eggIntegration for Visual Studio  
Support notes

# What does eggIntegration do?

eggIntegration for Visual Studio enables testers to execute eggPlant functional scripts from Microsoft Test Manager or from build scripts and for the results to be visible in Microsoft ALM.

# What is Microsoft ALM?

[Microsoft Application Lifecycle Management](https://msdn.microsoft.com/en-us/library/fda2bad5.aspx) (ALM) solution comprises multiple tools and functions:

* Team Foundation Server (TFS)   
  On-site server for code sharing and collaboration
* Visual Studio Online  
  Basically TFS in the cloud
* Source Code Control  
  Team Foundation version control (TFVC) or Git
* Planning, tracking, collaboration and progress reports.
* Build Processes, including CI
* Test Planning, including manual, automated and performance tests.  
  Test planning and test case management is done using Microsoft Test Manager (MTM) or from some versions of Visual Studio. There are several conditions about what tools and version can be used for test management.
* Visual Studio Professional, Ultimate and Test Professional

Only the above versions of Visual Studio integrate with Microsoft ALM, support Team working, Source Control, Project and Test Case management etc. VS Test Professional is a limited version of VS that “*integrates testers, product managers, and other stakeholders directly into the development process*”. It is not for used for software development.  
  
The automated and performance tests using Microsoft tools target testing applications using Microsoft application frameworks or you can call “Generic Tests”.

# What is eggIntegration for Visual Studio?

The only way to execute 3rd party tests such as ePF scripts is to create “[Generic Tests](https://msdn.microsoft.com/en-us/library/dd286732.aspx)” within Visual Studio. A Generic Test can call any executable or batch file with configurable command line arguments and environment variables. The test must also create a *Summary.xml* file with a specific format in a defined location so the results and test output can be displayed by Visual Studio and read by other Microsoft ALM tools.

eggIntegration for Visual Studio installs 3 components:

* A Visual Studio extension that automates the creation of Generic Tests from selected ePF scripts. A new Tool menu item is added that launches the dialog for selecting ePF scripts and whether they are executed command line or eggDrive.
* A command line tool that executes ePF script and translates the results into the required Visual Studio Format.
* A Visual Studio Extension for syntax highlighting SenseTalk scripts.  
  This is only for cosmetic value if a user chooses to open a SenseTalk script in Visual Studio.

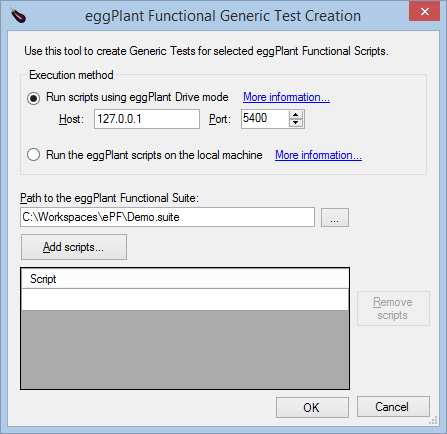
The eggIntegration setup installs the files in “%ProgramFiles(x86)%\ eggIntegration for Visual Studio” by default. The Visual Studio extension packages have the extension .vsix and are launched at the end of the setup package. You can also double-click to install them. Packages can also be installed, uninstalled and disabled from the VS menu, Tools, Extensions and updates. You can only uninstall the packages using this Tool.

Note that I have uploaded the eggPlant SenseTalk code highlighter to the Visual Studio extensions online gallery.

## eggIntegration Visual Studio Extension

After installing the extension eggPlantVSPackage.vsix, there is a new Tool Menu item “eggPlant Functional Tests…”

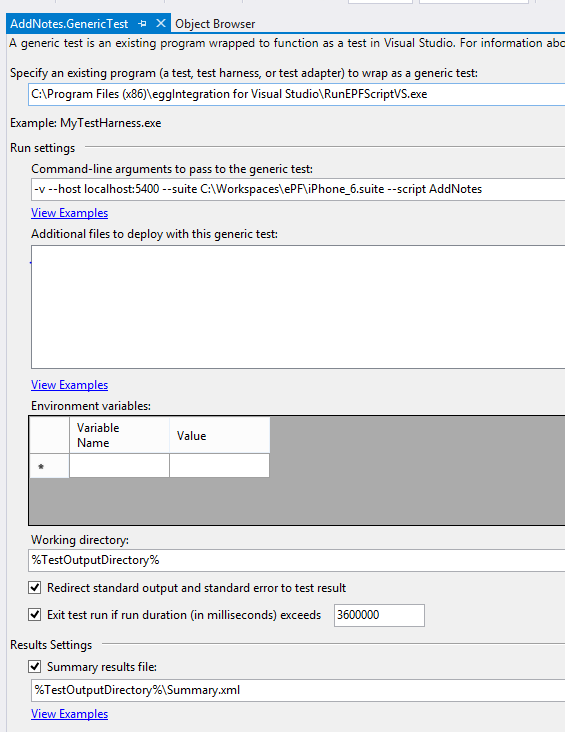
You must first open or create a Visual Studio Test Project, only C# and Visual Basic Test Projects can be used and you would normally create a Unit Test Project (is so delete the default UniTest1.cs or UnitTest.vb file).



* The search button and Add button are disabled if the Host is not an IP address or hostname for the local system.
* If the suite is on a remote system you must enter the suite path manually and it must be a path recognised by eggPlant Functional running in Drive mode on that remote system.
* You can multiple select scripts from the explorer file dialog
* Double click in the bottom empty line of the script table to add a new script manually.

## Generic Test Parameters

This is a standard VS dialog, not one created by eggIntegration.



**Program to wrap the test.**  
RunEPFScriptVS.exe is the wrapper that calls either the ePF runscript.bat or executes the script using XMLRPC against ePF running in Drive mode.

**Command-line arguments**See below

**Working Directory**The default %TestOutputDirectory% is a unique path created by Visual Studio e.g.   
C:\Users\Ian\Documents\Visual Studio 2013\Projects\UnitTestProject2\TestResults\Ian\_IMP-XPS 2015-01-26 14\_49\_44\In\88171556-2d8d-4023-9969-f24c60ad4f02\IMP-XPS

The ePF results are copied to this folder on test completion

Note: You must save any changes before running a test.

# RunEPFScriptVS.exe

RunEPFScriptVS.exe is the executable that runs the ePF scripts.  
It displays the following usage if called with incorrect arguments.

-h, --host The host:port of eggPlant Functional running in drive mode

e.g. 192.168.0.99:5400 or localhost:5400

-t, --suite Required. path to eggPlant Functional suite on the host

-s, --script Required. the eggPlant script name in the suite

-p, --parameters optional parameters to local RunScript.bat command

-v, --verbose (Default: False) Prints all messages to standard output

--help Display this help screen.

If it has the argument –host, then it uses eggDrive otherwise it calls runscript.bat.

RunEPFScriptVS.exe looks for the eggPlant runscript.bat file if the folders given by the following environment variables.  
%ProgramFile%, %ProgramFiles(x86)%,

Finally, it looks in %EGGPLANT%. This is to allow you to specify a system environment variable if ePF is not installed in one of the above locations.

If using RunScript.bat, the following command line argument is launched  
%eggPlantFolder%\runscript.bat -GlobalResultsFolder Results.

If additional parameters –p, is given on the GenericTest dialog, then these are copied as further parameters with no additional quotes. As using quoted arguments or arguments including spaces can be difficult, I would suggest users try to avoid these and code them into the ePF script instead.

If using eggDrive, then RunEPFScriptVS.exe uses XMLRPC and calls the following pseudo code:

url = http:// + host:port  
StartSession suite  
Execute RunWithNewResults script  
EndSession

XMLRPC errors are caught and converted into a suitable format for reporting.

On completion of the script, summary.xml is created that includes the pass/fail result of the script plus paths to the logfiles e.g.  
  
<?xml version="1.0" encoding="utf-8"?>

<SummaryResult xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<TestName>C:\Workspaces\ePF\iPhone\_6.suite dummy</TestName>

<TestResult>Passed</TestResult>

<DetailedResultsFile>LogFile.txt</DetailedResultsFile>

</SummaryResult>

All ePF logfiles and screen capture files apart from logfile.xml are copied to the VS *TestOutputDirectory*.  
Any file ending in .success or .failure is copied and renamed to ResultCsv.txt

Note: The *summary.xml* format has the option of creating “Inner tests” inside a test. I wondered if these could be used to represent the output of Start/EndTestCases but there was not clear mapping between them.