OpenCover Usage Guide

Intro

The following guide describes how to use OpenCover to gather coverage statistics of your application.

OpenCover can handle 32 and 64 bit .NET processes running on the .NET 2 and .NET 4 frameworks. OpenCover will gather sequence and branch coverage information of your assemblies that match the filters and for which the PDB files can be found. Currently OpenCover has no presentation of results other than the XML output file; ReportGenerator (http://www.palmmedia.de/Net/ReportGenerator) is currently the recommended tool for visualizing the results.

NOTE: No PDB for an assembly then no coverage data will be gathered; this is different to PartCover which will default to IL coverage under this situation but it was considered as not required as this is supposed to be a code-coverage tool which can relate such coverage to ***your*** code.

Table of Contents

Command Arguments

OpenCover has a number of arguments that can be used to control the code coverage gathering. If an argument requires spaces then use "'s to wrap the argument, where they are applicable they will be indicated with an optional syntax [].

Mandatory

["]-target:<target application>["]

> The name of the target application or service that will be started; this can also be a path to the target application.

Alternatively use *-?* to show command line help.

Optional

["]-targetdir:<path to the target directory>["]

> The path to the target directory; if the target argument already contains a path then this argument can be used to provide an alternate path where PDB files may be found.

["]-targetargs:<arguments for the target process>["]

> Arguments to be passed to the target process.

-register[:user]

> Use this switch to register and de-register the code coverage profiler. Alternatively use the optional user argument to do per-user registration where the user account does not have administrative permissions. Alternatively use an administrative account to register the profilers using the regsvr32 utility.

["]-output:<path to file>["]

> The location and name of the output xml file. If no value is supplied then the current directory will be used and the output filename will be results.xml.

["]-filter:<space separated filters>["]

> A list of filters to apply to selectively include or exclude assemblies and classes from coverage results. Filters have their own format **±[module-filter]class-filter**. If no filter(s) are supplied then a default include all filter is applied +[*]*. As can be seen you can use an * as a wildcard. Also an exclusion filter (-) takes precedence over an inclusion filter (+).

-nodefaultfilters

A list of default exclusion filters are usually applied, this option can be used to turn them off. The default filters are:

-[mscorlib]*
-[mscorlib.*]*
-[System]*
-[System.*]*
-[Microsoft.VisualBasic]*

-mergebyhash

Under some scenarios e.g. using MSTest, an assembly may be loaded many times from different locations. This option is used to merge the coverage results for an assembly regardless of where it was loaded assuming the assembly has the same file-hash in each location.

-showunvisited

Show a list of unvisited methods and classes after the coverage run is finished and the results are presented.

-returntargetcode[:<opencoverreturncodeoffset>]

Return the target process return code instead of the OpenCover console return code. Use the offset to return the OpenCover console at a value outside the range returned by the target process.

-excludebyattribute:<filter>[;<filter>][;<filter>]

Exclude a class or method by filter(s) that match attributes that have been applied that have been applied. An * can be used as a wildcard.

-excludebyfile:<filter>[;<filter>][;<filter>]

Exclude a class (or methods) by filter(s) that match the filenames. An * can be used as a wildcard.

-hideskipped:File|Filter|Attribute|MissingPdb|All [;File|Filter|Attribute|MissingPdb|All]

Remove information from output file (*-output:*) that relates to classes/modules that have been skipped (filtered) due to the use of the following switches *–excludebyfile:*, *-excludebyattribute:* and *–filter:* or where the PDB is missing.

-coverbytest:<filter>[;<filter>][;<filter>]

Gather coverage by test by analysing the assemblies that match these filters for Test methods. Currently only MSTest and NUnit tests are supported; other frameworks can be added on request – please raise support request on GitHub.

-log:[Off|Fatal|Error|Warn|Info|Debug|Verbose|All]

Change the logging level, default is set to Info. Logging is based on log4net logging levels and appenders.

-service

The value provided in the target parameter is the name of a service rather than a name of a process.

-oldstyle

Use old style instrumentation – the instrumentation is not Silverlight friendly and is provided to support environments where mscorlib instrumentation is not working.

Handling Spaces

If your argument needs to escape quotes i.e. to pass arguments with spaces to the target process then you can use \".

e.g.

-targetargs:"\"c:\program files\" arg2 arg3"

Understanding Filters

Filters are core to understanding how OpenCover works and how it is determined which assemblies are to be instrumented to provide coverage results.

Filters can be inclusive and exclusive represented by + and – prefix where exclusive filters take precedent over inclusive filters.

The next part of a filter is the module-filter and usually this happens to be the same name as the assembly but without the extension and this rule will normally apply 99.999% of the time. If this filter isn't working look in the coverage XML and compare the found <ModuleName/> entries against the filter.

The final part of the filter is the class-filter and this also means the namespace part of the class as well.

Examples

`+[Open*]* -[Open.Test]*`
Include all classes in modules starting with Open.* but exclude all those in modules Open.Test

`+[Open]* -[Open]Data.*`
Include all classes in module Open but exclude all classes in the Data namespace.

`+[Open]* -[Open]*Attribute`
Include all classes in module Open but exclude all classes ending with Attribute.

Running against an application

This most common usage is in a testing environment

Sample

```
OpenCover.Console.exe -register:user -target:..\..\..\tools\NUnit-2.5.10.11092\bin\net-2.0\nunit-console-x86.exe -targetargs:"OpenCover.Test.dll /noshadow" -filter:"+[Open*]* -[OpenCover.T*]*" -output:opencovertests.xml
```

Running against a Silverlight application

To run against a Silverlight application it is necessary to ensure the site hosting the application is running beforehand. To profile a Silverlight application it is necessary to launch a browser against the site and as the PDB files are not packaged in the XAP files it is necessary to give the console a hint where to look for the PDB files.

Sample

```
OpenCover.Console.exe -register:user "-target:C:\Program Files (x86)\Internet Explorer\iexplore.exe" "-targetargs:http://localhost:4128/SampleSilverlightTestPage.aspx "-targetdir:..\SampleSilverlight\SampleSilverlight\Bin\Debug"
```

Running against a Service application

It is preferable to run the service in a console mode if it has one rather than as a service however if you do decide to use it against a service then you will need to make sure you use an account that can access the windows synchronisation objects in the Global namespace (rather than Local namespace). “Local System” seems to work quite well and so do user accounts with the appropriate permissions; “Local Service” is usually problematic and is not recommended. The console host will also need to be run from an account that can access the Global namespace as such an Administrator account or an Administrative prompt is recommended.

Sample

```
OpenCover.Console.exe -target:"OpenCover Sample Service" -service –register
```

NOTE: Rather than use the *–register* switch, it is usually simpler to use the *regsvr32* utility to pre-register the two profiler assemblies (32 and 64-bit) beforehand.

Using the *-excludebyattribute* option

Normally you would include/exclude modules and classes by using the inclusion/exclusion filters, however there may be situations where you can’t get coverage via testing and you wish to ignore the uncovered method.

First create a “public” attribute that you can apply to class/method/property which you use to mark up something to ignore. You can have more than one and you can add other data to provide a reason why you are excluding it.

e.g.

```
[AttributeUsage(AttributeTargets.Class|AttributeTargets.Method|AttributeTargets.Property)]
public class ExcludeFromCoverageAttribute : Attribute{}
```

Then you apply this attribute to the class/method/property that you wish to exclude.

Then you add this attribute to the excludebyattribute option using namespaces and wildcards where necessary.

e.g.

```
-excludebyattribute:*.ExcludeFromCoverage*
```

NOTE: Use with care as you could exclude a method which you should be testing; also it can become too tempting to ignore a method and not test due to it being difficult and use this option to 'skip' it.

Using the *-excludebyfile* option

This is a useful option to use to ignore auto-generated files. This works on file and pathnames.

e.g. the following would ignore all code in files ending in *generated.cs*

```
-excludebyfile:*\*.generated.cs
```

NOTE: Use with care as you could exclude a method which you should be testing; also it can become too tempting to ignore a method and not test due to it being difficult and use this option to 'skip' it.

Build system integration

It is not unexpected that OpenCover will be used in a build environment and that the build will be running under a system account under these scenarios it is recommended that you pre-register the profiler DLLs using the regsvr32 utility where applicable for your environment.

```
regsvr32 x86\OpenCover.Profiler.dll
regsvr32 x64\OpenCover.Profiler.dll
```

To assist your build environment when you install OpenCover it will store in the registry a location of the installation folder. The location in the registry depends on whether it is a single-user or an all-user installation and also if you are on a 32/64 bit environment.

See the following examples based on default settings:

all-users (32-bit)

Registry Entry: `HKLM\Software\OpenCover\Location`
Install Location: `%PROGRAMFILES%\OpenCover`

all-users (64-bit)

Registry Entry: `HKLM\Software\Wow6432Node\OpenCover\Location`
Install Location: `%PROGRAMFILES(X86)%\OpenCover`

single-user

Registry Entry: `HKCU\Software\OpenCover\Location`
Install Location: `%LOCALAPPDATA%\Apps\OpenCover`

FAQ

Why do I have no results?

The usual reason for no results is because OpenCover cannot locate the PDBs for assemblies that match the filters to be profiled i.e. gather coverage results from. When each assembly is loaded the location and reason the assembly wasn't profiled is provided in the coverage results file e.g.

```
<Module skippedDueTo="Filter" hash="0C-69-37-C2-8F-AB-FC-E7-72-51-E9-17-19-99-1A-4A-4C-07-72-08">
  <FullName>C:\Personal\opencover.git\working\main\bin\Debug\OpenCover.Test.dll</FullName>
  <ModuleName>OpenCover.Test</ModuleName>
  <Classes />
</Module>
```

The two most common reasons provided are `Filter` and `MissingPdb`.

`Filter` is obviously connected to the `-filter:` argument and that the ModuleName was not matched.

`MissingPdb` is usually because the PDB is not where the assembly is being loaded from. The most common reason is due to test tools such as NUnit or MSTest which copy the assembly under test to a new location; this can be corrected by either using the /noshadow or /noisolation option. An alternative is to use the `-targetdir:` argument to provide an alternative location for OpenCover to use.

The other reasons are only applicable to classes and are related to the use of `-excludebyattribute:` and `-excludebyfile:` options.

All my tests are failing and I am getting MissingMethodException

This has been seen on a few systems where the following command has been executed to improve performance (or something similar)

```
ngen install /Profile "mscorlib"
```

There are two ways to fix or handle this issue

1) Undo the previous command - `ngen uninstall /Profile "mscorlib"`.
2) Use the `-oldstyle` switch, note however that this is not Silverlight friendly.