# Command Reference: CreateRegressionTestCommandFile()

## Create a command file to run software regression tests

Version 10.20.00, 2013-04-20

The `CreateRegressionTestCommandFile()` command is used for software testing and certification of processes used in operations. The command creates a command file that includes a `StartRegressionTestResultsReport()` and multiple `RunCommands()` commands. A starting search folder is provided and all files that match the given pattern (by convention *Test_*.TSTool*) are assumed to be command files that can be run to test the software. The resulting command file is a test suite comprised of all the individual tests and can be used to verify software before release. The goal is to have all tests pass before software release.

The following table lists tags (annotations) that can be placed in # comments in command files to provide information for testing, for example:

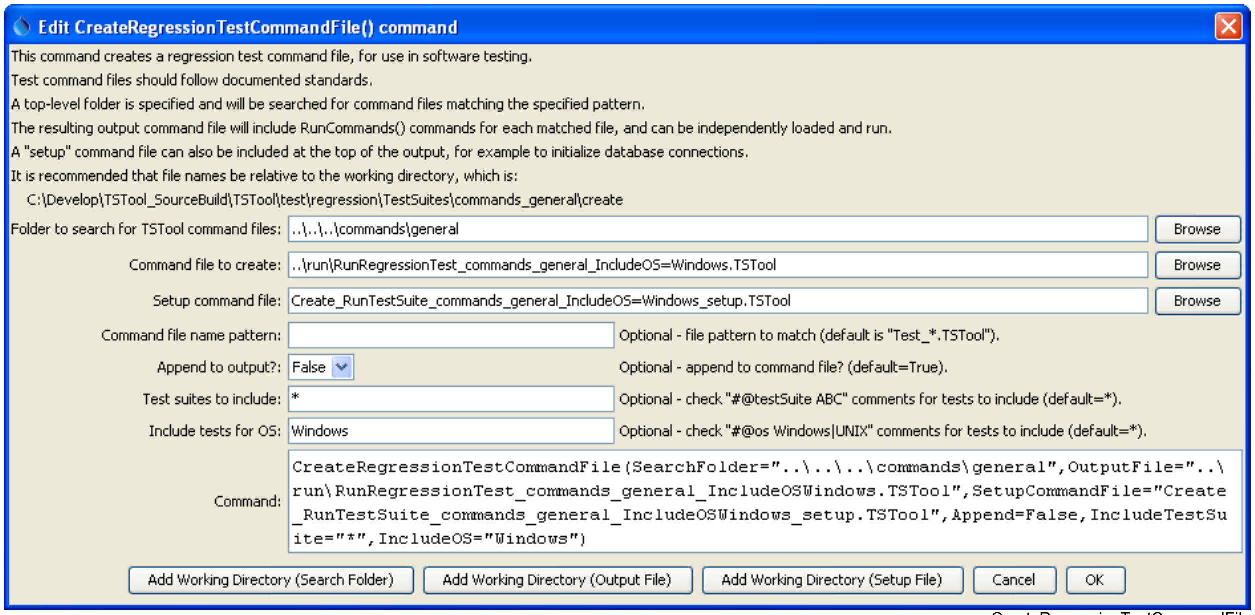
```
#@expectedStatus Failure
```

**Command # Comment Tags**

| Comment Tag | Description |
|---|---|
| `@enabled False` | The `RunCommands()` command will by default run the command file that is provided. However, if the `@enabled False` tag is specified in a comment in the command file, `RunCommands()` will skip the command file. This is useful to disable a test that needs additional work. |
| `@expectedStatus Failure`<br><br>`@expectedStatus Warning` | The `RunCommands()` command `ExpectedStatus` parameter is by default `Success`. However, a different status can be specified if it is expected that a command file will result in `Warning` or `Failure` and still be a successful test. For example, if a command is obsolete and should generate a failure, the expected status can be specified as `Failure` and the test will pass. Another example is to test that the software properly treats a missing file as a failure. |
| `@os Windows`<br>`@os UNIX` | The test is designed to work only on the specified platform and will be included in the test suite only if the `IncludeOS` parameter includes the corresponding operating system (OS) type. This is primarily used to test specific features of the OS and similar but separate test cases should be implemented for both OS types. If the OS type is not specified as a tag in a command file, the test is always included (see also the handling of included test suites). |
| `@readOnly` | Indicates that the command file should not be edited. TSTool will update old command syntax to current syntax when a command file is loaded. However, this tag will cause the software to warn the user when saving the command file, so that they can cancel. |
| `@testSuite ABC` | Indicate that the command file should be considered part of the specified test suite, as specified with the `IncludeTestSuite` |

| Comment Tag | Description |
|---|---|
| | parameter. The test is included in all test collections if the tag is not specified; therefore, for general tests, do not specify a test suite. This tag is useful if a group of tests require special setup, for example connecting to a database. The suite names should be decided upon by the test developer. |

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



This command creates a regression test command file, for use in software testing.
Test command files should follow documented standards.
A top-level folder is specified and will be searched for command files matching the specified pattern.
The resulting output command file will include RunCommands() commands for each matched file, and can be independently loaded and run.
A "setup" command file can also be included at the top of the output, for example to initialize database connections.
It is recommended that file names be relative to the working directory, which is:
C:\Develop\TSTool_SourceBuild\TSTool\test\regression\TestSuites\commands_general\create

| | | |
|---|---|---|
| Folder to search for TSTool command files: | ..\..\..\commands\general | Browse |
| Command file to create: | ..\run\RunRegressionTest_commands_general_IncludeOS=Windows.TSTool | Browse |
| Setup command file: | Create_RunTestSuite_commands_general_IncludeOS=Windows_setup.TSTool | Browse |
| Command file name pattern: | | Optional - file pattern to match (default is "Test_*.TSTool"). |
| Append to output?: | False | Optional - append to command file? (default=True). |
| Test suites to include: | * | Optional - check "#@testSuite ABC" comments for tests to include (default=*). |
| Include tests for OS: | Windows | Optional - check "#@os Windows\|UNIX" comments for tests to include (default=*). |

Command:

```
CreateRegressionTestCommandFile(SearchFolder="..\..\..\commands\general",OutputFile="..\
run\RunRegressionTest_commands_general_IncludeOSWindows.TSTool",SetupCommandFile="Create
_RunTestSuite_commands_general_IncludeOSWindows_setup.TSTool",Append=False,IncludeTestSu
ite="*",IncludeOS="Windows")
```

Add Working Directory (Search Folder)
Add Working Directory (Output File)
Add Working Directory (Setup File)
Cancel
OK

CreateRegressionTestCommandFile

**CreateRegressionTestCommandFile() Command Editor**

The command syntax is as follows:

```
CreateRegressionTestCommandFile(Parameter=Value,…)
```

**Command Parameters**

| Parameter | Description | Default |
|---|---|---|
| `SearchFolder` | The folder to search for regression test command files. All subfolders will also be searched. | None – must be specified. |
| `OutputFile` | The name of the command file to create, enclosed in double quotes if the file contains spaces or other special characters. A path relative to the command file containing this command can be specified. | None – must be specified. |
| `SetupCommandFile` | The name of a TSTool command file that supplies setup commands, and which will be prepended to output. Use such a file to open database connections and set other global settings that apply to the entire test run. | Do not include setup commands. |
| `FilenamePattern` | Pattern for TSTool command files, using wildcards. | `Test_*.TStool` |
| `Append` | Indicate whether to append to the output file (`True`) or overwrite (`False`). This allows multiple directory trees to be searched for tests, where the first command | `True` |

| Parameter | Description | Default |
|---|---|---|
| | typically specifies `False` and additional commands specify `True`. | |
| `IncludeTestSuite` | If *, all tests that match `FilenamePattern` and `IncludeOS` are included. If a test suite is specified, only include tests that have `@testSuite` tag values that match a value in `IncludeTestSuite`. One or more tags can be specified, separated by commas. | * – include all test cases. |
| `IncludeOS` | If *, all tests that match `FilenamePattern` and `IncludeTestSuite` are included. If an OS is specified, only include tests that have `@os` tag values that match a value in `IncludeTestSuite`. This tag is typically specified once or not at all. | * – include all test cases. |

See the **Quality Control** chapter of the TSTool documentation for how to set up a regression test. The following command file illustrates how to create a regression test suite.

```
CreateRegressionTestCommandFile(SearchFolder="..\..\..\commands\general",
  OutputFile="..\run\RunRegressionTest_commands_general.TSTool",Append=False)
```

An example of the output file from running the tests is:

```
# File generated by...
# program:      TSTool 10.20.00 (2013-04-10)
# user:         sam
# date:         Sat Apr 20 13:36:05 MDT 2013
# host:         AMAZON
# directory:    C:\Develop\TSTool_SourceBuild\TSTool\test\regression\TestSuites\commands_general\run
# command line: TSTool
#  -home test/operational/CDSS
#
# Command file regression test report from StartRegressionTestResultsReport() and RunCommands()
#
# Explanation of columns:
#
# Num: count of the tests
# Enabled: blank if test enabled or FALSE if "#@enabled false" in command file
# Run Time: run time in milliseconds
# Test Pass/Fail:
#    The test status below may be PASS or FAIL (or blank if disabled).
#    A test will pass if the command file actual status matches the expected status.
#    Disabled tests are not run and do not count as PASS or FAIL.
#    Search for *FAIL* to find failed tests.
# Commands Expected Status:
#    Default is assumed to be SUCCESS.
#    "#@expectedStatus Warning|Failure" comment in command file overrides default.
# Commands Actual Status:
#    The most severe status (Success|Warning|Failure) for each command file.
#
#    |       |       |Test  |Commands  |Commands   |
#    |       |Run    |Pass/ |Expected  |Actual     |
# Num|Enabled|Time   |Fail  |Status    |Status     |Command File
#----+-------+-------+------+----------+-----------+------------------------------------------------------------------------------------------
    1|       |    141| PASS |SUCCESS   |SUCCESS    |C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ARMA\Test_ARMA_Day.TSTool
    2|       |     31| PASS |SUCCESS   |SUCCESS    |C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ARMA\Test_ARMA_Legacy.TSTool
    3|       |     31| PASS |SUCCESS   |SUCCESS    |C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ARMA\Test_ARMA_Legacy_Ast.TSTool
    4|       |     15| PASS |SUCCESS   |SUCCESS    |C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ARMA\Test_ARMA_Legacy…
…
…
…
   17|FALSE  |      0|      |SUCCESS   |UNKNOWN
|C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\WriteReclamationHDB\Test_WriteReclamationHDB_...
#----+-------+-------+------+----------+-----------+------------------------------------------------------------------------------------------
FAIL count     = 0, 0.000%
PASS count     = 17, 100.000%
Disabled count = 1
#--------------------------------
Total          = 18
```

This page is intentionally blank.