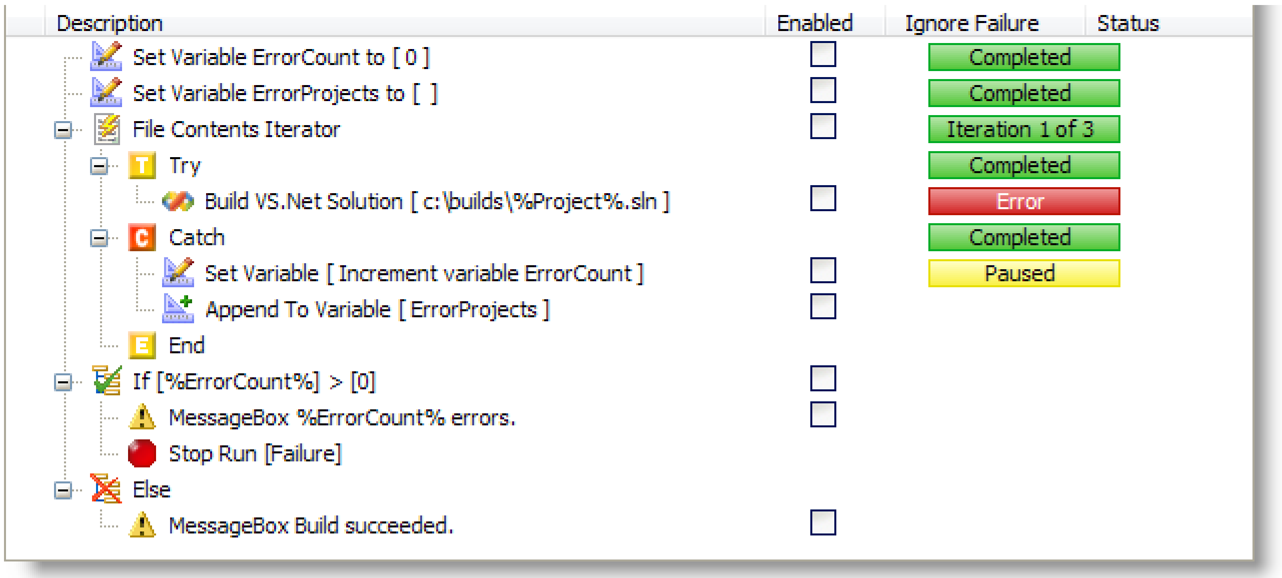
Counting Errors With Try/Catch Blocks

One good use of Try/Catch blocks is to record information about errors, in order to generate a report later in the build. There are several advantages to doing this:

- The build doesn't abort at the first error, so you get more information if several projects fail to compile.
- You can recover from minor errors
- You can treat each error differently. For example, build errors are emailed to developers, deployment errors emailed to operations.
- You can record your own statistics and logs.

Here's a simple example which builds all the projects listed in a text file. The variable ErrorCount records the total number of errors, while ErrorProjects builds a list of the projects which have failed.



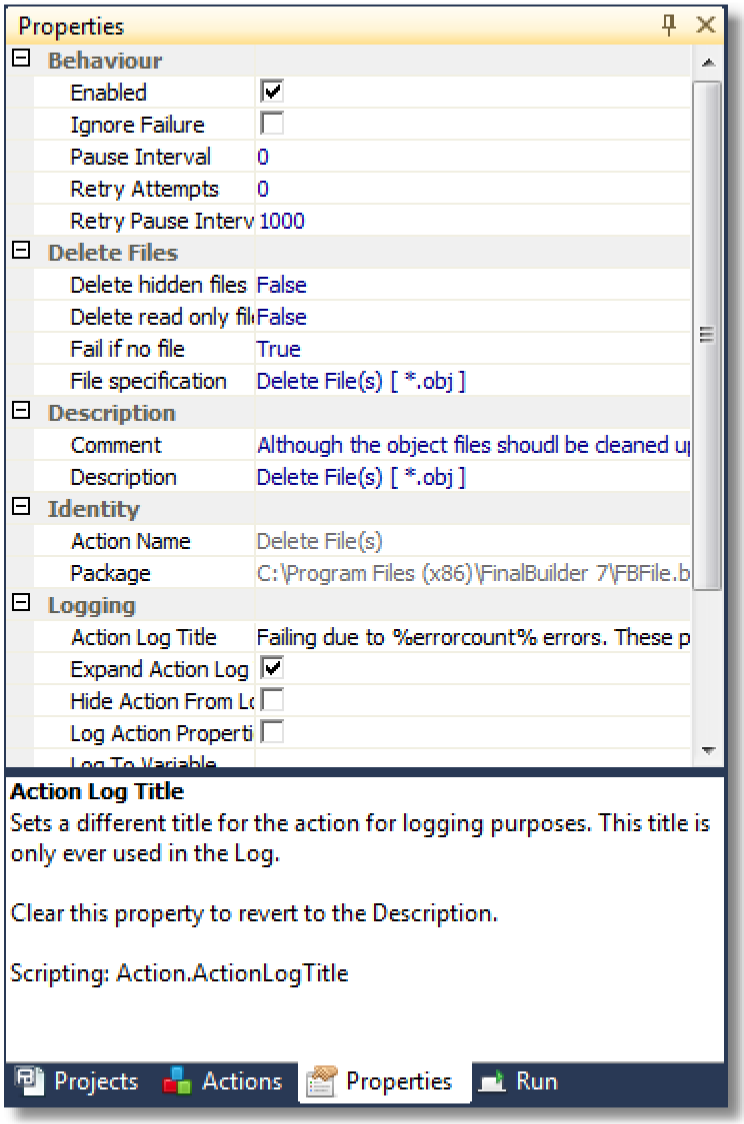
Description	Enabled	Ignore Failure	Status
Set Variable ErrorCount to [0]	☐		Completed
Set Variable ErrorProjects to [ ]	☐		Completed
File Contents Iterator	☐		Iteration 1 of 3
Try			Completed
Build VS.Net Solution [c:\builds\%Project%.sln]	☐		Error
Catch			Completed
Set Variable [Increment variable ErrorCount]	☐		Paused
Append To Variable [ErrorProjects]	☐		
End			
If [%ErrorCount%] > [0]	☐		
MessageBox %ErrorCount% errors.	☐		
Stop Run [Failure]			
Else			
MessageBox Build succeeded.	☐		

The steps are as follows:

1. Initialise the two variables.
2. Iterate over the contents of the file.
3. Use a Try action to wrap around the Build VS.Net Solution action. If the compilation succeeds, the Catch part is not run.
4. If the compilation fails, the Catch part is run: the ErrorCount variable is incremented, and the ErrorProjects variable is appended to. The build then continues on the next loop of the iterator.
5. After all the projects are built, a message is shown if there was at least one error. We then use a Stop Run action to signal that the build as a whole has failed.
6. If there was no error, a different message is shown. By default, builds terminate with a success code, so we don't need a Stop Run action here.

More ideas:

- Instead of showing a message, you could record the count and list of failed projects to a text file.
- To gain more information about any error, you could use Log to Variable. See the Analysing Output tutorial.
- You can use Try/Catch blocks at a very high level, wrapping calls to Action Lists or even other projects with the Include Project action.
- Set a custom Action Log Title on the Stop Run action to explain why the build is stopping:



Properties

Behaviour	
Enabled	✓
Ignore Failure	☐
Pause Interval	0
Retry Attempts	0
Retry Pause Interv	1000
Delete Files	
Delete hidden files	False
Delete read only fil	False
Fail if no file	True
File specification	Delete File(s) [*.obj]
Description	
Comment	Although the object files shoudl be cleaned u
Description	Delete File(s) [*.obj]
Identity	
Action Name	Delete File(s)
Package	C:\Program Files (x86)\FinalBuilder 7\FBFile.b
Logging	
Action Log Title	Failing due to %errorcount% errors. These p
Expand Action Log	✓
Hide Action From Lo	☐
Log Action Properti	☐
Log To Variable	

Action Log Title
Sets a different title for the action for logging purposes. This title is only ever used in the Log.

Clear this property to revert to the Description.

Scripting: Action.ActionLogTitle