Command Reference: ReadTableFromDelimitedFile()

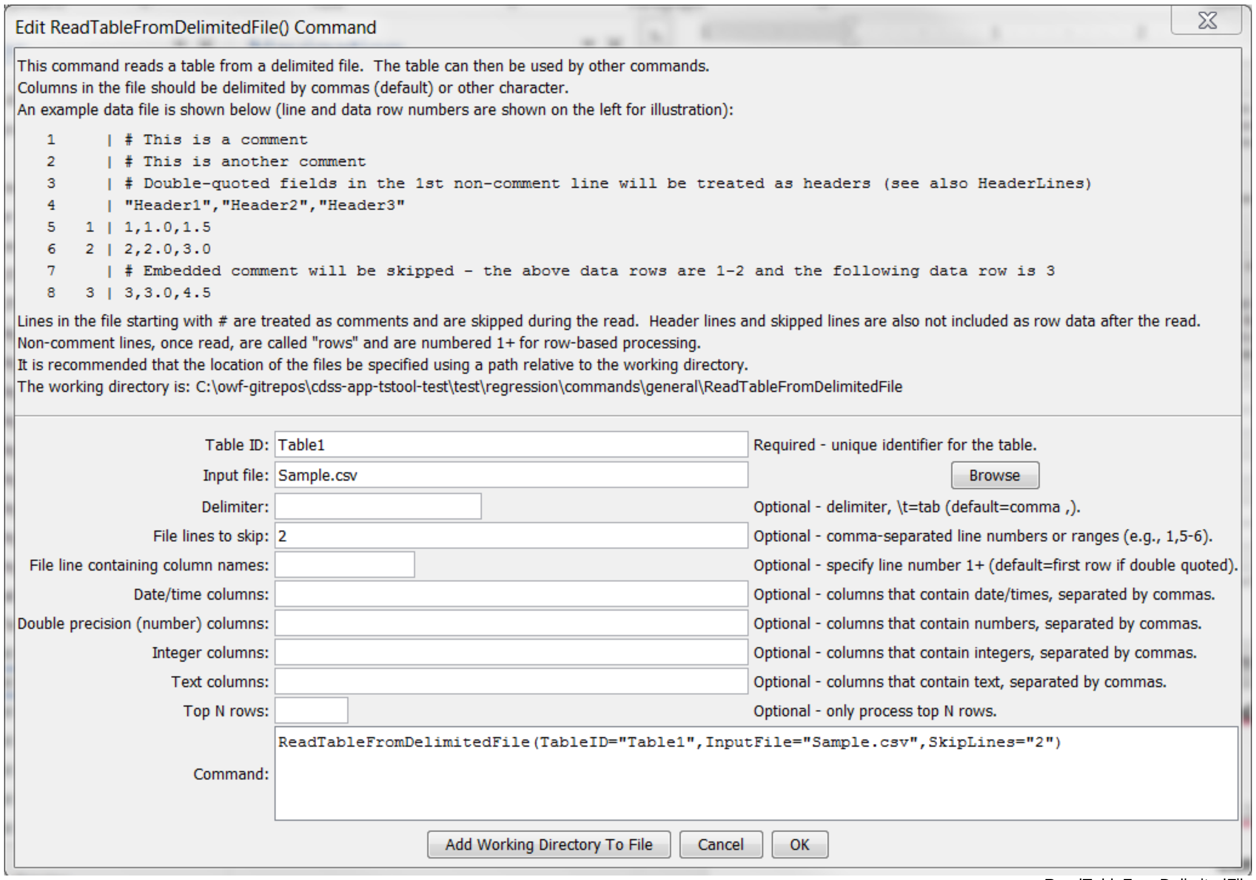
Read a table from a delimited file

Version 11.12.04, 2016-09-17

The `ReadTableFromDelimitedFile()` command reads a table from a comma-delimited file. Tables are used by other commands when performing lookups of information or generating summary information from processing. Table files have the following characteristics:

- Comments indicated by lines starting with `#` are stripped during the read.
- Extraneous lines in the file can be skipped during the read using the `SkipLines` parameter.
- Column headings indicated by "quoted" values in the first non-comment line will be used to assign string names to the columns. If no quoted values are present, columns will not have headings.
- Data in columns are assumed to be of consistent type (i.e., all numerical data or all text), based on rows after the header. The data type for the column will be determined automatically by examining all data.
- Missing values can be indicated by blanks. However, a line ending with the delimiter may cause warnings because blank is not assumed at the end of the line (this is a software limitation that may be addressed in the future) – work around by adding an extra delimiter or ensure that the last column is not blank.
- Strings containing the delimiter should be surrounded by double quotes. Strings that contain quotes are checked. If two sequential quotes are found in input, they are converted to one quote in the table values (see comma-separated-value [CSV] standards: http://en.wikipedia.org/wiki/Comma-separated_values). Subsequent writes of the table will re-introduce the repeated quote to indicate an embedded quote.

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


Edit ReadTableFromDelimitedFile() Command

This command reads a table from a delimited file. The table can then be used by other commands.
Columns in the file should be delimited by commas (default) or other character.
An example data file is shown below (line and data row numbers are shown on the left for illustration):

```
1     | # This is a comment
2     | # This is another comment
3     | # Double-quoted fields in the 1st non-comment line will be treated as headers (see also HeaderLines)
4     | "Header1","Header2","Header3"
5   1 | 1,1.0,1.5
6   2 | 2,2.0,3.0
7     | # Embedded comment will be skipped - the above data rows are 1-2 and the following data row is 3
8   3 | 3,3.0,4.5
```

Lines in the file starting with # are treated as comments and are skipped during the read. Header lines and skipped lines are also not included as row data after the read.
Non-comment lines, once read, are called "rows" and are numbered 1+ for row-based processing.
It is recommended that the location of the files be specified using a path relative to the working directory.
The working directory is: C:\owf-gitrepos\cdss-app-tstool-test\test\regression\commands\general\ReadTableFromDelimitedFile

Table ID:	Table1	Required - unique identifier for the table.
Input file:	Sample.csv	Browse
Delimiter:		Optional - delimiter, \t=tab (default=comma ,).
File lines to skip:	2	Optional - comma-separated line numbers or ranges (e.g., 1,5-6).
File line containing column names:		Optional - specify line number 1+ (default=first row if double quoted).
Date/time columns:		Optional - columns that contain date/times, separated by commas.
Double precision (number) columns:		Optional - columns that contain numbers, separated by commas.
Integer columns:		Optional - columns that contain integers, separated by commas.
Text columns:		Optional - columns that contain text, separated by commas.
Top N rows:		Optional - only process top N rows.
Command:	ReadTableFromDelimitedFile(TableID="Table1",InputFile="Sample.csv",SkipLines="2")	

ReadTableFromDelimitedFile

ReadTableFromDelimitedFile() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
`TableID`	Identifier to assign to the table that is read, which allows the table data to be used with other commands. Can be specified using processor `${Property}`.	None – must be specified.
`InputFile`	The name of the file to read, as an absolute path or relative to the command file location. Can be specified using processor `${Property}`.	None – must be specified.
`Delimiter`	The delimiter character between columns. Specify `\t` to indicate tab. Can be specified using processor `${Property}`.	Comma.
`SkipLines`	Indicates the number of lines in the file to skip, which otherwise would interfere with reading row data. Individual row numbers	No lines are skipped.

	and ranges can be specified, for example: `1,5-6,17`	
`HeaderLines`	Indicate the rows that include header information, which should be used for column names. Currently this should only be one row, although a range may be fully supported in the future.	If the first non-comment line contains quoted field names, they are assumed to be headers. Otherwise, no headers are read.
`DateTime Columns`	List of comma-separated column names for columns that should be treated as containing date/time values. Can be specified using processor `${Property}`.	Date/times default to string (text) columns.
`DoubleColumns`	List of comma-separated column names for columns that should be treated as containing floating point double precision values. Can be specified using processor `${Property}`.	Automatically determine column types from data.
`IntegerColumns`	List of comma-separated column names for columns that should be treated as containing integer values. Can be specified using processor `${Property}`.	Automatically determine column types from data.
`TextColumns`	List of comma-separated column names for columns that should be treated as containing text values. Can be specified using processor `${Property}`.	Automatically determine column types from data.
`Top`	Specify the number of data rows to read, useful when prototyping an analysis process.	Process all rows.

The following example command file illustrates how to read a table from a delimited file:

```
ReadTableFromDelimitedFile(TableID="Table1",
    InputFile="Sample.csv",SkipRows="2")
```

An excerpt from a simple delimited file is:

```
# A comment
some junk to be skipped
"Header1","Header2","Header3"
1,1.0,1.0
2,2.0,1.5
3,3.0,2.0
```