
Command Reference: RunCommands()

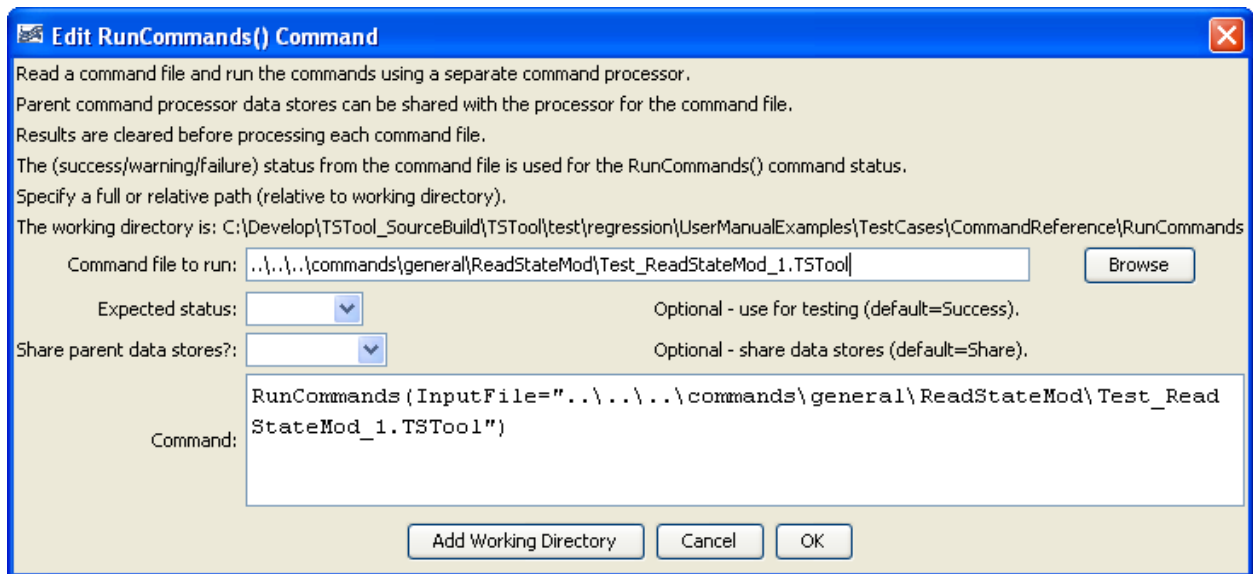
Run a command file

Version 11.04.02, 2015-07-10

The `RunCommands()` command runs a command file using a separate command processor as a “child” of the main processor. This command can be used to manage workflow where multiple commands files are run, and is also used extensively for testing, where a test suite consists of running separate test case command files.

- Command files that are run can themselves include `RunCommands()` commands. Each command file that is run has knowledge if its initial working directory and relative paths referenced in the command file are relative to this directory. This allows a master command file to reside in a different location than the individual command files that are being run. The current working directory is reset to that of the command file being run.
- Data stores from the parent command processor are by default passed to the child command processor, which allows database connections can be opened once and shared.
- **Currently the properties from the parent command file are NOT applied to the initial conditions when running the command file.** Therefore, global properties like input and output period are reset to defaults before running the command file. A future enhancement may implement a command parameter to indicate whether to share the properties with the parent processor. The output from the command is also not added to the parent processor. Again, a future enhancement may be to append output so that one final set of output is generated.
- There is currently no special handling of log files; consequently, if the main command file opens a log file and then a command file is run that opens a new log file, the main log file will be closed. This behavior is being evaluated.

The following dialog is used to edit the command and illustrates the syntax of the command.



Edit RunCommands() Command

Read a command file and run the commands using a separate command processor.
Parent command processor data stores can be shared with the processor for the command file.
Results are cleared before processing each command file.
The (success/warning/failure) status from the command file is used for the RunCommands() command status.
Specify a full or relative path (relative to working directory).
The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\RunCommands

Command file to run:

Expected status: Optional - use for testing (default=Success).

Share parent data stores?: Optional - share data stores (default=Share).

Command:

RunCommands

RunCommands() Command Editor

The command syntax is as follows:

```
RunCommands (Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
InputFile	The name of the command file to run, enclosed in double quotes if the file contains spaces or other special characters. A path relative to the master command file can be specified.	None – must be specified.
Expected Status	Used for testing – indicates the expected status from the command, one of: <ul style="list-style-type: none"> Unknown Success Warning Failure If this parameter is NOT used, the command log messages from commands that are run will be appended to the RunCommands() log. However, using this parameter will not append those messages – this is used in automated testing to allow a successful test even when there are warning and failure messages.	Success
Share DataStores	Indicate whether data stores in the parent should be shared with the child command processor. Normally this should be done so that databases can be opened once. Note that opening data stores in the child command file will not make the data stores available in the parent.	Share

The following example illustrates how the RunCommands() command can be used to test TSTool software (or any implementation of commands). First, individual command files are implemented to test specific functionality, which will result in warnings if a test fails:

```
StartLog(LogFile="Results/Test_ReadStateMod_1.TSTool.log")
NewPatternTimeSeries(NewTSID="MyLoc..MyData.Day",Alias="TS",
  Description="Test data",SetStart="1950-01-01",
  SetEnd="1951-03-12",Units="CFS",PatternValues="5,10,12,13,75")
# Uncomment the following command to regenerate the expected results file.
# WriteStateMod(TSList=AllTS,
#   OutputFile="ExpectedResults\Test_ReadStateMod_1_out.stm")
ReadStateMod(InputFile="ExpectedResults\Test_ReadStateMod_1_out.stm")
CompareTimeSeries(Precision=3,Tolerance=".001",DiffFlag="X",
  WarnIfDifferent=True)
```

Next, use the RunCommands() command to run one or more tests:

```
StartRegressionTestResultsReport(
  OutputFile="RunRegressionTest_commands_general.TSTool.out.txt")
RunCommands(InputFile="..\..\..\commands\general\ReadStateMod\Test_ReadStateMod_1.TS
Tool")
```

Each of the above command files should produce expected results, without warnings. If any command file unexpectedly produces a warning, a warning will also be visible in TSTool. The issue can then be evaluated to determine whether a software or configuration change is necessary. See the StartRegressionTestResultsReport() command documentation for an explanation of how to disable a command file with #@enabled False or indicate its expected status for testing (e.g., #@expected Status Warning).