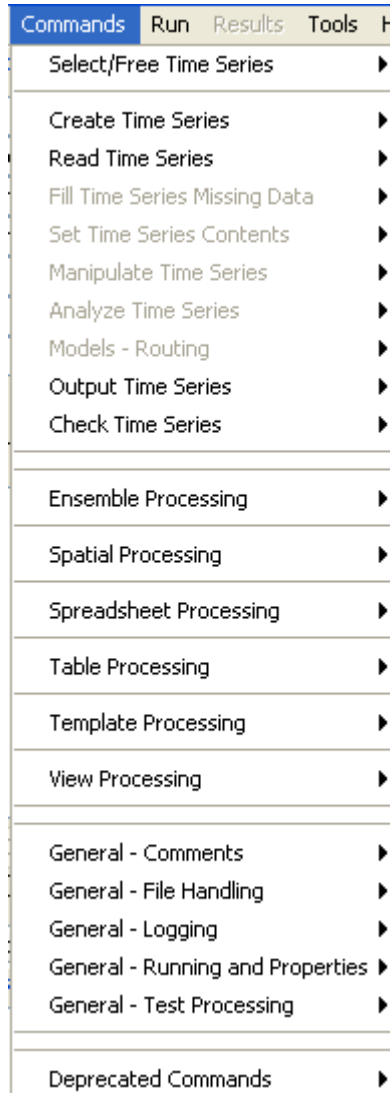# 4   Commands

Version 10.17.00, 2013-02-18

The **Commands** menu provides choices to insert commands in the **Commands** list.

Commands Menu

Menu_Commands

Commands are organized into the following categories:

Select/Free Time Series – select or deselect time series for processing, free time series
Create Time Series – create one or more new time series
Read Time Series – read time series from a file or database
Fill Time Series Missing Data – fill missing data
Set Time Series Contents – set time series data or properties
Manipulate Time Series – manipulate data (e.g., scale a time series' data values)
Analyze Time Series – perform analysis on time series (e.g., determine wet/dry/average pattern)
Models – Routing – lag and attenuate time series

Commands - 1

Output Time Series – write time series results to a file or produce graphical products
Check Time Series – check time series values and statistics against criteria
Ensemble Processing – commands that are specific to ensemble processing
Spatial Processing – commands that process spatial data
Spreadsheet Processing – commands that process spreadsheet files
Table Processing – commands that are specific to table processing
Template Processing – commands that are specific to template processing
View Processing – commands that are specific to view processing (alternate views of results)
General – Comments – insert comments
General – File Handling – commands to manipulate files and perform FTP and web retrieval
General – Logging – commands for logging (e.g., open a log file, set message levels)
General – Running and Properties – commands to control processing and run external programs
General – Test Processing – commands to process tests, to validate software and procedures
Deprecated Commands – commands that are planned for removal

The menus for each category are discussed in the following sections. Selecting a command menu item will display a command editor. The editor dialog and command syntax are described in detail in the **Command Reference** at the end of this documentation. Menus are enabled/disabled depending on the state of the application (e.g., whether time series are available). When *OK* is pressed in the command editor, the command will be inserted before the first selected command. If no commands are selected, the command will be added to the end of the command list. If necessary, right click on the command list and use *Deselect All Commands* – this will ensure that commands are added at the end of the list.

The `TSList` command parameter is available for many commands to specify the list of time series to process. Criteria can be specified to match the alias and TSID strings. The pattern typically can contain wildcards. For example, specify `TSID=A*` to match aliases starting with `A`, or `TSID=A*.*.*.Month` to match TSIDs with a location starting with `A`. An identifier string with periods indicates that the string may be a dot-delimited TSID string, and the TSID parts are compared individually.

For example, `Command(TSList=AllTS,…)` will process all available time series and `Command(TSList=AllMatchingTSID,TSID="A*",…)` will process all time series with an alias that starts with `A`. Refer to the command reference for a description of available parameters for a specific command. The standard values for the `TSList` parameter are as follows:

TSList Parameter Value	Description
AllMatchingTSID	Process all time series that match the `TSID` parameter value.
AllTS	Process all time series.
EnsembleID	Process all time series for the ensemble identifier specified by the `EnsembleID` parameter.
FirstMatchingTSID	Process the first time series (from the start of commands to the previous command) that matches the `TSID` parameter.
LastMatchingTSID	Process the last (previous to the current command) time series that matches the `TSID` parameter.
SelectedTS	Process all time series that have been selected with `SelectTimeSeries()` and `DeselectTimeSeries()` commands.
SpecifiedTSID	Process all time series in the specified list of time series identifiers (for example when used with the `Add()` command the `AddTSID` parameter is used to provide the specified identifiers).

4.1 Select/Free Time Series

The **Commands…Select/Free Time Series** menu inserts commands for selecting, deselecting, and freeing time series.

| DeselectTimeSeries()… <deselect time series for output/processing> |
| SelectTimeSeries()… <select time series for output/processing> |
| Free()… <free time series (will not be available to later commands)> |

Menu_Commands_SelectTimeSeries

Commands...Select/Free Time Series Menu

Time series can be selected for specific actions and are processed by using the `TSList=SelectedTS` command parameters. The `Free()` command frees time series resources, for example when temporary time series are no longer needed or to free up memory for a large command file.

4.2 Create Time Series

The **Commands…Create Time Series** menu inserts commands for creating new time series.

| NewPatternTimeSeries()… <create a time series with repeating data values> |
| NewTimeSeries()… <create and initialize a new time series> |
| ChangeInterval()… <create time series with new interval (timestep)> |
| Copy()… <copy a time series> |
| Delta()… <create new time series as delta between values> |
| Disaggregate()… <disaggregate longer interval to shorter> |
| NewDayTSFromMonthAndDayTS()… <create daily time series from monthly total and daily pattern> |
| NewEndOfMonthTSFromDayTS()… <convert daily data to end of month time series> |
| Normalize()… <Normalize time series to unitless values> |
| RelativeDiff()… <relative difference of time series> |
| ResequenceTimeSeriesData()… <resequence years to create new scenarios> |
| NewStatisticTimeSeries()… <create a time series as repeating statistics from a time series> |
| NewStatisticYearTS()… <create a year time series using a statistic from a time series> |
| RunningStatisticTimeSeries()… <create a statistic time series from a running sample> |

Menu_Commands_CreateTimeSeries

Commands...Create Time Series Menu

These commands create new time series from user-supplied data (see `NewTimeSeries()`) or data from input time series. A time series created from an existing time series is fundamentally different from the original and cannot take its place with the same identifier. For example, the data interval or identifier is different in the new time series. Consequently, the commands force users to provide new TSID and/or alias information to identify the time series.

Commands may create a single or multiple output time series, although the trend is to continue enhancing commands to allow multiple time series to be processed. TSIDs and/or aliases for new time series are used during the discovery phase of command processing to provide identifiers for later commands. This allows command editors to provide time series choices even when the commands have not been run.

4.3 Read Time Series

The **Commands…Read Time Series** menu inserts commands to read time series from a database, file, or web service (internet).

Commands...Read Time Series Menu

Read commands are alphabetized and are shown for enabled input types. The commands that are shown will depend on the input types and datastores that are enabled. Several commands perform supporting functions, such as setting the input period to read. Some data formats allow only a single time series to be read whereas other formats allow multiple time series to be read. Using read commands rather than TSID commands allows more control over the read, such as assigning an alias and converting data units.

The `CreateFromList()` command uses a delimited input file to provide location information and internally creates a list of TSIDs to read.

4.4 Fill Time Series Missing Data

The **Commands…Fill Time Series Data** menu inserts commands for filling missing data in time series.

Commands...Fill Time Series Missing Data Menu

Fill commands will change only values that are missing, whereas set commands (see next section) will change values regardless of whether they are missing or non-missing. These commands can be executed in sequence to apply multiple fill techniques to time series.

Time series may contain missing data due to the following or other reasons:

1. The data collection system is unavailable because of a failure, maintenance cycle, or hardware that is turned off because of seasonal use
2. In a real-time system the most current data have not yet been received
3. Data collection hardware was not in place during a period (e.g., an early period)
4. Measured values are suspected of being in error and are changed to missing
5. Values in a computed time series cannot be computed (e.g., input data are missing)
6. A data source stores only observed values and non-recorded values are assumed to be missing rather than a specific value (e.g., zero)

Observations that are available typically are either measured values or values that have been estimated by the organization that collects and/or maintains the data. Data flags indicating missing data may or may not be available in the original source data (e.g., an 'm' or 'e' character flag often is used to indicate missing and estimated data).

TSTool handles missing data by internally assigning a special numeric value where data are missing. Different input types may have different missing data values but typically -999, a similar extreme value, or NaN (not a number) is used. If the output period is specified using SetOutputPeriod(), then extensions to the available time series period are filled with the missing data value. Data flags are

supported for some input types. TSTool displays and output products indicate missing data as blanks, by showing the missing data value, or a string (e.g., NC), depending on the constraints of the product. For example, an HTML time series highlights missing values and shows flags as a superscript.

Filled time series often are required for use in computer models.   TSTool provides a number of features to fill time series data. The data filling process consists of analyzing available data and using the results to estimate missing data values. The estimation process can be simple or complex, resulting in varying degrees of estimation error and statistical characteristics of the final time series. The data analysis uses data that are available at the time that the fill command is encountered. Consequently if values have been changed since the initial read (e.g., because of layered fill commands), the changed values may impact the analysis. Basic statistical properties of the original data are saved after the initial read to allow use in later fill commands. For example, for monthly time series, the historical monthly averages are computed after the initial read to allow use with a FillHistMonthAverage() command. Fill commands often provide a FillFlag parameter, which allows filled values to be annotated. The flags can then be displayed in reports and graphs.

The overall period that is being filled is controlled by the time series period or analysis period that is specified with fill commands.   TSTool will not automatically extend the period of a filled time series after the time series is initially read. Use the SetInputPeriod() and SetOutputPeriod() commands to control the time series period.

The following table lists the fill techniques that are supported by TSTool.

**TSTool Fill Techniques and Associated Commands**

Technique	Command	Typical Use
Constant	FillConstant()	Use when missing data can be estimated as a constant. For example, if only the early period of a "regulated" (e.g., reservoir) time series is missing, it may be appropriate to set the values to zero.
Monthly total, daily pattern	FillDayTSFrom 2MonthTSAnd1DayTS()	Use to estimate a daily time series by applying the pattern of a related daily time series to monthly totals from the related and current time series. For example, use to estimate daily streamflow from monthly total values.
Fill from time series	FillFromTS()	Use non-missing values from a time series to fill missing values in another time series.
Historical Monthly Average	FillHist MonthAverage()	Use with monthly time series to estimate missing monthly values as the average of historic monthly values. For example, if applied to monthly precipitation data, a missing July value would be set to the average of observed July precipitation values (zero is an observation).
Historical Year Average	FillHist YearAverage()	Use with yearly time series to estimate missing data as the average of annual values.
Interpolation	FillInterpolate()	Use to estimate missing data by interpolating between non-missing values. For example, use to estimate reservoir level changes.
Mixed Station	FillMixedStation()	This command tries various combinations of FillRegression() and FillMove2()

Technique	Command	Typical Use
		parameters with time series at different locations, to use the best combination.
Maintenance of Variance	`FillMOVE1()`	Use to estimate missing data using the Maintenance of Variance Extension (MOVE.1). For example, use to estimate unregulated streamflow from a related gage. **This command is currently not enabled.**
Maintenance of Variance	`FillMOVE2()`	Use to estimate missing data using the Maintenance of Variance Extension (MOVE.2). For example, use to estimate unregulated streamflow from a related gage. This approach has been shown to be slightly better than the MOVE.1 approach.
Historical Pattern Averages	`FillPattern()`	Similar to filling with historic averages with additional complexity of classifying historic months into categories. For example, historic averages for wet, dry, and average periods are computed and used as the historic averages. This command requires that the `SetPatternFile()` control command also be used.
Prorate	`FillProrate()`	Fill a time series by prorating known values with another time series.
Regression	`FillRegression()`	Use to estimate missing data by using ordinary least squares regression. For example, use to estimate streamflow from a related gage.
Repeat	`FillRepeat()`	Use when it can be assumed that the last observed value before a missing period is a good estimate for missing data. For example, use with "forecasted" data where no future value is available for interpolation.
Using diversion comments	`FillUsing DiversionComments()`	This command is only available with the HydroBase input type and uses diversion comments and the "not in use" flag to set additional diversion amounts to zero.

Fill commands can be layered (e.g., use `FillRegression()`, then `FillInterpolate()`, then `FillConstant()`). However, the analysis that occurs for each command may be impacted by earlier fill commands. If necessary, use the `SetFromTS()` command to piece together the results of independent fill commands into a final time series. The *Results...Graph – XY-Scatter* output provides options for selecting different fill techniques and viewing analysis details.

4.5 Set Time Series Data

The **Commands…Set Time Series Contents** menu inserts commands that set time series data and properties. Unlike fill commands, set commands reset values regardless of whether the values were missing in the time series.
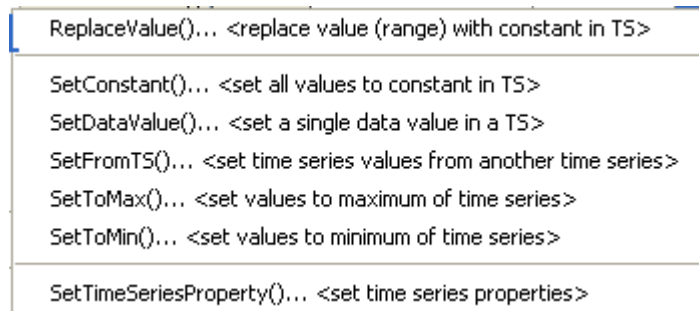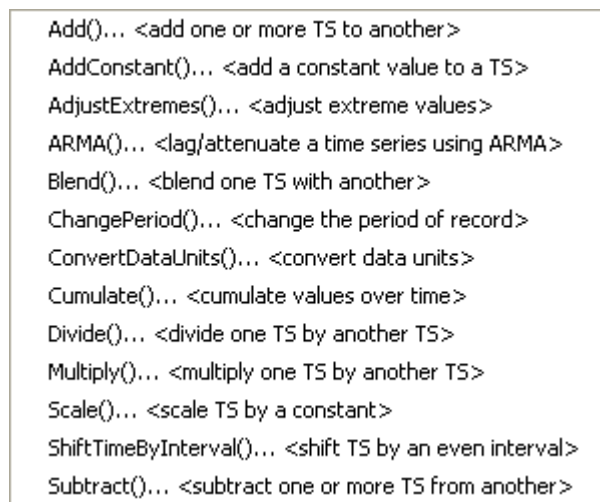

Menu_Commands_SetTimeSeries

Commands...Set Time Series Contents Menu

4.6 Manipulate Time Series

The **Commands…Manipulate Time Series** menus insert commands for manipulating time series.

Menu_Commands_Manipulate

Commands...Manipulate Time Series Menu

Because the fundamental nature of the time series (e.g., data type, interval) is not changed by using these commands, the commands do not result in the creation of a new time series but change the data values in existing time series. Manipulation commands typically add comments to the time series history, which can be viewed with time series properties. If it is necessary to create a separate time series to contain the result of a manipulation, use a `Copy()`, `NewTimeSeries()`, or other command to create a "receiving" time series, and then manipulate this new time series.

4.7 Analyze Time Series

The *Commands…Analyze Time Series* menu inserts commands for analyzing time series, which typically produce a report, result time series, or other output product:

Menu_Commands_AnalyzeTimeSeries

Commands…Analyze Time Series Menu

4.8 Models

The *Commands…Models – Routing* menu inserts commands that perform tasks that are more complex than simple data processes. TSTool does not retrieve or save model states, leaving the handling of states to each model. Currently, the following routing models are available:

Menu_Commands_Models

Commands…Models Menu

Minimal model features are currently available. However, it is envisioned that additional capabilities will be added in the future to facilitate calibration and model evaluation.

4.9 Output Time Series

The *Commands…Output Time Series* menu inserts commands for outputting time series.

Menu_Commands_OutputTimeSeries

Commands...Output Time Series Menu

Commands that set global configuration values (e.g., output period) are listed at the start of the menu. Output commands are shown for input types and datastores that are enabled. The `ProcessTSProduct()` command is used to automate the production of graphs.

Using the output commands allows the results of processing to be saved but does increase processing time. If commands are processed repeatedly during analysis or debugging, the following steps may be taken to increase overall efficiency:

1. Output commands that produce output files are not executed if the **Commands** list is processed with **Run… All Commands (ignore output commands)** or **Run…Selected Commands (ignore output commands)**. Therefore, use this menu choice to ignore the output commands.
2. Only selected commands are processed. Therefore select all but the output commands.
3. Use an `Exit()` control command before output commands to skip the output commands. This command can then be deleted or commented out when not needed.
4. Commands can be converted to comments using the **Commands** menu or the popup menu that is displayed when right-clicking on the **Commands** list. Therefore, output commands can be temporarily converted to comments until output needs to be created.

4.10 Commands to Check Time Series

The **Commands…Check Time Series** menu inserts commands for checking time series.

CheckTimeSeries()... <check time series data values>

CheckTimeSeriesStatistic()... <check time series statistic>

WriteCheckFile()... <write check file>

Menu_Commands_CheckTimeSeries

Commands...Check Time Series Menu

These commands can be used for quality control on raw data and processed time series. A summary of check results can be written to a file to preserve an artifact of data processing.

4.11 Commands for Ensemble Processing

The **Commands…Ensemble Processing** menu provides commands specific to time series ensembles. These commands can only be used with ensembles. However, many commands available in menus described above can be used to process ensembles by processing all of the time series in the ensemble. See the `TSList=EnsembleID` parameter in commands.

CreateEnsembleFromOneTimeSeries()... <convert 1 time series into an ensemble>

CopyEnsemble()... <create a copy of an ensemble>

NewEnsemble()... <create a new ensemble from 0+ time series>

ReadNwsrfsEspTraceEnsemble()... <read 1(+) time series from an NWSRFS ESP trace ensemble file>

InsertTimeSeriesIntoEnsemble()... <insert 1+ time series into an ensemble>

NewStatisticEnsemble()... <create an ensemble of statistic time series>

NewStatisticTimeSeriesFromEnsemble()... <create a time series as a statistic from an ensemble>

WeightTraces()... <weight traces to create a new time series>

WriteNwsrfsEspTraceEnsemble()... <write NWSRFS ESP trace ensemble file>

Menu_Commands_EnsembleProcessing

Commands…Ensemble Processing Menu

4.12 Commands for Spatial Data Processing

The ***Commands…Spatial Processing*** menu provides commands specific to spatial data processing. These commands are under development.

4.13 Commands for Spreadsheet Processing

The ***Commands…Spreadsheet Processing*** menu provides commands specific to spreadsheet processing. These commands are under development.

4.14 Commands for Table Processing

The ***Commands…Table Processing*** menu provides commands specific to table processing. Tables are defined as row/column data (e.g., from delimited files or databases) where comments can be present in the header and data records, the header defines labels for columns, and columns contain consistent data types (i.e., a column has all dates, all integers, all strings, all floating point numbers). Table rows can be related to a time series by using time series properties such as the location part of the time series identifier.

```
NewTable()... <create a new empty table>
CopyTable()... <create a new table as a full/partial copy of another>

ReadTableFromDataStore()... <read a table from a database data store>
ReadTableFromDelimitedFile()... <read a table from a delimited file>
ReadTableFromDBF()... <read a table from a dBASE file>

TimeSeriesToTable()... <copy time series to a table>
TableToTimeSeries()... <create time series from a table>

ManipulateTableString()... <perform simple manipulation on table strings>
TableMath()... <perform simple math on table columns>
TableTimeSeriesMath()... <perform simple math on table columns and time series>

SetTimeSeriesPropertiesFromTable()... <set time series properties from table>
CopyTimeSeriesPropertiesToTable()... <copy time series properties to table>

CompareTables()... <compare two tables (indicate differences)>

WriteTableToDelimitedFile()... <write a table to a delimited file>
WriteTableToHTML()... <write a table to an HTML file>

FreeTable()... <free a table (will not be available to later commands)>
```
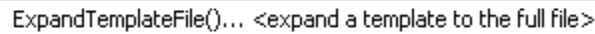
Menu_Commands_TableProcessing

Commands…Table Processing Menu

4.15 Commands for Template Processing

The **Commands…Template Processing** menu provides commands specific to template processing.

Commands…Template Processing Menu

Menu_Commands_TemplateProcessing

Templates are text files that can be expanded to reflect dynamic content. For example, a template command file can be used to repeat a block of commands for many time series. The documentation for the ExpandTemplateFile() command provides examples of how templates can be used.

4.16 Commands for View Processing

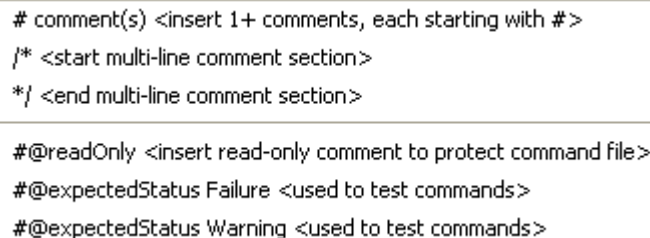The **Commands…View Processing** menu provides commands specific to view processing.

Commands…View Processing Menu

Menu_Commands_ViewProcessing

TSTool lists time series results in the order that time series are created by commands. However, this can lead to very long lists of time series that are difficult to review. The NewTreeView() command allows time series results to be organized in a way that is more appropriate. Other views may be implemented in the future to facilitate viewing results.

4.17 General Commands – Comments

The **Commands…General – Comments** menu provides choices to insert comments.

Commands…General – Comments Menu

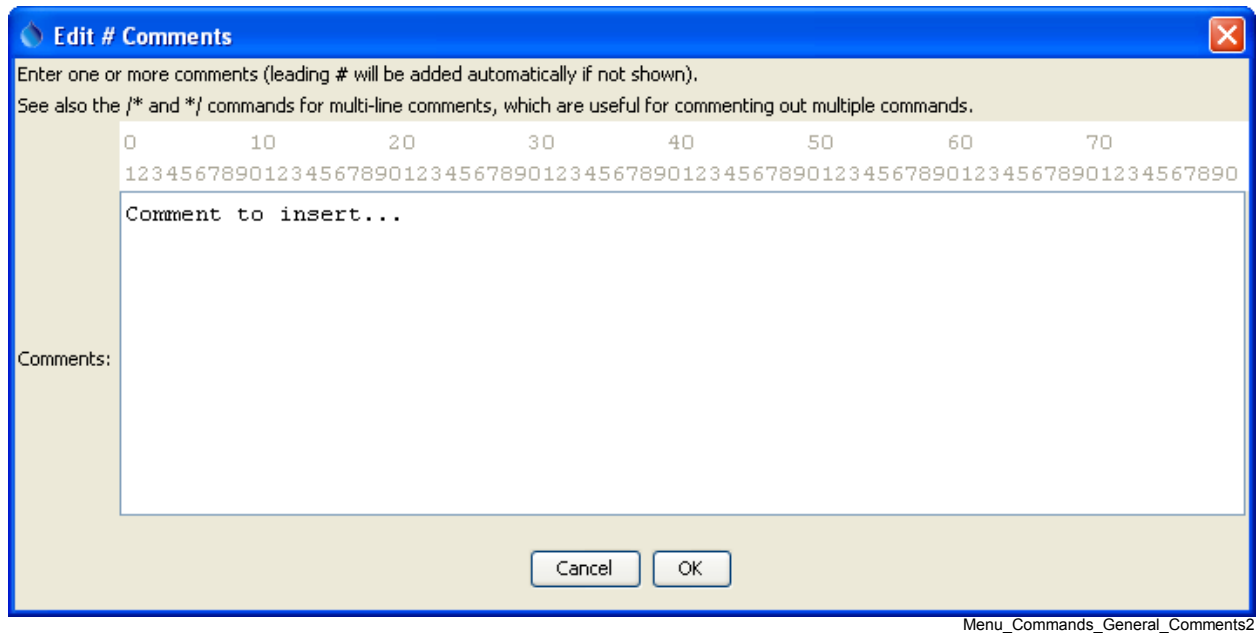Menu_Commands_General_Comments

Comments are treated as special commands. The # character indicates a single-line comment. The /* and */ commands indicate multi-line comments (for example to disable blocks of commands without removing them from the command file). The #@readOnly comment is used to protect a command file – TSTool will warn if an attempt is made to save the file. For example, this special comment is useful to protect old command files that are archived and should not be changed, even if TSTool command syntax changes.

4.17.1 Inserting # Comments

Comments are inserted by selecting a line in the **Commands** list and then selecting **Commands…General – Comments…# Comment(s)**.   The comments will be inserted before the

selected commands. Unlike most other command editors, multiple command lines can be selected. The interface will automatically insert the # character. The following dialog is used to edit comments.

Comment Editor

4.17.2 Start /* */ Comments

Multiple commands can be commented using the following notation:

```
/*
commented lines
commented lines
*/
```

This syntax is consistent with a number of programming languages, including C, C++, and Java, and can be used to quickly disable multiple commands. Use the ***Commands…General – Comments…/* <start comment>*** menu to start a comment above the selected command. Matching start and end comments should be inserted. See also the `exit` control command. Currently there is no way to edit a block of commented code. The notation is meant to be used to comment large blocks of commands, for example during troubleshooting.

4.17.3 End /* */ Comments

Use the ***Commands…General…*/ <end comment>*** menu to end a multi-line comment in commands.

4.18 General Commands – File Handling

The ***Commands…General – File Handling*** menu provides choices to insert commands that process files.

Menu_Commands_General_FileHandling

Commands…General – File Handling Menu

The `RemoveFile()` command is often used in testing. The `FTPGet()` and `WebGet()` commands are used to retrieve files from the internet.

4.19 General Commands – Logging

The ***Commands…General – Logging*** menu provides choices to insert commands used in logging. It is recommended that each command file use a `StartLog()` command as the first command, to create a log file that can facilitate troubleshooting and reviewing work at a later time. Setting the debug and warning level with commands can facilitate troubleshooting specific command logic.

Menu_Commands_General_Logging

Commands…General – Logging Menu

4.20 General Commands – Running Commands and External Software

The ***Commands…General - Running*** menu provides choices to insert commands related to processing commands and external programs.
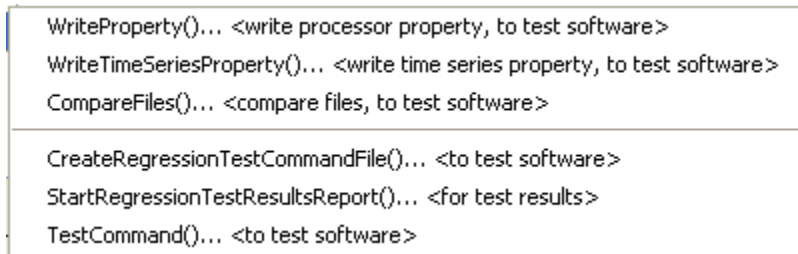

Menu_Commands_General_Run

Commands…General – Running Menu

The command processor uses properties to manage controlling information. These properties can be set with commands to facilitate overall workflow logic, for example to allow configuration information to be defined at the start of a command file, and be used by other commands. TSTool commands will continue to be enhanced to utilize these properties.

The `RunCommands()` command can be used to create a "master" command file that runs other command files. This approach is used to create test suites to validate the TSTool software. Commands also are available to run external programs, Python scripts, and the Army Corps of Engineers DSSUTL software, which provides time series processing capabilities.

4.21 General Commands – Test Processing

The **Commands…General – Test Processing** menu provides choices to insert commands related to software and process testing. A test case can be a simple test (e.g., test of a single command with a specific combination of parameters) or a more complex test (e.g., a test of a command file used to process a data set file). The `CreateRegressionTestCommandFile()` can be used to search a folder and sub-folders for command files matching a pattern (e.g., *Test_*.TSTool*). This will create a master command file that includes `RunCommands()` commands. These commands are used by software developers to create test suites to verify TSTool software functionality and can also be used by software users to verify that a process is certified and gives expected results. Comparing the results from a specific software version with expected results is useful for diagnosing errors.


```
WriteProperty()... <write processor property, to test software>
WriteTimeSeriesProperty()... <write time series property, to test software>
CompareFiles()... <compare files, to test software>

CreateRegressionTestCommandFile()... <to test software>
StartRegressionTestResultsReport()... <for test results>
TestCommand()... <to test software>
```

Menu_Commands_General_Testing

Commands…General – Test Processing Menu

The following is an example command file to run the `CreateRegressionTestCommandFile()` command:

```
#
# Create the regression test runner for the
# TSTool/test/regression/TestSuites/commands_general files.
#
# Only command files that match Test_*.TSTool are included in the output.
# Don't append the generated commands, in order to force the old file to be
# overwritten.
#
CreateRegressionTestCommandFile(SearchFolder="..\..\..\commands\general",
OutputFile="..\run\RunRegressionTest_commands_general.TSTool",Append=False)
```

The following command file is generated from the above and can be run to execute the individual tests. Typically each test uses the `CompareTimeSeries()` or `CompareFiles()` command to generate a warning if results are not as expected.

```
StartRegressionTestResultsReport(
OutputFile="RunRegressionTest_commands_general.TSTool.out.txt")
RunCommands(InputFile="..\..\..\commands\general\add\Test_Add_1.TSTool")
RunCommands(InputFile="..\..\..\commands\general\add\Test_Add_Ensemble_1.TSTool")
/RunCommands(InputFile="..\..\..\commands\general\ChangeInterval\
Test_ChangeInterval_IrregINST_To_3HourINST.TSTool")
… etc. …
```

4.22 Deprecated Commands

The ***Commands…Deprecated Commands*** menu provides commands that are slated for removal at some point in the future. These commands should be avoided and newer alternatives should be used. Refer to the command documentation for information in migrating to other commands.

Menu_Commands_Deprecated

Commands…Deprecated Commands Menu