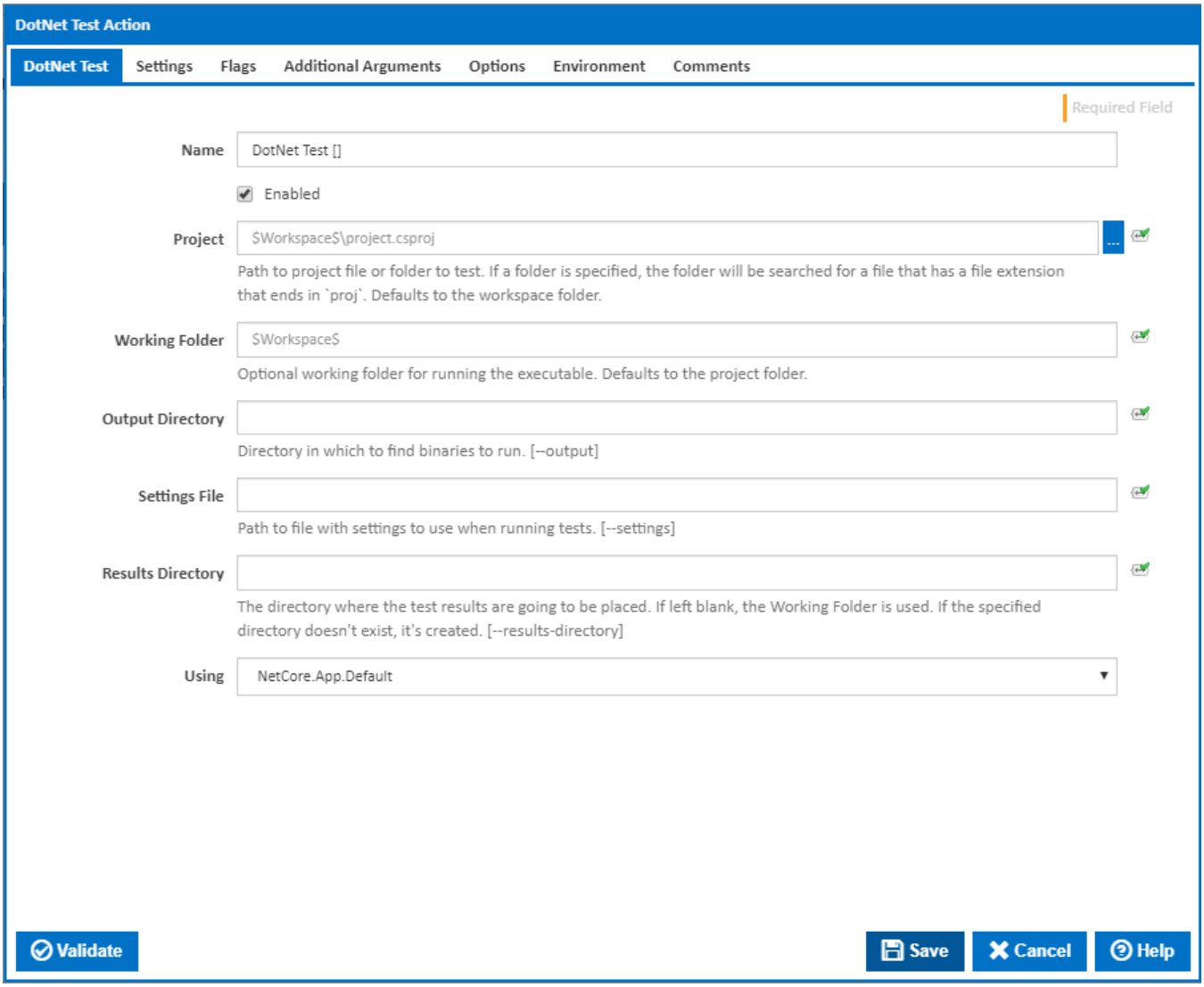
DotNet Test Action

The DotNet Test action in Continua CI is a wrapper around the .Net Core command line tools. If you're having trouble using the DotNet Test action, please refer to the .NET Core Command Line Tools documentation.

The DotNet Test action is used to run unit tests on a .Net project using the configured test runner.

DotNet Test



Name

A friendly name for this action (will be displayed in the actions workflow area).

Enabled

Determines if this action will be run within the relevant stage.

Project

Path to project file or folder to test. If a folder is specified, the folder will be searched for a file that has a file extension that ends in `proj`. Defaults to the workspace folder.

Working Folder

Optional working folder for running the executable. Defaults to the project folder.

Output Directory

The path to the folder in which to find the binaries to run the tests. Relative paths will be anchored to the workspace folder. If left empty, it will default to "/bin/[configuration]/[framework]/" for portable applications or "/bin/[configuration]/[framework]/[runtime]" for self-contained applications. [--output]

Settings File

The path to file with settings to use when running tests. [--settings]

Results Directory

The directory where the test results are going to be placed. If left blank, the Working Folder is used. If the specified directory doesn't exist, it's created. [--results-directory]

Using

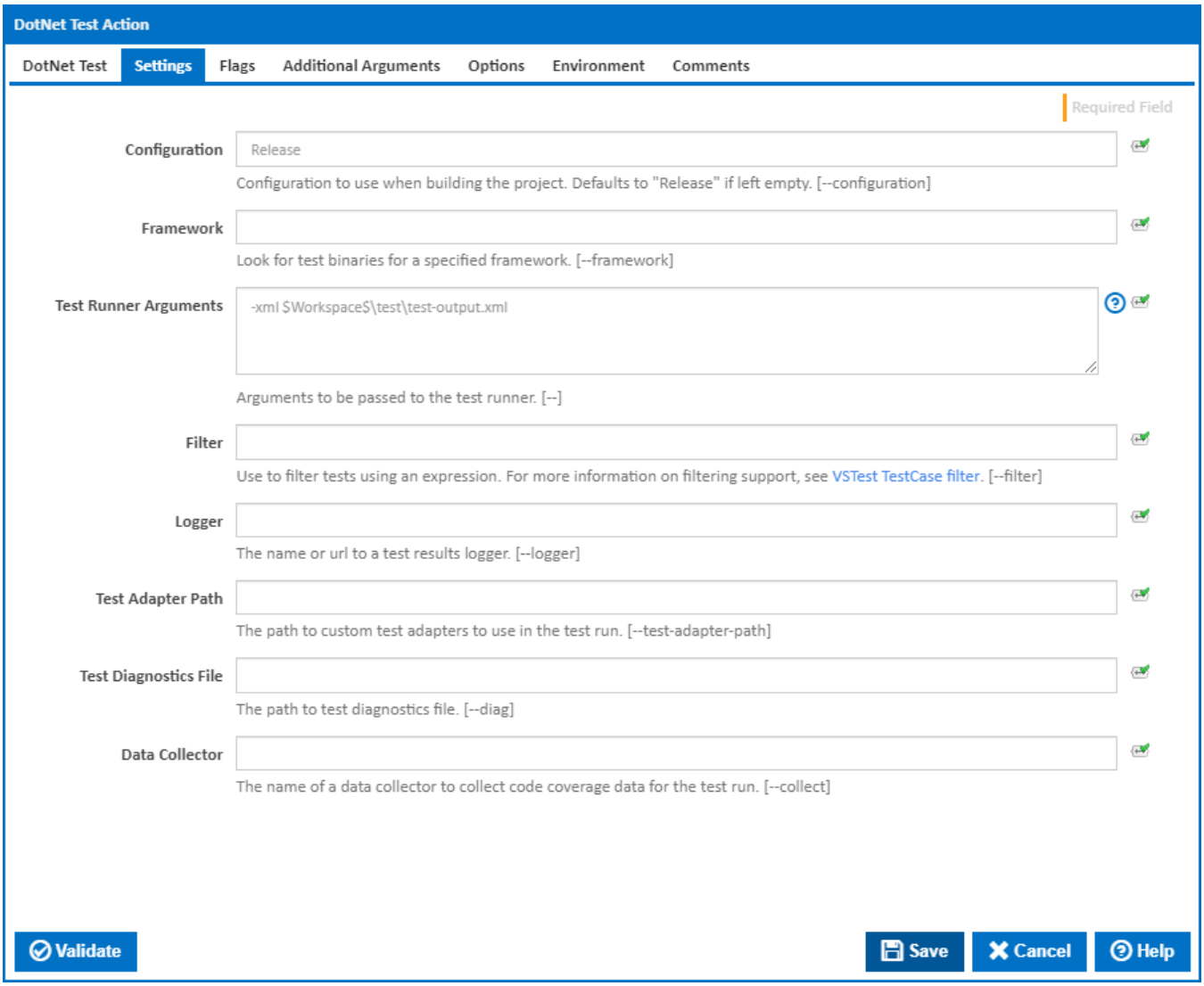
The Using drop down is populated with any property collector whose namespace matches the pattern defined by the DotNet CLI actions. The pattern for this action is `^DotNet.Cli.*`

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 "**DotNet Test**".

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.

Settings



Configuration

The configuration to use when building the project. This defaults to "Debug" if left empty. [--configuration]

Framework

Look for test binaries for a specified framework. [--framework]

Test Runner Arguments

Arguments to be passed to the test runner. The relevant arguments depend on the test runner specified in the project.json file. [--]

> To display the test results in Continua CI, you need to export the test results as a file an use one of the Import Tests actions to read the results from the file. Use this field to specify the path to the results file using the relevant argument for the test runner you are using e.g. -xml for XUnit and --resultxml for NUnit.

Filter

Use to filter tests using an expression. For more information on filtering support, see VSTest TestCase filter. [--filter]

Logger

The name or URL to a test results logger. [--logger]

Test Adapter Path

The path to custom test adapters to use in the test run. [--test-adapter-path]

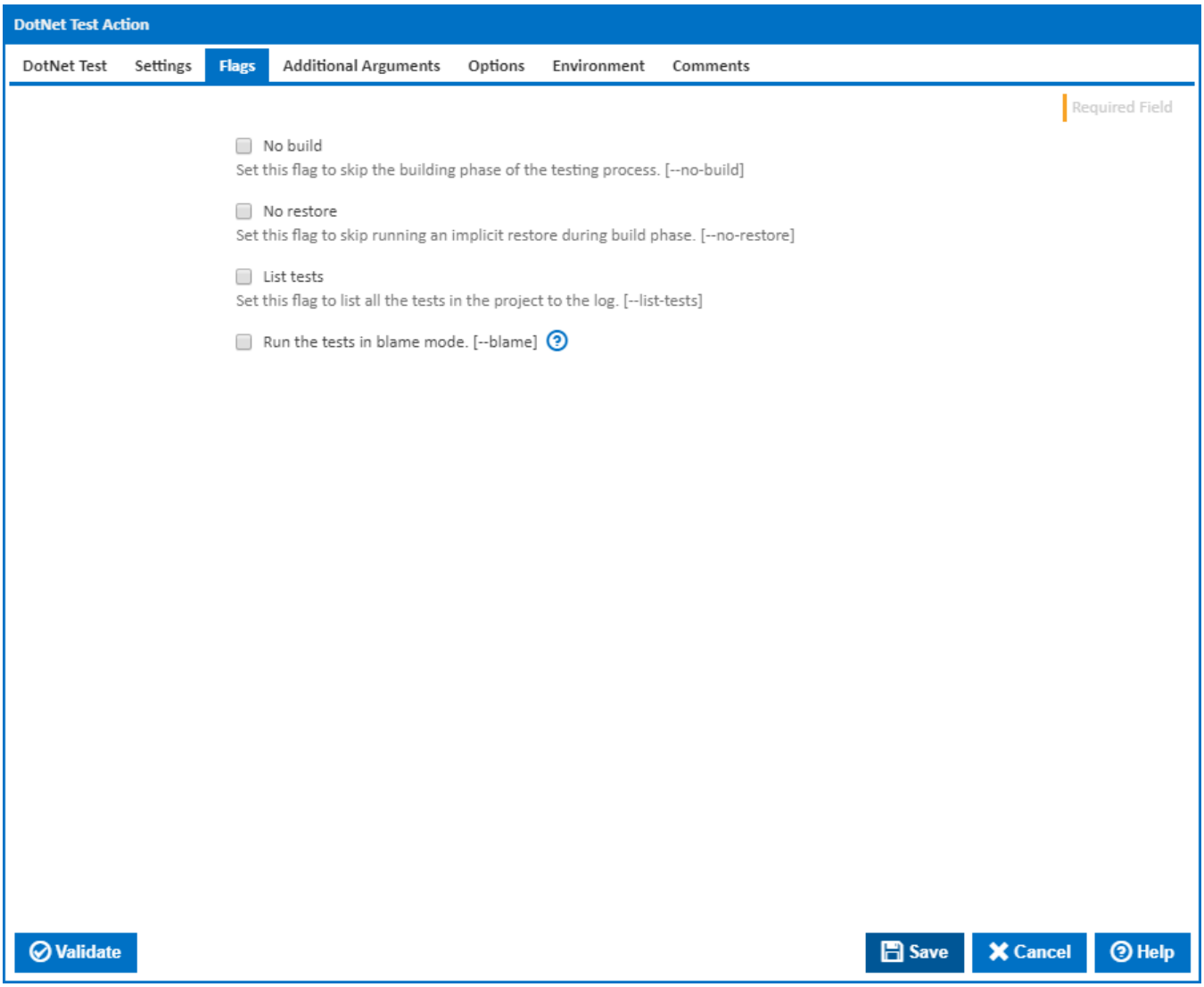
Test Diagnostics File

The path to test diagnostics file. [--diag]

Data Collector

The name of a data collector to collect code coverage data for the test run. [--collect]

Flags



No build

Set this flag to skip the building phase of the testing process. [--no-build]

No Restore

Set this flag to skip running an implicit restore during build. [--no-restore]

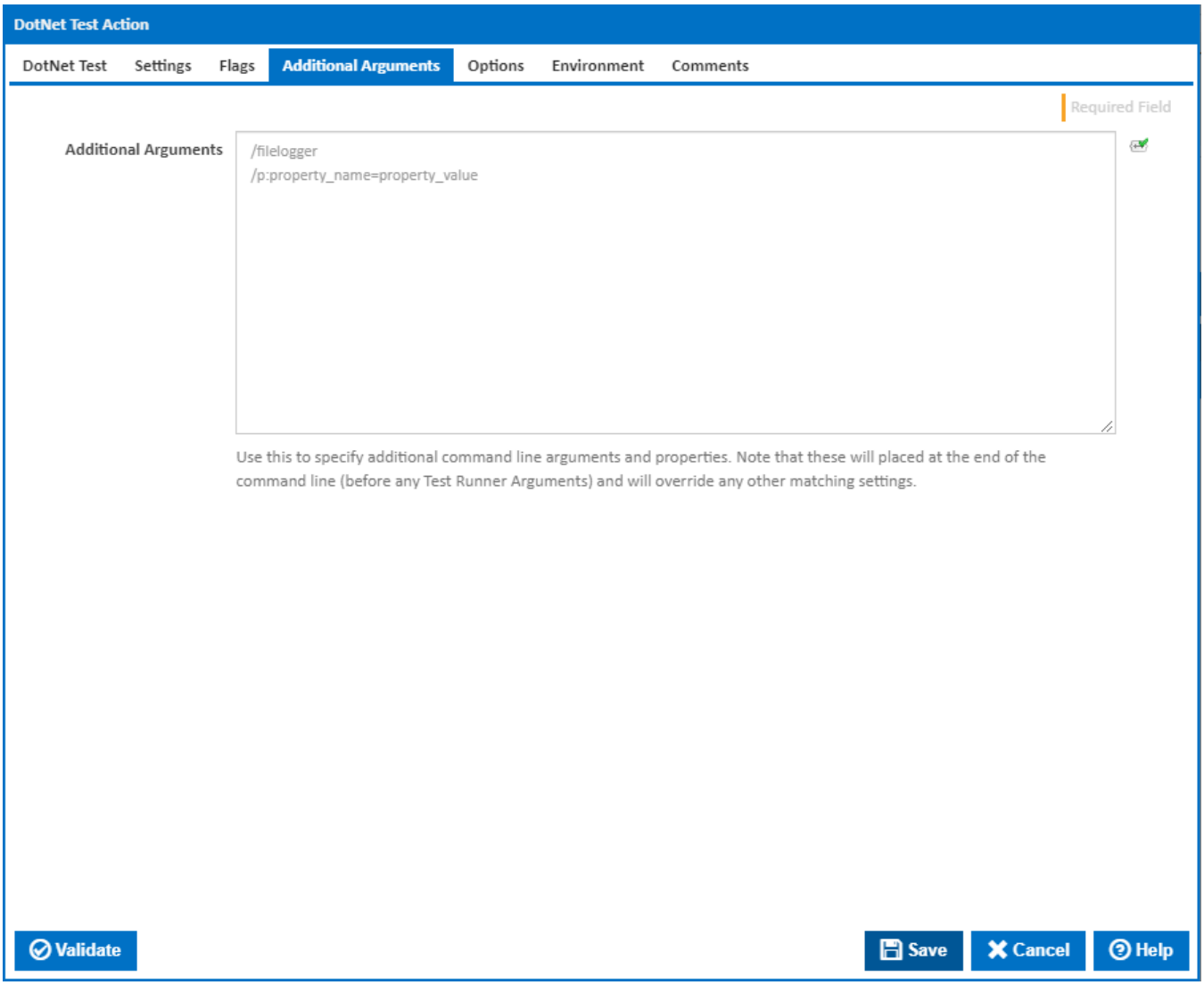
List tests

Set this flag to list all the tests in the project to the log. [--list-tests]

Run the tests in blame mode.

This option is helpful in isolating the problematic tests causing test host to crash. It creates an output file in the working directory as *Sequence.xml* that captures the order of tests execution before the crash. [--blame]

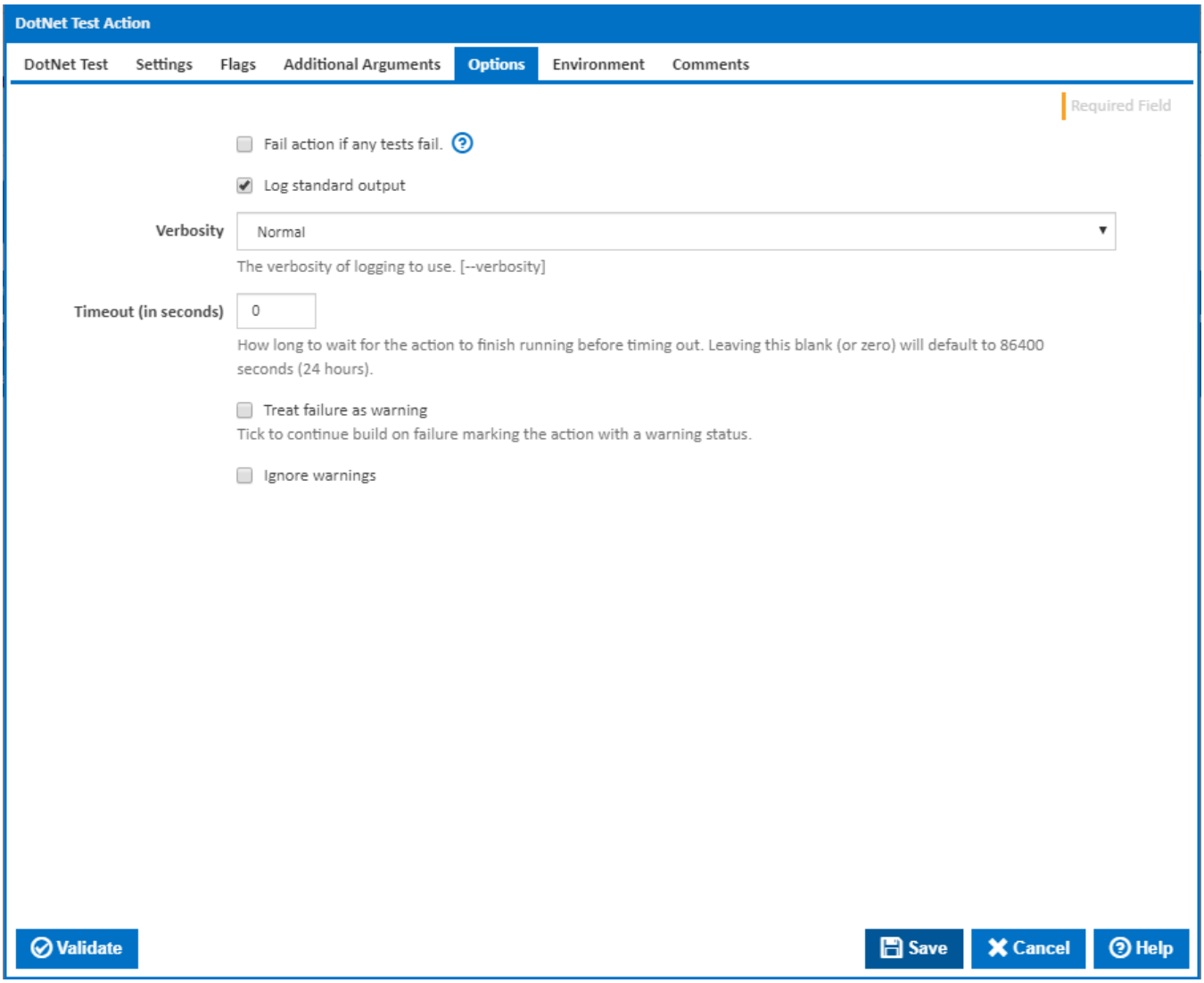
Additional Arguments



Additional Arguments

Use this to specify additional MSBuild command line arguments and properties. Note that these will placed at the end of the command line and will override any other matching settings.

Options



Fail action if any tests fail

Don't tick this if you want to use a Import Unit Tests action to report on test output.

Log standard output

If this is ticked, the command line output is written to the build log.

Verbosity

The amount of information detail to display in the build log. [--verbosity]

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).

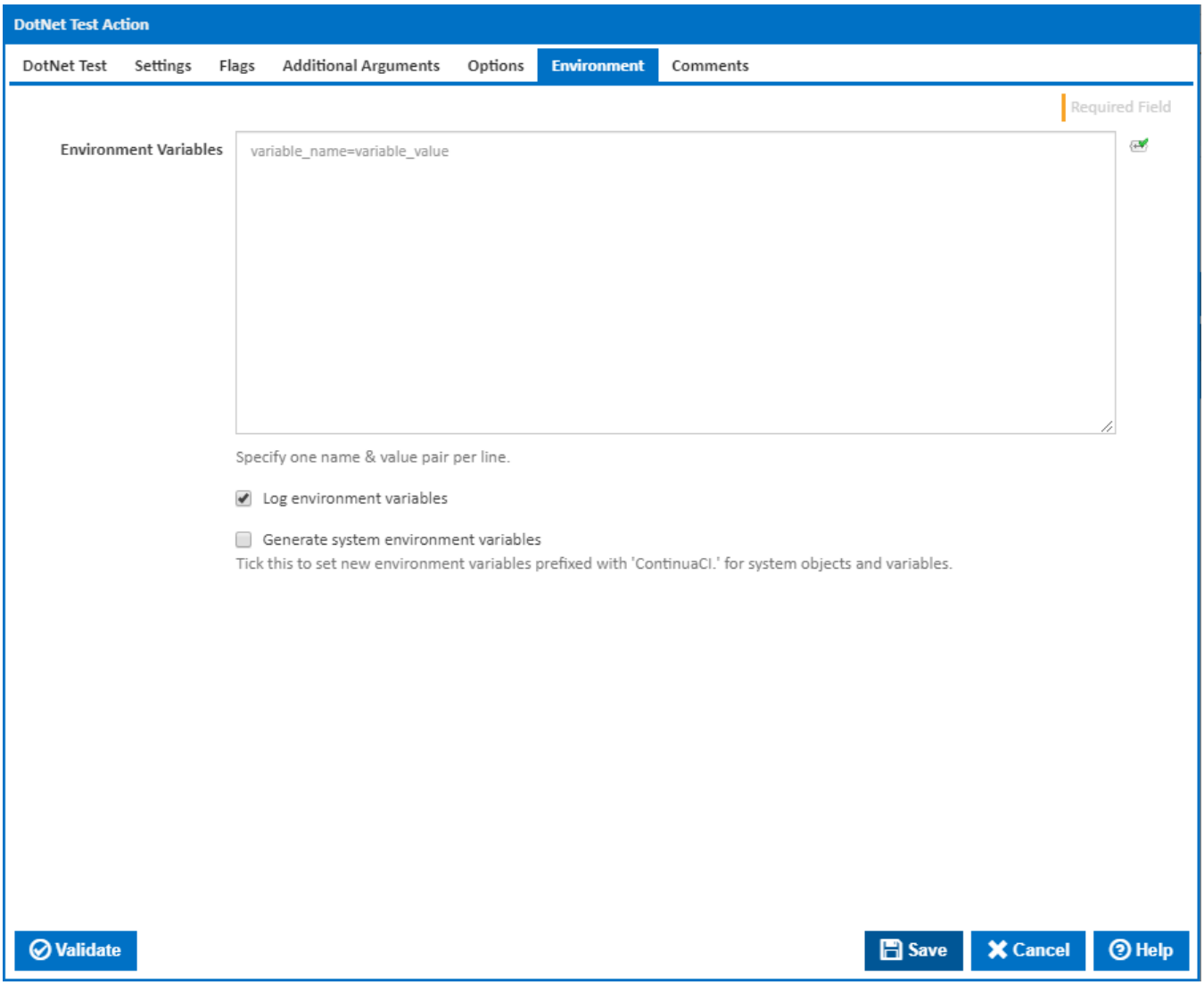
Treat failure as warning

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.