Command Reference: CompareFiles()

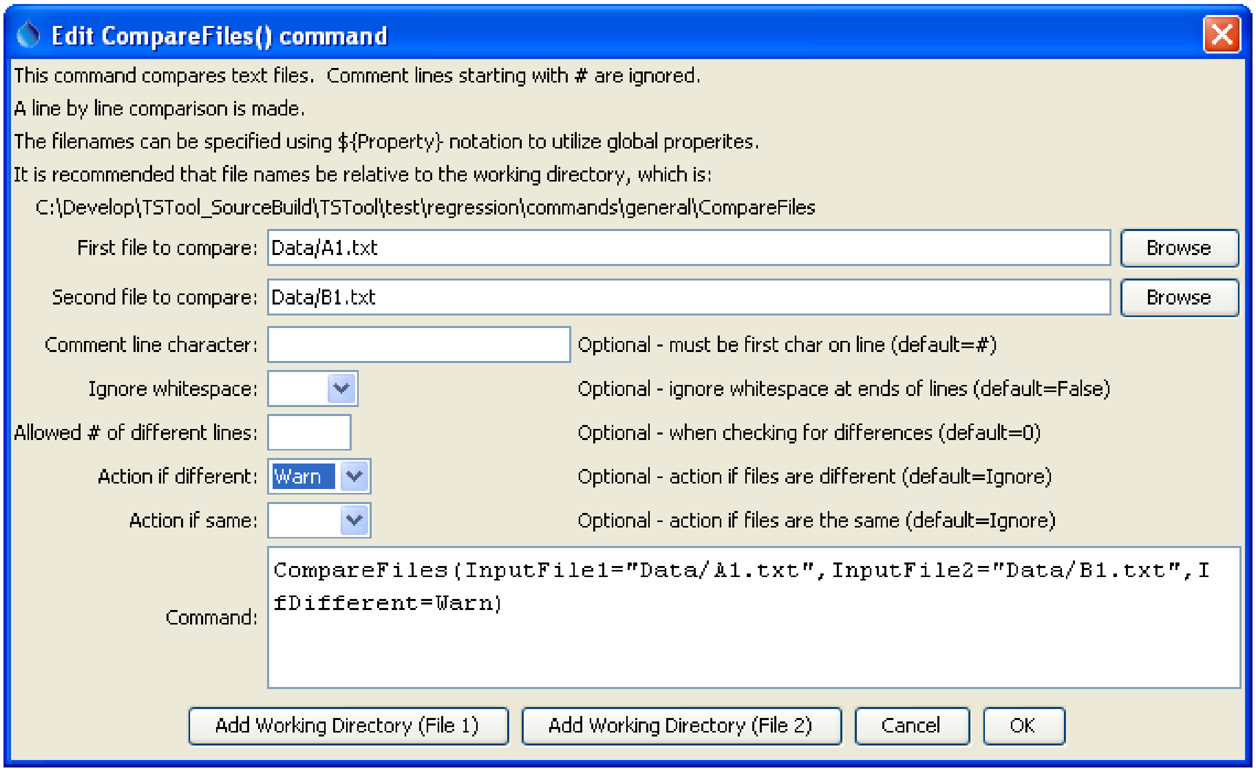
Compare text files to determine whether they are different

Version 10.01.00, 2011-11-15

The `CompareFiles()` command compares text files to determine differences. For example, the command can be used to compare old and new files produced by a software process. This command is suitable for comparing files that are similar, but is not suitable for comparing files that are very different, although it may be enhanced in the future to provide more sophisticated comparison features.

Each line in the file is compared. By default, lines beginning with # are treated as comment lines and are ignored (see `CommentLineChar` to specify the comment indicator). Therefore, only non-comment lines are compared. Comment lines in the middle of the file are simply discarded. Differences and simple statistics are printed to the log file. A warning can be generated if a difference is detected or if no differences are detected (see also the `CompareTimeSeries()and CompareTables()` commands).

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



Edit CompareFiles() command

This command compares text files. Comment lines starting with # are ignored.
A line by line comparison is made.
The filenames can be specified using ${Property} notation to utilize global properites.
It is recommended that file names be relative to the working directory, which is:
C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\CompareFiles

First file to compare: Data/A1.txt

Second file to compare: Data/B1.txt

Comment line character: Optional - must be first char on line (default=#)

Ignore whitespace: Optional - ignore whitespace at ends of lines (default=False)

Allowed # of different lines: Optional - when checking for differences (default=0)

Action if different: Warn — Optional - action if files are different (default=Ignore)

Action if same: Optional - action if files are the same (default=Ignore)

Command:

```
CompareFiles(InputFile1="Data/A1.txt",InputFile2="Data/B1.txt",I
fDifferent=Warn)
```

Browse
Browse
Add Working Directory (File 1)
Add Working Directory (File 2)
Cancel
OK

CompareFiles

CompareFiles() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
`InputFile1`	The name of the first file to read. Enclose the name in double quotes to protect whitespace and special characters. Global properties can be used with the `${Property}` syntax.	None – the file name is required.
`InputFile2`	The name of the second file to read. Enclose the name in double quotes to protect whitespace and special characters. Global properties can be used with the `${Property}` syntax.	None – the file name is required.
`CommentLineChar`	The character(s) that if found at the start of a line indicate comment lines. Comment lines are ignored in the comparison because they typically may include information such as date/time that changes even if the remainder of the file contents is the same.	`#`
`IgnoreWhitespace`	If `True`, then each line is trimmed to remove leading and trailing whitespace characters (spaces and tabs) before doing the comparison. If `False`, then whitespace is retained for the comparison.	`False`
`AllowedDiff`	The number of lines allowed to be different, when checking for differences. This is useful, for example, when a non-comment line contains the date/time when the file was generated.	`0`
`IfDifferent`	Indicates the action to be taken if the files are different: • `Ignore` – do not generate warning • `Warn` – generate a warning message • `Fail` – generate a failure message	Do not generate a warning if the files are different. Differences are printed to the log file.
`IfSame`	Indicates the action to be taken if the files are the same: • `Ignore` – do not generate warning • `Warn` – generate a warning message • `Fail` – generate a failure message	Do not generate a warning if the files are the same.

The following example illustrates how two files can be compared. For example, use similar commands to compare results from two model runs, two database queries, or when testing software:

```
CompareFiles(InputFile1="Data/A1.txt",InputFile2="Data/B1.txt",
  WarnIfDifferent=True)
```