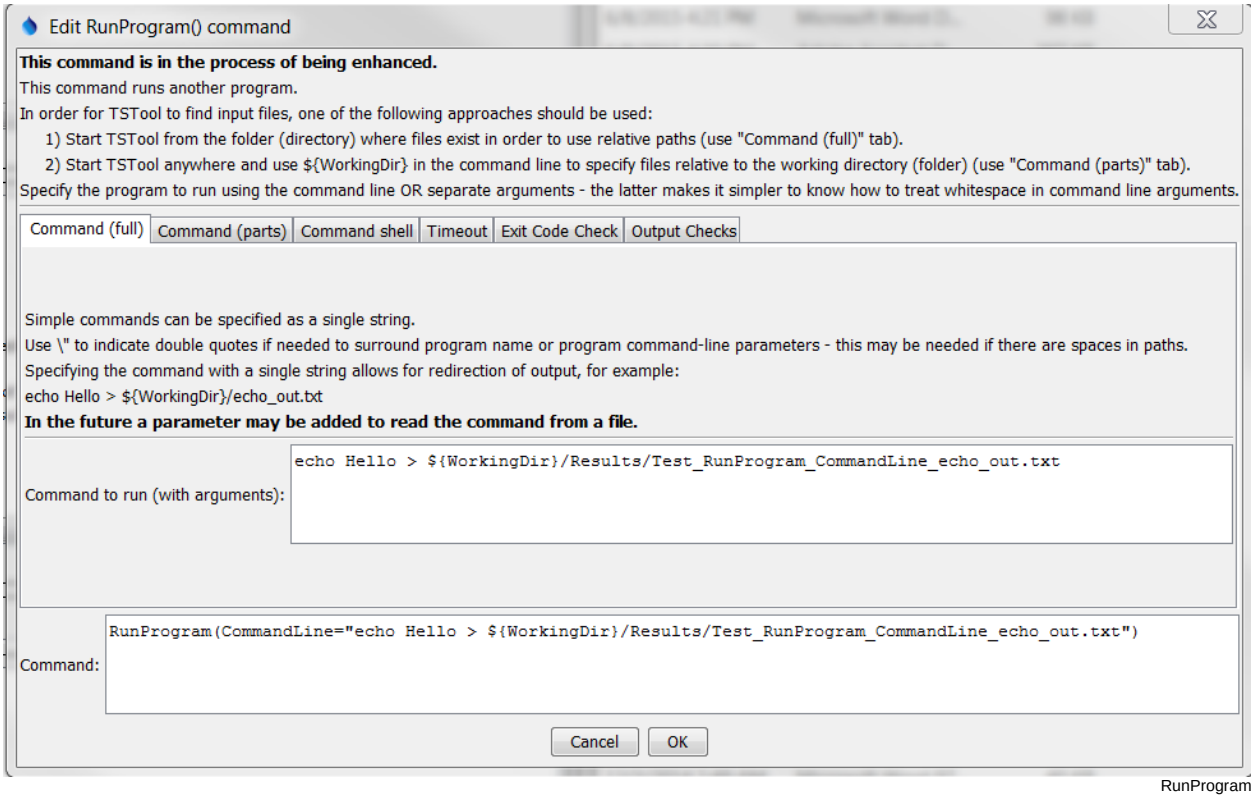
Command Reference: RunProgram()

Run an external program

Version 11.08.01, 2016-02-15

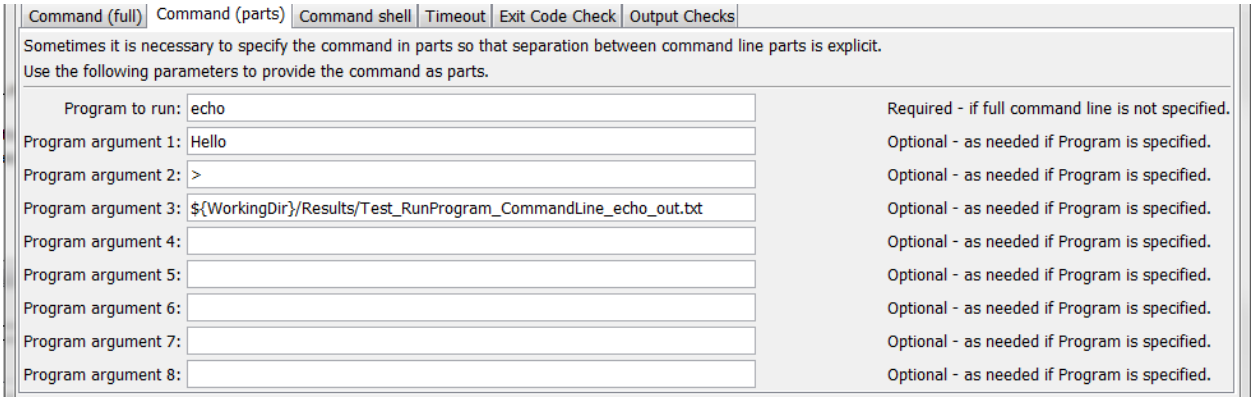
The `RunProgram()` command runs an external program, given the full command line or individual command line parts, and waits until the program is finished before processing additional commands. The TSTool command will indicate a failure if the exit status from the program being run is non-zero. It is 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.



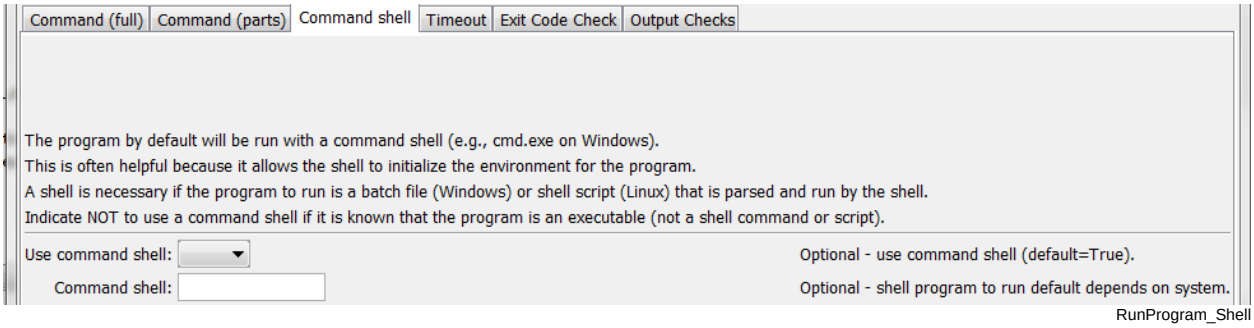
RunProgram

RunProgram() Command Editor when Specifying Command Line in Full



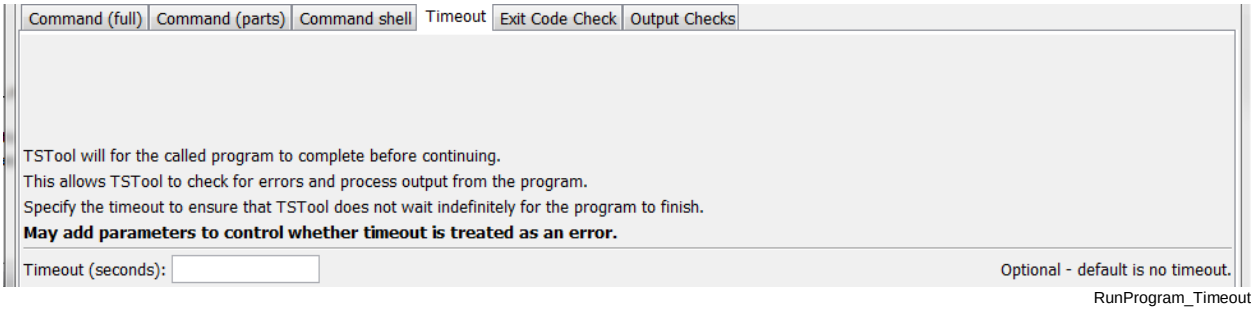
RunProgram_Parts

RunProgram() Command Editor when Specifying Command Line in Parts



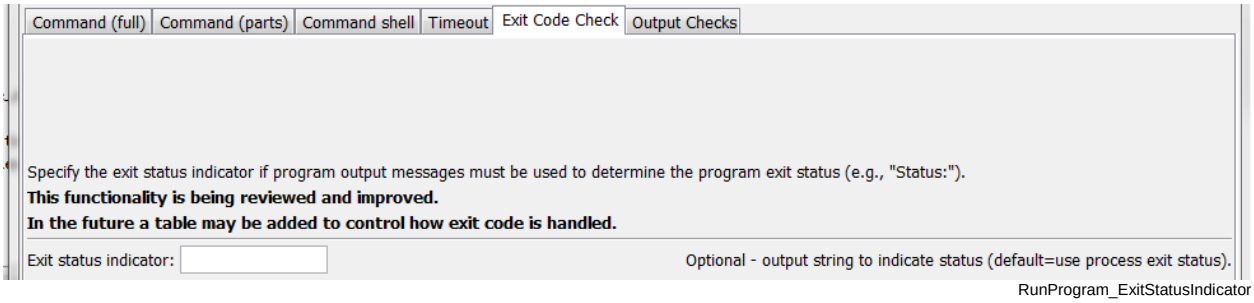
RunProgram_Shell

RunProgram() Command Editor showing Command Shell Parameters



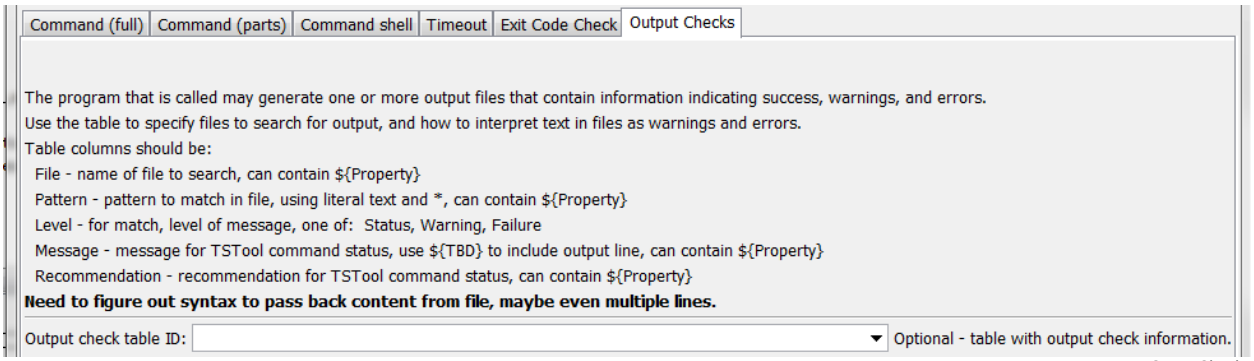
RunProgram_Timeout

RunProgram() Command Editor showing Timeout Parameters



RunProgram_ExitStatusIndicator

RunProgram() Command Editor showing Exit Code Parameters



RunProgram_OutputChecks

RunProgram() Command Editor showing Output Check Parameters

The command syntax is as follows:

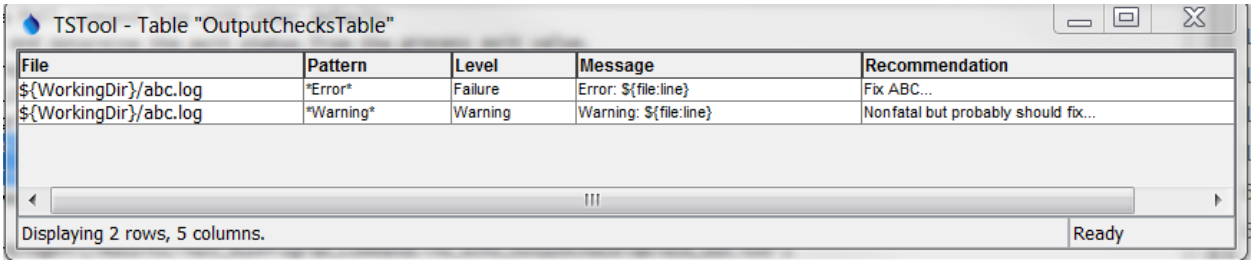
```
RunProgram(Parameter=Value…)
```

Command Parameters

Parameter	Description	Default
`CommandLine`	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. The `${WorkingDir}` property can be used in the command line to indicate the working directory (command file location) when specifying file names. For Windows, it may be necessary to place a `\"` at the start and end of the command line, if a full command line is specified.	Must be specified if the `Program` parameter is not specified. The `Program` parameter will be used if both are specified.
`Program`	The name of the program to run. Program arguments are specified using the `ProgramArg#` parameter(s). See the `CommandLine` parameter for more information about parameter formatting and locating the executable.	Must be specified if the `CommandLine` parameter is not specified.
`ProgramArg1`, `ProgramArg2`, `etc.`	Command like arguments used with `Program`. If necessary, use `${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. **This behavior varies and is being enhanced**.	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.



RunProgram_OutputCheckTable

Example Output Check Table

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

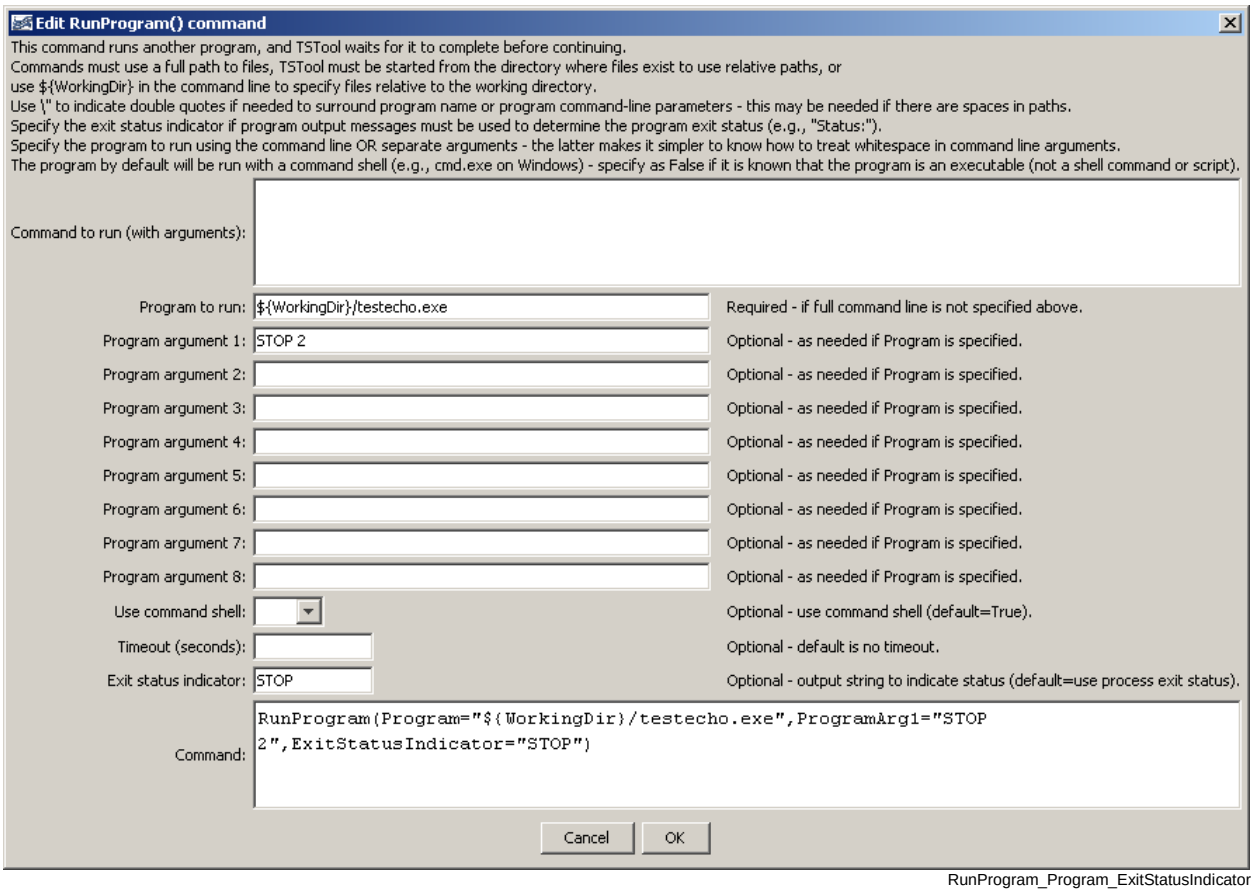
Output Check Table Column Descriptions

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. Is there a need for more pattern control?
`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. What if error messages are multi-line?
`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.



RunProgram_Program_ExitStatusIndicator

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

This page is intentionally blank.