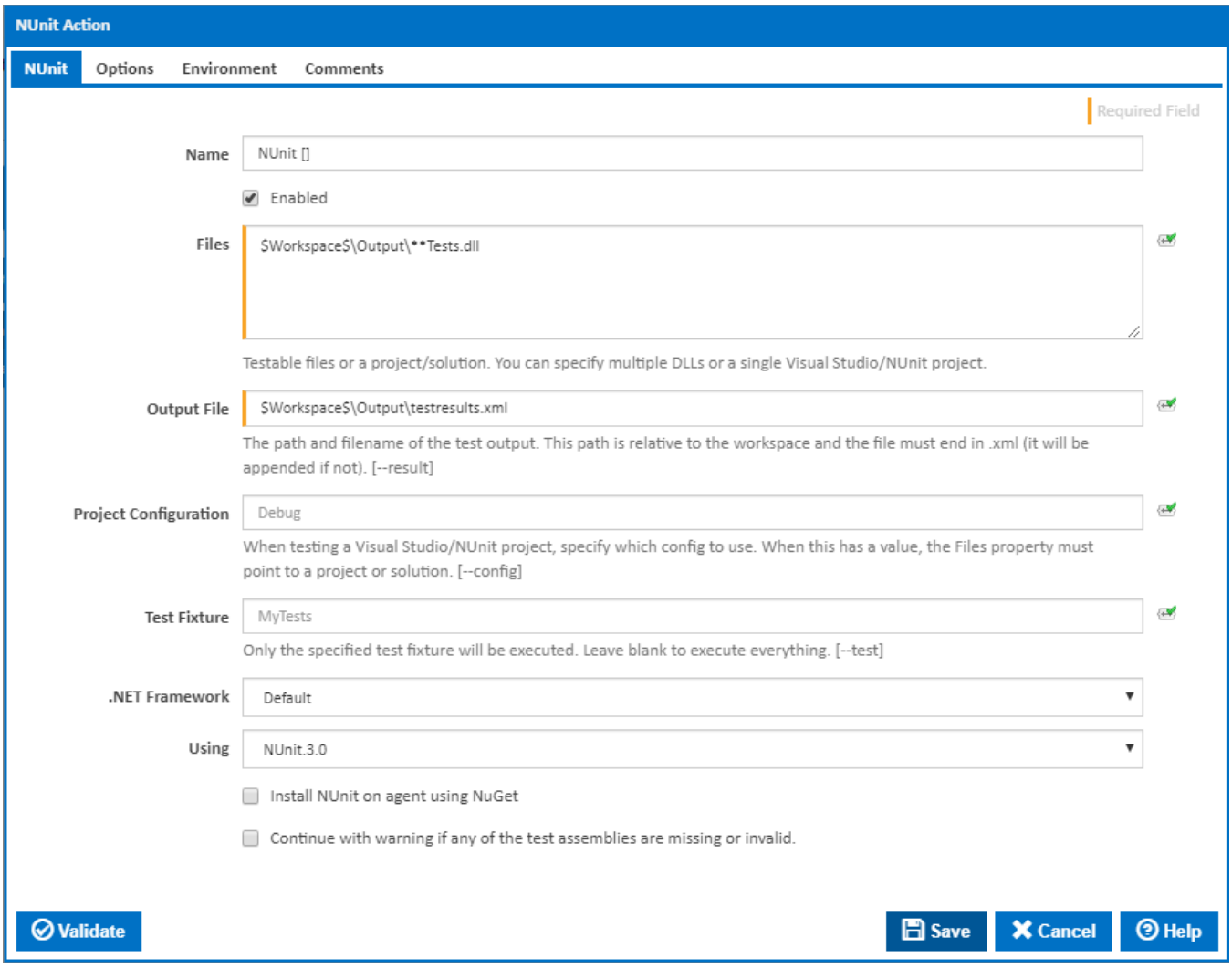
NUnit Action

The NUnit action in Continua is a wrapper around the NUNIT3-CONSOLE command line. If you're having trouble using the NUnit action, please refer to the Command Line Reference.

NUnit is a unit testing framework for the .NET environment. The NUnit action can be pointed to either run tests via multiple .dll files or through a .NET /NUnit project or solution.

NUnit



Name

The name that will be used to identify this action. Note that **Name** is auto-populated with the file names that are entered in the **Files** property.

Enabled

Only enabled actions will run when a build is executed. If an action is disabled then it will be ignored at runtime.

Files

This field can a contain single testable file or multiple testable files including .dll files, NUnit projects or .NET projects and solutions. All files specified are **rel ative to the agent's workspace** so to reference the output directory of the workspace, you only need **Output** (not $Workspace$\Output). However you can still specify absolute files such as C:\myTests and the NUnit action will point to the correct location.

This field also supports Ant Patterns to specify multiple files. For example, **\Output**tests.dll** would test all .dll's that end with **tests** which are located in the Workspace's output folder and any of its subdirectories.

Each file/pattern must start on a new line.

Output File

When NUnit executes, the output will be stored in this file. Even when specifying multiple files to test, all of the output it put into a single file. [--result]

The output file specified is **relative to the agent's workspace** so to reference the output directory of the workspace, you only need **Output** (not $Workspace$\Output). However you can still specify an absolute file location such as C:\myOutputFile.xml and the NUnit action will generate the output file in the correct location.

The output file must end in .xml. If it does not then .xml will be appended to the end of the output file.

Project Configuration

When Project Configuration has a value it will run the NUnit tests from either a Visual Studio or NUnit project rather then .dlls. This value tells NUnit which configuration should be used running the project (ie. Debug, Release, etc). This property must be given a value when you are running NUnit against projects. [--config]

Test Fixture

Test Fixture allows you to specify a specific test fixture to execute. You must specify the full name of the test fixture along with the containing assembly. [--test]

For example: **Tests.MyTests myProject.tests.dll**.

If you leave this input empty then NUnit will test every fixture.

.NET Framework

Specify which version of the .NET framework should be used when running the NUnit action. [--framework]

Using

The Using drop down is populated with any property collector whose namespace matches the pattern defined by the NUnit action. The pattern for this action is `^NUnit.*`

If you create a property collector for this action, make sure you select the **Path Finder PlugIn** type and give it a name that will match the pattern above in blue. Example names listed here, search the table's Plugin column for "**NUnit**".

For more in-depth explanations on property collectors see Property Collectors.

Alternatively, you can select the **Custom** option from the Using drop down list and specify a path in the resulting input field that will be displayed. Please read Why it's a good idea to use a property collector before using this option.

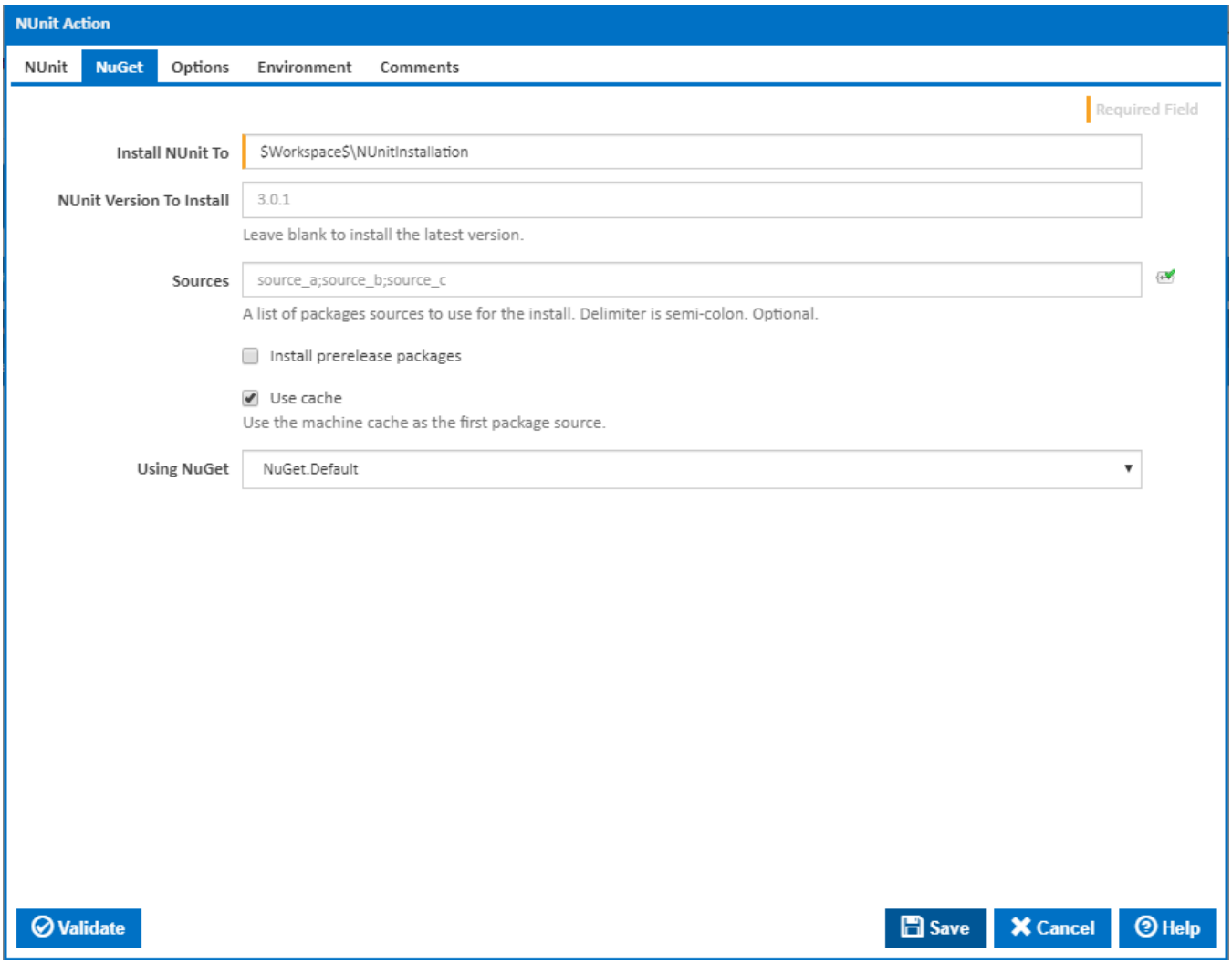
Install NUnit on agent using NuGet

Optionally run NuGet to install NUnit to the agent before running the action. A new **NuGet** tab will also be displayed with options for the NuGet command line.

Continue with warning if any of the test assemblies are missing or invalid

If this is ticked, it continues with warning if any of the test assemblies are missing or invalid.

NuGet



Install NUnit To

The folder NuGet should install NUnit to.

Sources

A list of package sources to install the NUnit package from. Optional. You can separate multiple sources with semi-colons.

Install prerelease packages

Optionally install prerelease versions of the NUnit package.

Use cache

Optionally attempt to source the NUnit package from the NuGet machine cache.

Using

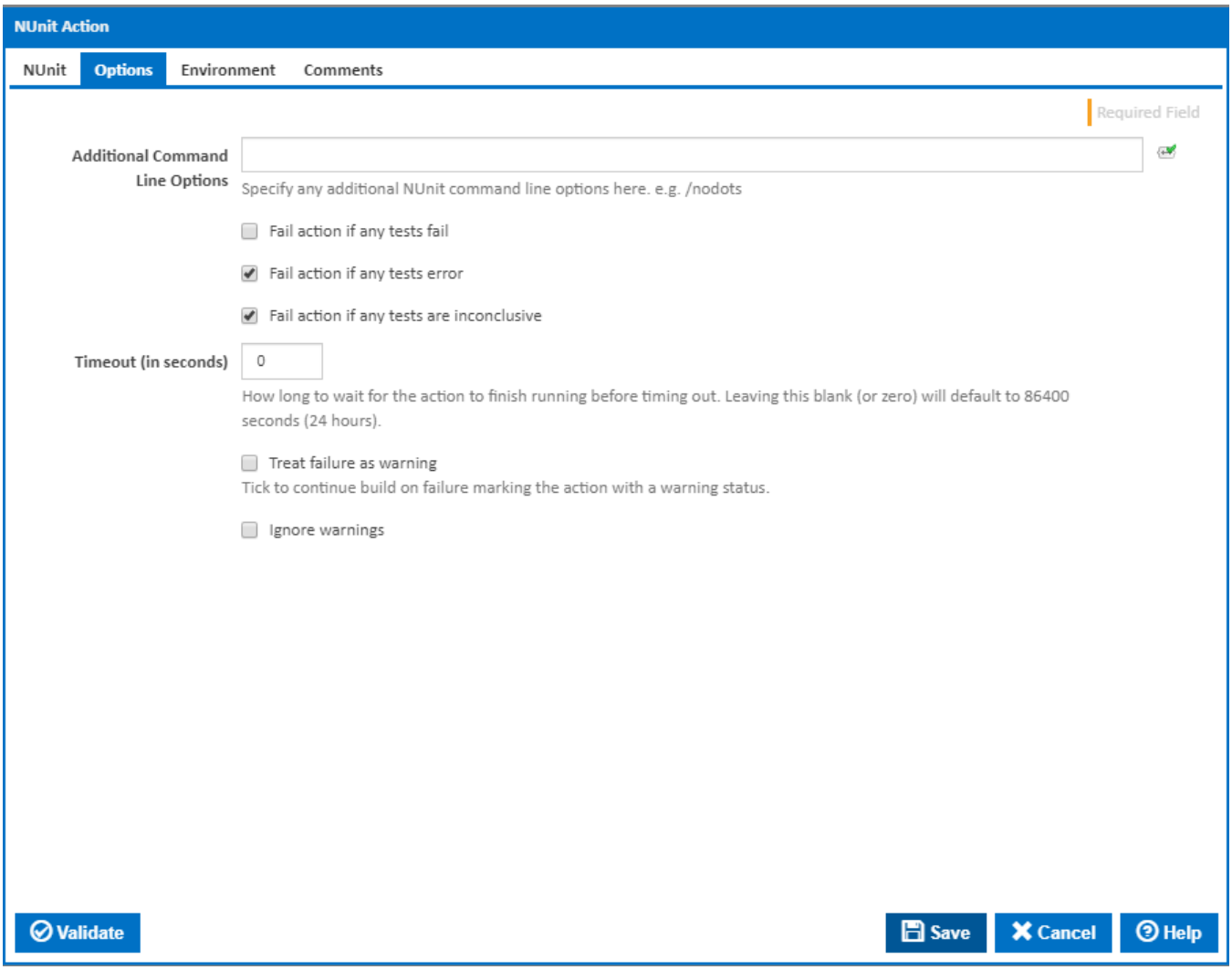
The Using drop down is populated by any property collector properties whose namespace matches the pattern defined by the NuGet action. The pattern for this action is `^NuGet\..*` . The default property collector searches the environment path for "NuGet.exe".

If you create a property collector for this action, make sure you select the **Path Finder PlugIn** type and give it a name that will match the pattern above in blue.

For more in-depth explanations on property collectors see Property Collectors.

Alternatively, you can select the **Custom** option from the Using drop down list and specify a path in the resulting input field that will be displayed. Please read Why it's a good idea to use a property collector before using this option.

Options



Additional Command Line Options

Optional - Additional command line arguments which will be passed to NUnit.

Fail action if any tests fail

Tick this to cause the build to fail if any tests fail.

Fail action if any tests error

Tick this to cause the build to fail if an error occurred while running any test.

Fail action if any tests are inconclusive

Tick this to cause the build to fail if any tests are inconclusive.

Timeout (in seconds)

How long to wait for the action to finish running before timing out. Leaving this blank (or zero) will default to 86400 seconds (24 hours).

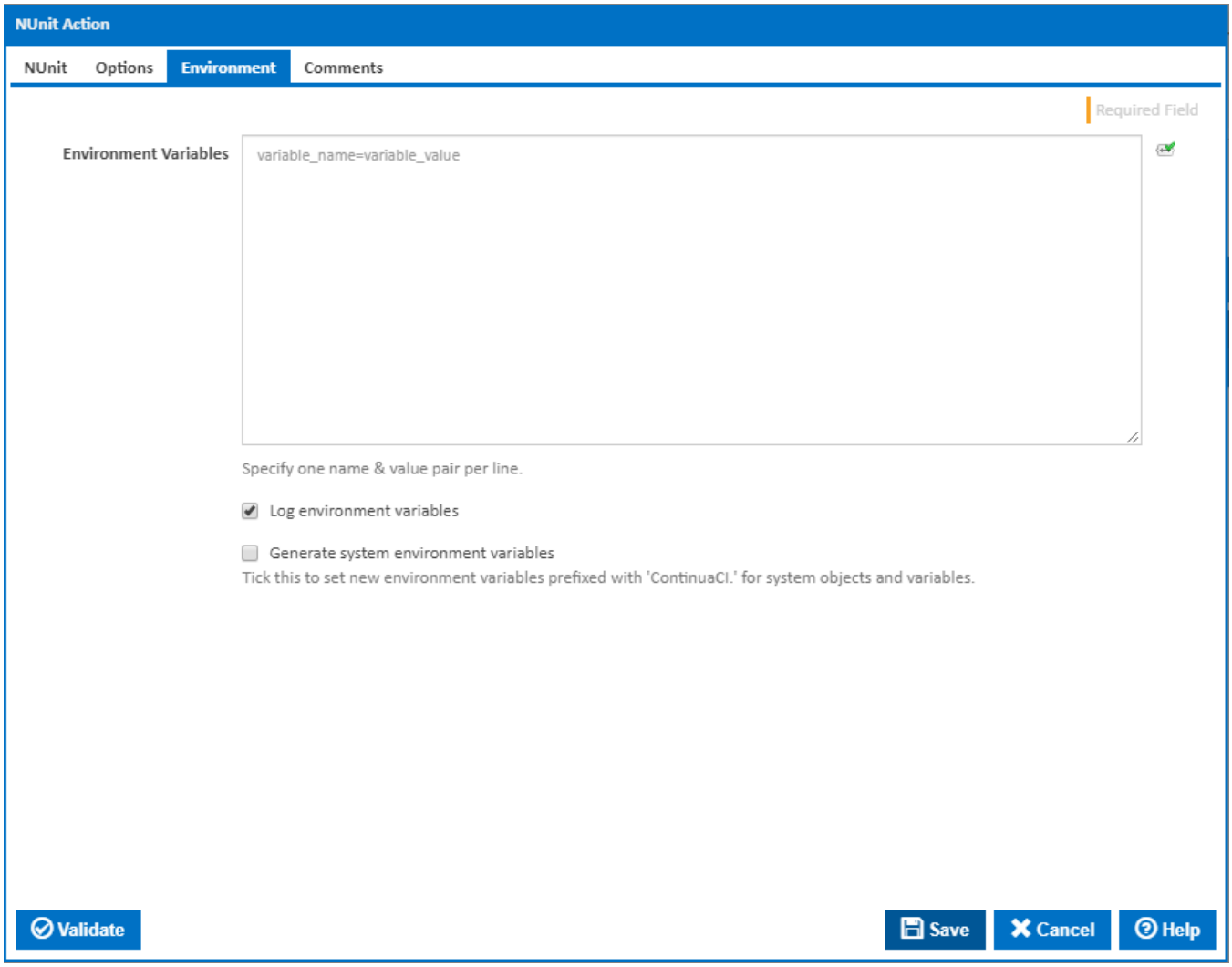
Treat failure as warning

Tick to continue build on failure marking the action with a warning status.

Ignore warnings

If this is ticked, any warnings logged will not mark the action with a warning status.

Environment



Environment Variables

Multiple environment variables can be defined - one per line. These are set before the command line is run.

Log environment variables

If this is ticked, environment variable values are written to the build log.

Generate system environment variables

Tick this checkbox to set up a list of new environment variables prefixed with 'ContinuaCI.' for all current system expression objects and variables.

Mask sensitive variable values in system environment variables

This checkbox is visible only if the '**Generate system environment variables**' checkbox is ticked.

If this is ticked, the values of any variables marked as sensitive will be masked with **** when setting system environment variables. Clear this to expose the values.