS platform or the VS Test platform to produce test reports identifying possible problems in your code. (You may also run tests using both platforms, if you wish to verify that they produce the same results).

For brevity, in the following instructions I shall use "VS" to refer to either Visual Studio Community 2015 or Visual Studio Community 2017.

If your copy of VS does not contain the TDS "TestMethodSnippet" code snippet, import that using the Tools, Code Snippets Manager (see section 4.4.4 in the TDS User Guide).

In VS, in the Solution Explorer window, set up a new VS Project, which will automatically become the basis of a new VS Solution containing that Project. For example, create a new Visual C# Windows Desktop Windows Forms App(.NET Framework), a new Class Library(.NET Framework), or a new Console App(.NET Framework) Project. (I have used TDS successfully with all of these, but the following instructions may fail to work with some other types of Projects.)

Add an accessible (for example, "public") function member (a method, property, indexer, etc.) in an accessible class in the new VS Project to serve as the object of a TDS test method that you will presently create. As a trivial example, you might add the method "public static int AddOne(int n) { return n + 1; }" to the Program{} or Class1{} class to be tested. If needed, declare that class to be public.

In the Solution Explorer window, to the VS Solution, use pop-up menu item Add, New Project to add a new "Visual C#, Test, Unit Test Project(.NET Framework)". Change the Name of the new Project from "UnitTestProject1" (or similar) to "TDS". Change its pathname if desired.

In the TDS Project's Properties window, on the Application tab, change the Output Type of the TDS Project from Class Library to Console Application.

Delete the existing UnitTest1.cs (or similarly named) file from the TDS Project.

To the TDS Project, use pop-up menu item Add, Existing Item to add a copy of the downloaded TDS.cs file. Open TDS.cs for editing in VS (double-click on it).

In file TDS.cs, delete the "#define TDS_platform" line (line 58), or comment it out by prefixing it with "//".

In the TDS Project, use Add, Reference to set a reference to the Project (for example, "ClassLibrary1", "ConsoleApp1", or "WindowsFormsApp1") containing the function member(s) to be tested.

Open the Task List window, which allows you to quickly navigate to Tasks in the code by double-clicking on their names. At the "TODO: Usings" Task, change "using NewCodeNamespace" to refer to the namespace (for example, "ConsoleApp1", "ClassLibrary1", or "WindowsFormsApp1") containing any public function member(s) to be tested.

At the "TODO: TestableConsoleMethodTest() example" Task, delete the example TestableConsoleMethodTest() method, as directed.

At the "TODO: InitializeClasses(), static variables" Task, delete the example Boolean expressions.

In the "TODO: TestMethodsSourceFiles" Task, delete the name of file "TDS_Ex01.cs".

(TDS is now set up to define new test methods, using the TdsTest code snippet. As you develop a new public function member, or modify an existing one, you can add and use a test method for it by proceeding from this point.)

At the "TODO: New TDS methods may be placed here:" Task, use the TdsTest code snippet to create a new Unit Test Method to test the accessible (I suggest public) function member, such as "AddOne()", that you have added to your Solution (but that are outside the TDS Project). Do this by typing "TdsTest<tab><tab>" or "TdsT<tab><tab>" and changing the "TestableFunctionMember" field to "AddOne" (no parentheses); press <enter> to finish.

At the "TODO: AddOneTest() -- Provide a suitable calling expression" Task in your new TDS test method, test the "AddOne()" method, if your startup class is, say, Program{}, by using "actual = Program.AddOne(tCase.Arg);". You may delete the "//TODO:" comment.

Also satisfy some or all of the other Tasks in the Task List. Specifically, in the "TODO: TestMethodsToBeRun" Task, replace the list of example names of test methods to include the new one, "AddOneTest". (I suggest leaving this "//TODO:" comment in the program, so that you can easily find it later.)

To run the TDS Task as a stand-alone app, right-click its name in Solution Explorer and set it to "Set as Startup Project". (In a Windows Forms Solution, you may need to close existing windows to make this pop-up menu item visible.) Since the TDS Assert methods are more limited than the Visual Studio versions, test methods developed outside of TDS (for example, other than by using the TdsTest snippet) may raise compile-time exceptions.

As you run the TDS tests, you will likely hit an "Assert..." exception pop-up the first time you encounter a "Failed" or "Inconclusive" test. Uncheck the "Break when this exception type is user-unhandled" box, close the pop-up, and resume running (use F5). Having hidden these two types of exceptions, you should not see them again during the test session. (See the TDS User Guide, section 4.4.2.)

In VS, run the tests by, for example, selecting "Run All" in the Test Explorer window or under the VS "Test" menu item. The test results should be similar to those obtained by running stand-alone TDS, and they should run regardless of which Project is set as the Startup Project. However, work done by TDS in Tasks such as "TODO: InitializeClasses()" or "TODO: CleanupTestSession()" will not have been performed by the VS Test facility.

=====