

# Command Reference: RunProgram()

## Run an external program

Version 11.08.01, 2016-02-15

The `RunProgram()` command runs an external program, given the full command line or individual command line parts, and waits until the program is finished before processing additional commands. The TSTool command will indicate a failure if the exit status from the program being run is non-zero. It is therefore possible to call an external program that reads and/or writes recognized time series formats to perform processing that TSTool cannot. One use of this command is to create a calibration environment where a model is run and then the results are read and displayed using TSTool. It is also useful to use TSTool's testing features to implement quality control checks for other software tools.

TSTool internally maintains a working directory that is used to convert relative paths to absolute paths to locate files. The working directory is by default the location of the last command file that was opened. The external program may assume that the working directory is the location from which TSTool software was started (or the installation location if started from a menu). Therefore, it may be necessary to run TSTool in batch mode from the directory where the external software's data files exist, use absolute paths to files, or use the `${WorkingDir}` property in the command line. Use `\` in the command line or arguments to surround whitespace. Some operating systems may have limitations on command line length. The following dialog is used to edit the command and illustrates the command syntax.

RunProgram() Command Editor when Specifying Command Line in Full

Command (full)	Command (parts)	Command shell	Timeout	Exit Code Check	Output Checks
<p>Sometimes it is necessary to specify the command in parts so that separation between command line parts is explicit. Use the following parameters to provide the command as parts.</p>					
Program to run:	echo				Required - if full command line is not specified.
Program argument 1:	Hello				Optional - as needed if Program is specified.
Program argument 2:	>				Optional - as needed if Program is specified.
Program argument 3:	\${WorkingDir}/Results/Test_RunProgram_CommandLine_echo_out.txt				Optional - as needed if Program is specified.
Program argument 4:					Optional - as needed if Program is specified.
Program argument 5:					Optional - as needed if Program is specified.
Program argument 6:					Optional - as needed if Program is specified.
Program argument 7:					Optional - as needed if Program is specified.
Program argument 8:					Optional - as needed if Program is specified.

RunProgram\_Parts

### RunProgram() Command Editor when Specifying Command Line in Parts

Command (full)	Command (parts)	Command shell	Timeout	Exit Code Check	Output Checks
<p>The program by default will be run with a command shell (e.g., cmd.exe on Windows). This is often helpful because it allows the shell to initialize the environment for the program. A shell is necessary if the program to run is a batch file (Windows) or shell script (Linux) that is parsed and run by the shell. Indicate NOT to use a command shell if it is known that the program is an executable (not a shell command or script).</p>					
Use command shell:	<input type="checkbox"/>				Optional - use command shell (default=True).
Command shell:					Optional - shell program to run default depends on system.

RunProgram\_Shell

### RunProgram() Command Editor showing Command Shell Parameters

Command (full)	Command (parts)	Command shell	Timeout	Exit Code Check	Output Checks
<p>TSTool will for the called program to complete before continuing. This allows TSTool to check for errors and process output from the program. Specify the timeout to ensure that TSTool does not wait indefinitely for the program to finish. <b>May add parameters to control whether timeout is treated as an error.</b></p>					
Timeout (seconds):					Optional - default is no timeout.

RunProgram\_Timeout

### RunProgram() Command Editor showing Timeout Parameters

Command (full)	Command (parts)	Command shell	Timeout	Exit Code Check	Output Checks
<p>Specify the exit status indicator if program output messages must be used to determine the program exit status (e.g., "Status:"). <b>This functionality is being reviewed and improved.</b> <b>In the future a table may be added to control how exit code is handled.</b></p>					
Exit status indicator:					Optional - output string to indicate status (default=use process exit status).

RunProgram\_ExitStatusIndicator

### RunProgram() Command Editor showing Exit Code Parameters

Command (full)	Command (parts)	Command shell	Timeout	Exit Code Check	Output Checks
----------------	-----------------	---------------	---------	-----------------	---------------

The program that is called may generate one or more output files that contain information indicating success, warnings, and errors. Use the table to specify files to search for output, and how to interpret text in files as warnings and errors.

Table columns should be:

- File - name of file to search, can contain \${Property}
- Pattern - pattern to match in file, using literal text and \*, can contain \${Property}
- Level - for match, level of message, one of: Status, Warning, Failure
- Message - message for TSTool command status, use \${TBD} to include output line, can contain \${Property}
- Recommendation - recommendation for TSTool command status, can contain \${Property}

**Need to figure out syntax to pass back content from file, maybe even multiple lines.**

Output check table ID:  Optional - table with output check information.

RunProgram\_OutputChecks

### RunProgram() Command Editor showing Output Check Parameters

The command syntax is as follows:

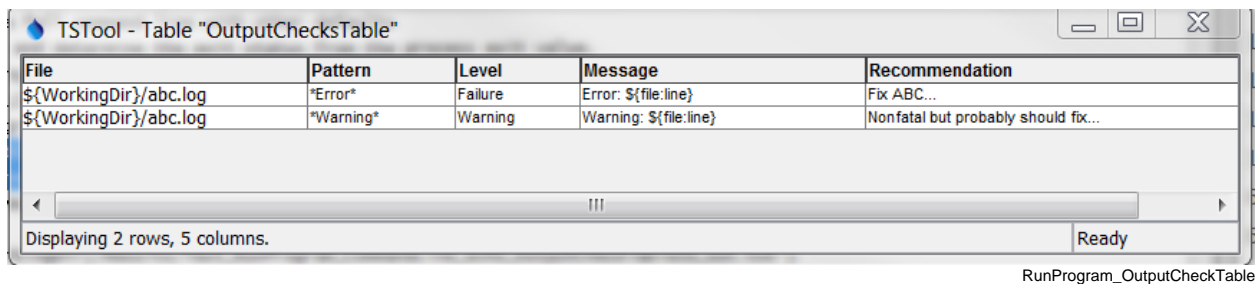
```
RunProgram(Parameter=Value...)
```

### Command Parameters

Parameter	Description	Default
CommandLine	<p>The full program command line, with arguments. If the program executable is found in the PATH environment variable, then only the program name needs to be specified. Otherwise, specify an absolute path to the program or run TSTool from a command shell the same directory.</p> <p>The \${WorkingDir} property can be used in the command line to indicate the working directory (command file location) when specifying file names.</p> <p>For Windows, it may be necessary to place a \" at the start and end of the command line, if a full command line is specified.</p>	<p>Must be specified if the Program parameter is not specified.</p> <p>The Program parameter will be used if both are specified.</p>
Program	The name of the program to run. Program arguments are specified using the ProgramArg# parameter(s). See the CommandLine parameter for more information about parameter formatting and locating the executable.	Must be specified if the CommandLine parameter is not specified.
ProgramArg1, ProgramArg2, etc.	Command like arguments used with Program. If necessary, use \${WorkingDir} to specify the working directory to locate files.	No arguments will be used with Program.
UseCommandShell	If specified as False, the program will be run without using a command shell. A command shell is needed if the program is a script (batch file), a shell command, or uses >,  , etc.	True, using cmd.exe /C on Windows and /bin/sh -c on UNIX/Linux.
CommandShell	The command shell program to run.	Determine automatically.

Parameter	Description	Default
Timeout	The timeout in seconds – if the program has not yet returned, the process will be ended. Zero indicates no timeout. <b>This behavior varies and is being enhanced.</b>	No timeout.
ExitStatus Indicator	By default, the program exit status is determined from the process that is run. Normally 0 means success and non-zero indicates an error. However, the program may not exit with a non-zero exit status when an error occurs. If the program instead uses an output string like STOP 3 to indicate the status, use this parameter to indicate the leading string, which is followed by the exit status (e.g., STOP).	Determine the exit status from the process exit value.
OutputCheck TableID	Output file content can be scanned for patterns to detect success, warning, and errors. See the example file below for syntax of the table.	Output is not checked

The following figure illustrates the output check table specified by the OutputCheckTableID command.



File	Pattern	Level	Message	Recommendation
\${WorkingDir}/abc.log	*Error*	Failure	Error: \${file:line}	Fix ABC...
\${WorkingDir}/abc.log	*Warning*	Warning	Warning: \${file:line}	Nonfatal but probably should fix...

Displaying 2 rows, 5 columns. Ready

Example Output Check Table

The table columns are described below. The column names must be adhered to.

#### Output Check Table Column Descriptions

Parameter	Description
File	Name of the file to check. Use \${WorkingDir} to specify the location of the working directory (folder for command file). Can use the \${Property} notation.
Pattern	The pattern to search for in the file. Can use * for wildcard. Searches are case-insensitive. <b>Is there a need for more pattern control?</b>
Level	The message level to use in TSTool command status messages: Success, Warning, or Failure.
Message	The message to include in TSTool command status messages, generally the cause of the issue. Use \${file:line} to output the line from the output file. <b>What if error messages are multi-line?</b>
Recommendation	A recommendation to fix the problem, as shown in TSTool command status messages.

The following legacy functionality is being reviewed, in particular to clarify where the status string is printed (standard output, etc.).

The following figure illustrates how a command can be run without a command shell and using the program output to determine the exit status. The *testecho.exe* program is a compiled executable and can therefore be run without a command shell. Because the standard output is being evaluated for the exit value, the output cannot be redirected to a file with `>` (this would result in no output being available to TSTool to evaluate), and `>` is only recognized if running with a command shell in any case.

The following approach is suitable, for example, when running a compiled model or data analysis tool. However, if the tool is run using a script or batch file, then a command shell must be used.

**Edit RunProgram() command**

This command runs another program, and TSTool waits for it to complete before continuing.  
 Commands must use a full path to files, TSTool must be started from the directory where files exist to use relative paths, or use `${WorkingDir}` in the command line to specify files relative to the working directory.  
 Use `'` to indicate double quotes if needed to surround program name or program command-line parameters - this may be needed if there are spaces in paths.  
 Specify the exit status indicator if program output messages must be used to determine the program exit status (e.g., "Status:").  
 Specify the program to run using the command line OR separate arguments - the latter makes it simpler to know how to treat whitespace in command line arguments.  
 The program by default will be run with a command shell (e.g., `cmd.exe` on Windows) - specify as `False` if it is known that the program is an executable (not a shell command or script).

Command to run (with arguments):

Program to run:  Required - if full command line is not specified above.

Program argument 1:  Optional - as needed if Program is specified.

Program argument 2:  Optional - as needed if Program is specified.

Program argument 3:  Optional - as needed if Program is specified.

Program argument 4:  Optional - as needed if Program is specified.

Program argument 5:  Optional - as needed if Program is specified.

Program argument 6:  Optional - as needed if Program is specified.

Program argument 7:  Optional - as needed if Program is specified.

Program argument 8:  Optional - as needed if Program is specified.

Use command shell:  Optional - use command shell (default=True).

Timeout (seconds):  Optional - default is no timeout.

Exit status indicator:  Optional - output string to indicate status (default=use process exit status).

Command:

```
RunProgram(Program="${WorkingDir}/testecho.exe", ProgramArg1="STOP 2", ExitStatusIndicator="STOP")
```

Cancel OK

RunProgram\_Program\_ExitStatusIndicator

**RunProgram() Command Editor when Specifying Program, Arguments, and Exit Status Indicator**

This page is intentionally blank.