TAEF Overview

TAEF, the Test Authoring and Execution Framework, is used extensively within the Windows organization to test the operating system code in a unified manner for system, driver, and application code. As the console is a Windows OS Component, we strive to continue using the same system such that tests can be ran in a unified manner both externally to Microsoft as well as inside the official OS Build/Test system.

The official documentation for TAEF describes the basic architecture, usage, and functionality of the test system. It is similar to Visual Studio test, but a bit more comprehensive and flexible.

Writing Tests

You may want to read the section Authoring Tests in C++ before getting your hands dirty. Note that the quoted header name in `#include "WexTestClass.h"` might be a bit confusing. You are not required to copy TAEF headers into the project folder.

Use the TAEF Verify Macros for C++ in your test code to perform verifications.

Running Tests

If you have Visual Studio and related C++ components installed, and you have successfully restored NuGets, you should have the TAEF test runner `te.exe` available locally as part of the `Microsoft.Taef` package.

> Note that you cannot easily run TAEF tests directly through Visual Studio. The `Microsoft.Taef` NuGet package comes with an adapter that will let you browse and execute TAEF tests inside of Visual Studio, but its performance and reliability prevent us from recommending it here.

In a "normal" CMD environment, `te.exe` may not be directly available. Try the following command to set up the development environment first:

```
.\tools\razzle.cmd
```

Then you should be able to use `%TAEF%` as an alias of the actual `te.exe`.

For the purposes of the OpenConsole project, you can run the tests using the `te.exe` that matches the architecture for which the test was built (x86/x64):

```
te.exe Console.Unit.Tests.dll
```

Replacing the binary name with any other test binary name that might need running. Wildcard patterns or multiple binaries can be specified and all found tests will be executed.

Limiting the tests to be run is also useful with:

```
te.exe Console.Unit.Tests.dll /name:*BufferTests*
```

Any pattern of class/method names can be specified after the */name:* flag with wildcard patterns.

For any further details on the functionality of the TAEF test runner, please see the Executing Tests section in the official documentation. Or run the embedded help with

```
te.exe /!
```

If you use PowerShell, try the following command:

```
Import-Module .\tools\OpenConsole.psm1
Invoke-OpenConsoleTests
```

`Invoke-OpenConsoleTests` supports a number of options, which you can enumerate by running `Invoke-OpenConsoleTests -?`.

Debugging Tests

If you want to debug a test, you can do so by using the TAEF /waitForDebugger flag, such as:

```
runut *Tests.dll /name:TextBufferTests::TestInsertCharacter /waitForDebugger
```

Replace the test name with the one you want to debug. Then, TAEF will begin executing the test and output something like this:

```
TAEF: Waiting for debugger - PID <some PID> @ IP <some IP address>
```

You can then attach to that PID in your debugger of choice. In Visual Studio, you can use Debug -> Attach To Process, or you could use WinDbg or whatever you want. Once the debugger attaches, the test will execute and your breakpoints will be hit.