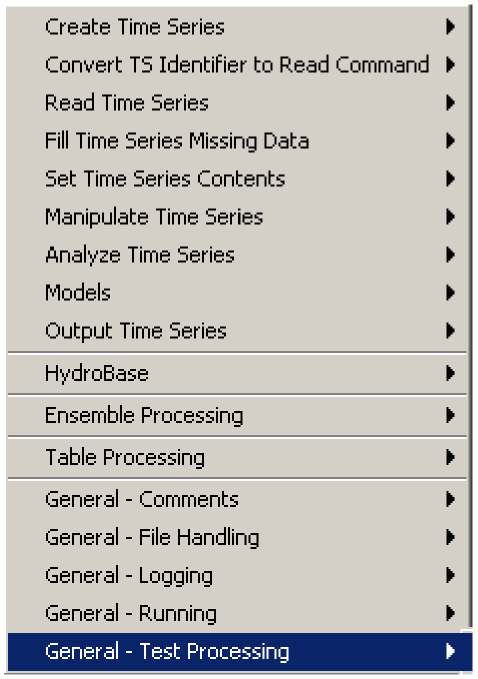
4 Commands

Version 09.06.01, 2010-02-22

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



Menu_Commands

Commands Menu

Commands are organized into the following categories:

1. ***Create Time Series -*** create one or more new time series in memory
2. ***Convert TS Identifier to Read Command*** – convert a time series identifier in the ***Commands*** list area to a read command
3. ***Read Time Series*** – read time series from a file or database
4. ***Fill Time Series Missing Data -*** fill missing data
5. ***Set Time Series Contents*** – set time series data or properties
6. ***Manipulate Time Series -*** manipulate data by transforming the contents of the time series (e.g., scale a time series' data values)
7. ***Analyze Time Series*** – perform analysis on time series, without modifying the time series
8. ***Models*** – advanced or specific models that operate on time series data
9. ***Output Time Series*** – write time series results to a file or produce graphical products
10. ***HydroBase*** – commands specific to the HydroBase database
11. ***Ensemble Processing*** – commands that are specific to ensemble processing
12. ***Table Processing*** – commands that are specific to table processing
13. ***General – Comments*** – insert comments
14. ***General – File Handling*** – commands to manipulate files (e.g., remove)
15. ***General – Logging*** – commands for logging (e.g., open a log file, set message levels)
16. ***General – Running*** – commands to control running external programs
17. ***General – Test Processing*** – commands to process tests, to validate software and procedures

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).

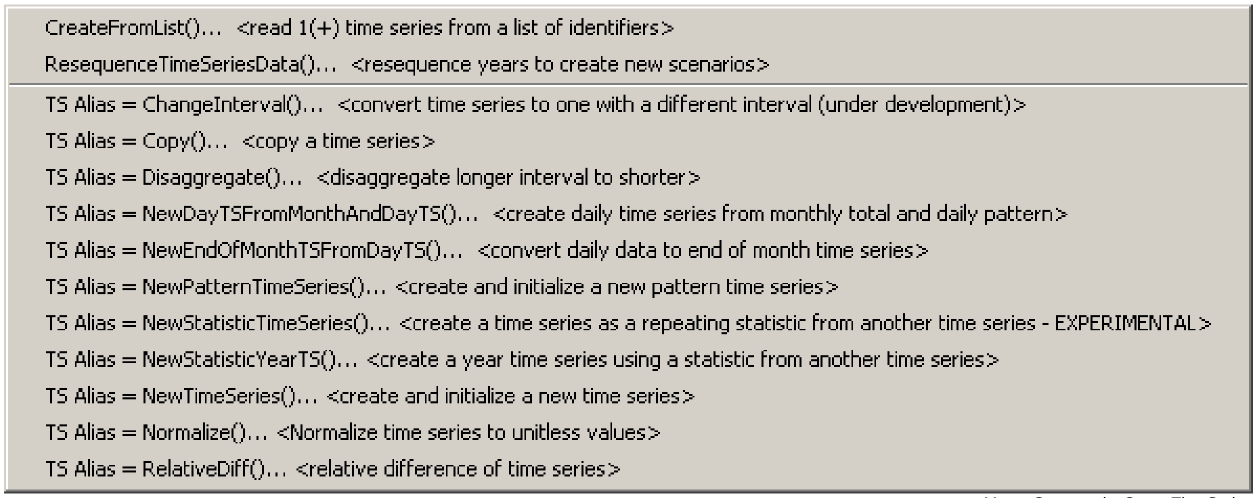
Many commands allow a list of 1+ time series to be processed. The `TSList` command parameter is available for many commands to indicate how the list of time series to be processed is determined. For example, `TSList=AllTS` will process all available time series. Refer to the command reference for a description of available parameters for a 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 (typically can contain wildcards). For example, specify `TSID=A*` or `TSID=A*.*.*.Month`
`AllTS`	Process all time series.
`EnsembleID`	Process all time series for the ensemble identifier.
`LastMatchingTSID`	Process the last (previous to the current command) TSID that matches the `TSID` parameter.
`SelectedTS`	Process all time series that have been selected with `SelectTimeSeries()` 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).

The `TSList` parameter provides a flexible way to specify time series to be processed, and allows new capabilities to be added for commands that use this parameter.

4.1 Create Time Series

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


CreateFromList()... <read 1(+) time series from a list of identifiers>
ResequenceTimeSeriesData()... <resequence years to create new scenarios>
TS Alias = ChangeInterval()... <convert time series to one with a different interval (under development)>
TS Alias = Copy()... <copy a time series>
TS Alias = Disaggregate()... <disaggregate longer interval to shorter>
TS Alias = NewDayTSFromMonthAndDayTS()... <create daily time series from monthly total and daily pattern>
TS Alias = NewEndOfMonthTSFromDayTS()... <convert daily data to end of month time series>
TS Alias = NewPatternTimeSeries()... <create and initialize a new pattern time series>
TS Alias = NewStatisticTimeSeries()... <create a time series as a repeating statistic from another time series - EXPERIMENTAL>
TS Alias = NewStatisticYearTS()... <create a year time series using a statistic from another time series>
TS Alias = NewTimeSeries()... <create and initialize a new time series>
TS Alias = Normalize()... <Normalize time series to unitless values>
TS Alias = RelativeDiff()... <relative difference of time series>

Menu_Commands_CreateTimeSeries

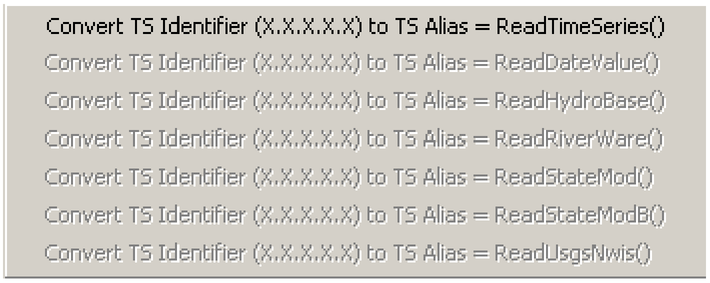
Commands...Create Time Series Menu

These commands create new time series from external data (see `CreateFromList()`), by using user-supplied data (see `NewTimeSeries()`), or by operating on existing time series. Time series created from existing time series are 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.

One important difference between the commands is whether one or multiple time series are created as the result of a command. For example, the `TS Alias =` commands create a single time series, which can be referenced using the time series alias (`Alias`). However, other commands may create more than one time series, and the identifiers for the time series are determined during the discovery phase of command processing (e.g., when time series identifiers are read from a file, but the data are not).

4.2 Converting Time Series Identifier to Read Command

The ***Commands…Convert Time Series Identifier to Read Command*** menu converts a selected time series identifier in the ***Commands*** list to a read command.


Convert TS Identifier (X.X.X.X.X) to TS Alias = ReadTimeSeries()
Convert TS Identifier (X.X.X.X.X) to TS Alias = ReadDateValue()
Convert TS Identifier (X.X.X.X.X) to TS Alias = ReadHydroBase()
Convert TS Identifier (X.X.X.X.X) to TS Alias = ReadRiverWare()
Convert TS Identifier (X.X.X.X.X) to TS Alias = ReadStateMod()
Convert TS Identifier (X.X.X.X.X) to TS Alias = ReadStateModB()
Convert TS Identifier (X.X.X.X.X) to TS Alias = ReadUsgsNwis()

Menu_Commands_ConvertTSID

Commands...Convert Time Series Identifier to Read Command Menu

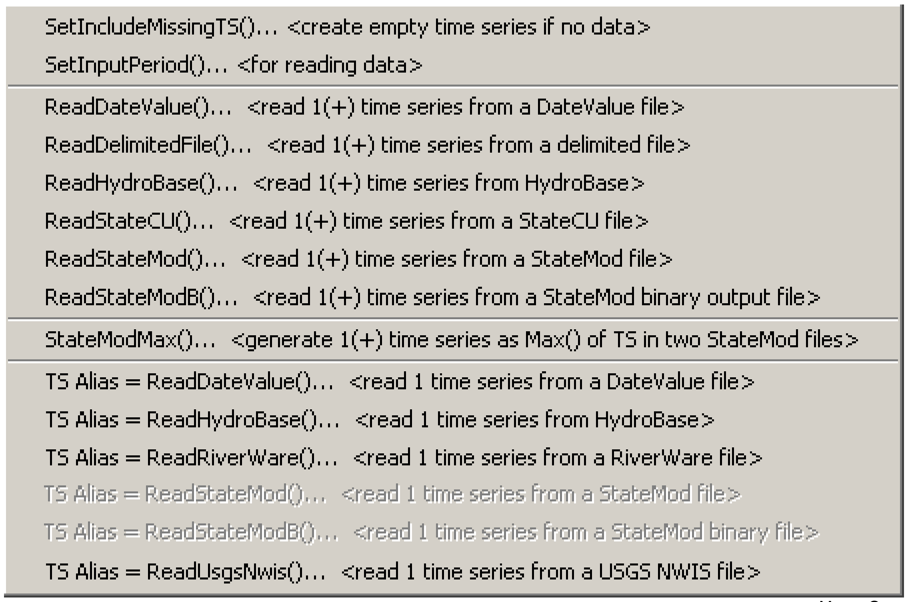
Currently, the only enabled feature is to convert a time series identifier to a `TS Alias = ReadTimeSeries()` command. This is used to assign an alias to the time series, simplifying its use in other commands. The input type from the time series identifier is interpreted by TSTool to determine how to read the time series. Unlike other commands menus, this menu does not insert a new command. It converts a single selected time series identifier.

In the future, it is envisioned that this menu will be enhanced to enable conversion of a time series identifier to a specific read command – this will allow the features of each read command to be used.

See also the specific read commands discussed in the next section.

4.3 Read Time Series

The ***Commands…Read Time Series*** menu inserts commands to read time series from a database or file.



SetIncludeMissingTS()... <create empty time series if no data>
SetInputPeriod()... <for reading data>
ReadDateValue()... <read 1(+) time series from a DateValue file>
ReadDelimitedFile()... <read 1(+) time series from a delimited file>
ReadHydroBase()... <read 1(+) time series from HydroBase>
ReadStateCU()... <read 1(+) time series from a StateCU file>
ReadStateMod()... <read 1(+) time series from a StateMod file>
ReadStateModB()... <read 1(+) time series from a StateMod binary output file>
StateModMax()... <generate 1(+) time series as Max() of TS in two StateMod files>
TS Alias = ReadDateValue()... <read 1 time series from a DateValue file>
TS Alias = ReadHydroBase()... <read 1 time series from HydroBase>
TS Alias = ReadRiverWare()... <read 1 time series from a RiverWare file>
TS Alias = ReadStateMod()... <read 1 time series from a StateMod file>
TS Alias = ReadStateModB()... <read 1 time series from a StateMod binary file>
TS Alias = ReadUsgsNwis()... <read 1 time series from a USGS NWIS file>

Menu_Commands_ReadTimeSeries

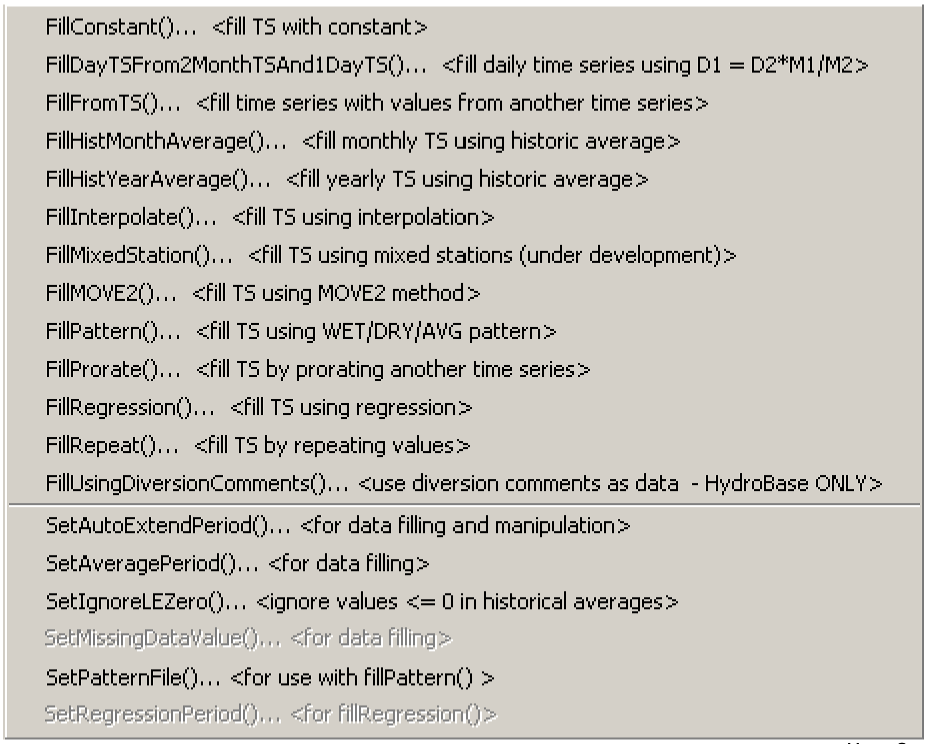
Commands...Read Time Series Menu

Read commands are grouped into commands that process one or more time series in a file (`Read*` choices in menu) and commands that read a specific time series (`TS Alias =` commands). The `TS Alias =` commands assign an alias to the time series, which can then be referenced by other commands.

It may be useful to read many time series and then use the `SelectTimeSeries()` and `DeselectTimeSeries()` commands to select a subset of the results (see output commands).

4.4 Fill Time Series Data

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



FillConstant()... <fill TS with constant>
FillDayTSFrom2MonthTSAnd1DayTS()... <fill daily time series using D1 = D2*M1/M2>
FillFromTS()... <fill time series with values from another time series>
FillHistMonthAverage()... <fill monthly TS using historic average>
FillHistYearAverage()... <fill yearly TS using historic average>
FillInterpolate()... <fill TS using interpolation>
FillMixedStation()... <fill TS using mixed stations (under development)>
FillMOVE2()... <fill TS using MOVE2 method>
FillPattern()... <fill TS using WET/DRY/AVG pattern>
FillProrate()... <fill TS by prorating another time series>
FillRegression()... <fill TS using regression>
FillRepeat()... <fill TS by repeating values>
FillUsingDiversionComments()... <use diversion comments as data - HydroBase ONLY>
SetAutoExtendPeriod()... <for data filling and manipulation>
SetAveragePeriod()... <for data filling>
SetIgnoreLEZero()... <ignore values <= 0 in historical averages>
SetMissingDataValue()... <for data filling>
SetPatternFile()... <for use with fillPattern() >
SetRegressionPeriod()... <for fillRegression()>

Menu_Commands_FillTimeSeries

Commands...Fill Time Series Missing Data Menu

Fill commands will only change 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. Many commands accept the `*` wildcard for the time series identifier, allowing all time series to be filled.

Time series may contain missing data due to the following 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 are typically 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 is often 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 by blanks, showing the missing data value, or a string (e.g., `NC`).

Filled time series are often 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 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.

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.
Historic Monthly Average	`FillHistMonthAverage()`	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).

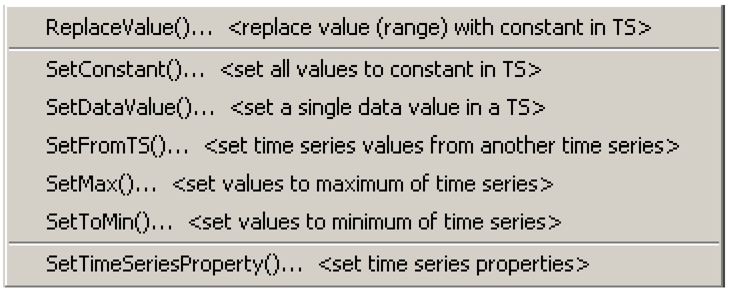
TSTool Fill Techniques and Associated Commands (continued)

Technique	Command	Typical Use
Historic Year Average	`FillHistYearAverage()`	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()` 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.
Historic 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	`FillUsingDiversionComments()`	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.

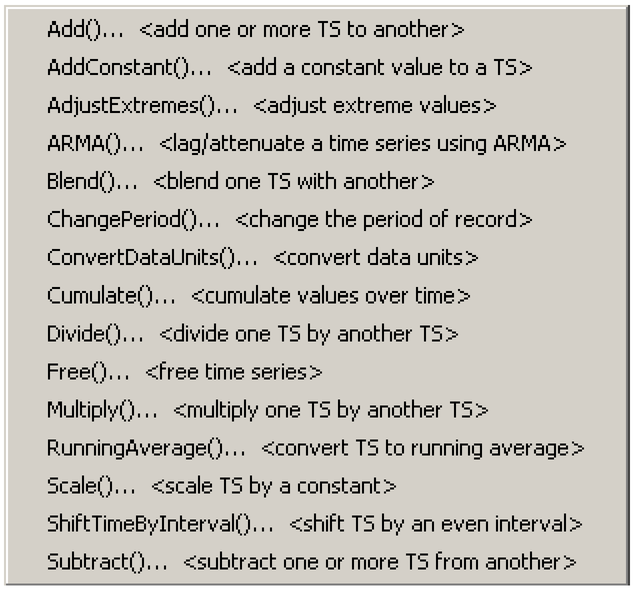

ReplaceValue()... <replace value (range) with constant in TS>
SetConstant()... <set all values to constant in TS>
SetDataValue()... <set a single data value in a TS>
SetFromTS()... <set time series values from another time series>
SetMax()... <set values to maximum of time series>
SetToMin()... <set values to minimum of time series>
SetTimeSeriesProperty()... <set time series properties>

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.


Add()... <add one or more TS to another>
AddConstant()... <add a constant value to a TS>
AdjustExtremes()... <adjust extreme values>
ARMA()... <lag/attenuate a time series using ARMA>
Blend()... <blend one TS with another>
ChangePeriod()... <change the period of record>
ConvertDataUnits()... <convert data units>
Cumulate()... <cumulate values over time>
Divide()... <divide one TS by another TS>
Free()... <free time series>
Multiply()... <multiply one TS by another TS>
RunningAverage()... <convert TS to running average>
Scale()... <scale TS by a constant>
ShiftTimeByInterval()... <shift TS by an even interval>
Subtract()... <subtract one or more TS from another>

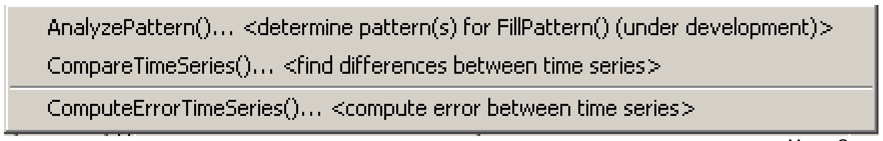
Menu_Commands_Manipulate

Commands...Manipulate Time Series Menu

Because the fundamental nature of the time series (e.g., data type, interval) is not changed, these commands do not result in the creation of a new time series. Manipulation commands typically add comments to the time series history, which can be viewed with time series properties.

4.7 Analyze Time Series

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

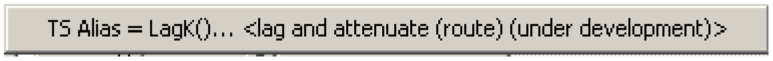

AnalyzePattern()… <determine pattern(s) for FillPattern() (under development)>
CompareTimeSeries()… <find differences between time series>
ComputeErrorTimeSeries()… <compute error between time series>

Menu_Commands_AnalyzeTimeSeries

Commands…Analyze Time Series Menu

4.8 Models

The ***Commands…Models*** 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.


TS Alias = LagK()… <lag and attenuate (route) (under development)>

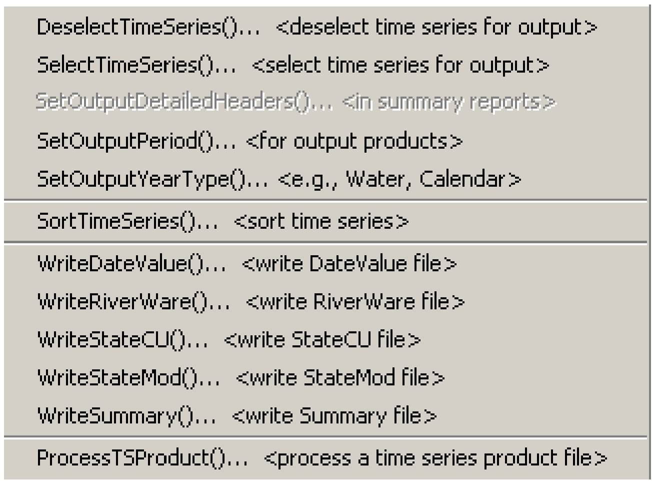
Menu_Commands_Models

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.


DeselectTimeSeries()... <deselect time series for output>
SelectTimeSeries()... <select time series for output>
SetOutputDetailedHeaders()... <in summary reports>
SetOutputPeriod()... <for output products>
SetOutputYearType()... <e.g., Water, Calendar>
SortTimeSeries()... <sort time series>
WriteDateValue()... <write DateValue file>
WriteRiverWare()... <write RiverWare file>
WriteStateCU()... <write StateCU file>
WriteStateMod()... <write StateMod file>
WriteSummary()... <write Summary file>
ProcessTSProduct()... <process a time series product file>

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 and commands that operate on time series are listed last.

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

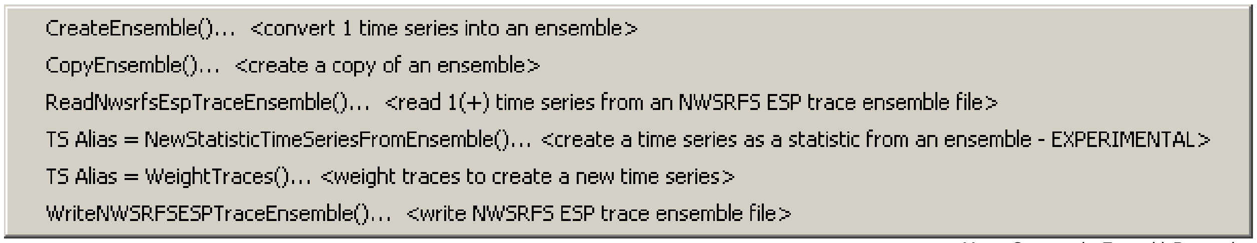
1. Output commands that produce output files are not executed if the ***Commands*** list is processed with ***Run… All Commands (ignore output commands)*** or ***Run…Selected Commands (ignore output commands)***. Therefore, use this menu choice to ignore the output commands.
2. Only selected commands 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 for Specific Input Types

Depending on the input types that are enabled, additional command menus may be enabled. For example, if the HydroBase input type is enabled, a HydroBase menu will be enabled, and will list commands specific to HydroBase. In particular, commands like `OpenHydroBase()` are provided to make database connections while processing commands.

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.

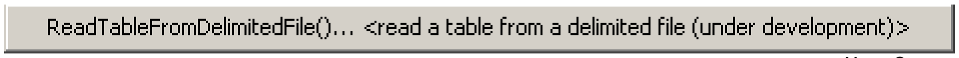

CreateEnsemble()... <convert 1 time series into an ensemble>
CopyEnsemble()... <create a copy of an ensemble>
ReadNwsrfsEspTraceEnsemble()... <read 1(+) time series from an NWSRFS ESP trace ensemble file>
TS Alias = NewStatisticTimeSeriesFromEnsemble()... <create a time series as a statistic from an ensemble - EXPERIMENTAL>
TS Alias = 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 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, or all data values). Table commands are being phased in to support processing such as creating time series lists, rating curve conversions and other transformations, summarizing statistics, and report generation.

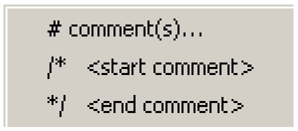

ReadTableFromDelimitedFile()... <read a table from a delimited file (under development)>

Menu_Commands_TableProcessing

Commands…Table Processing Menu

4.13 General Commands – Comments

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

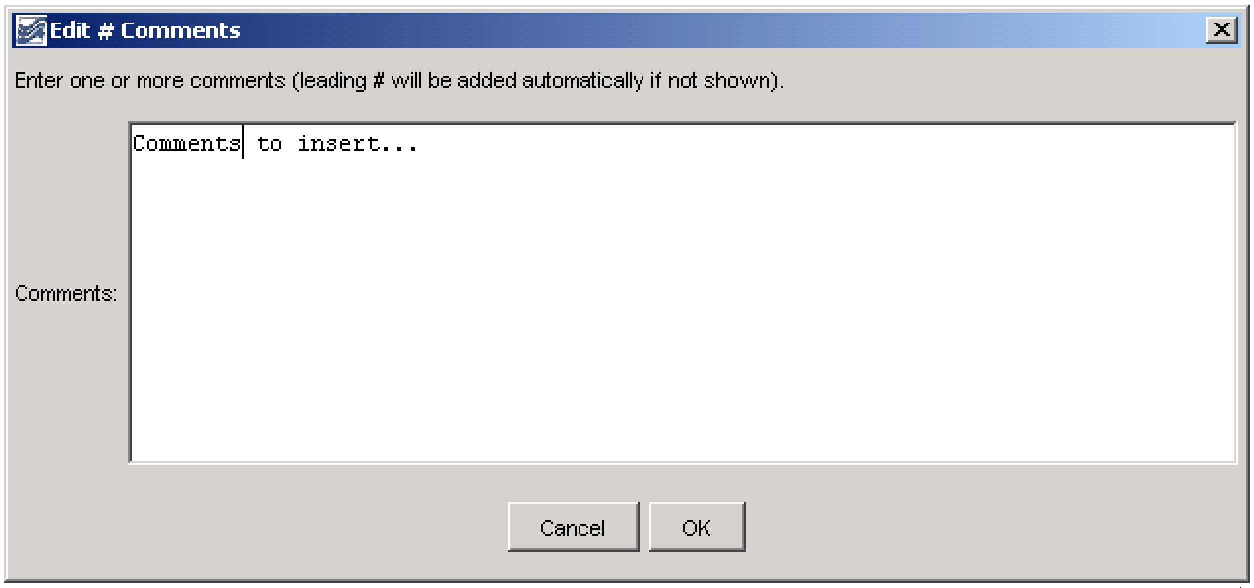


Menu_Commands_General_Comments

Commands…General – Comments Menu

4.13.1 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.



Enter one or more comments (leading # will be added automatically if not shown).

Comments: Comments to insert...

comments

Comment Editor

4.13.2 Start /* */ Comments

Multiple commands can be commented using the following notation:

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

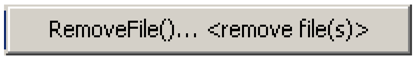
This syntax is consistent with a number of programming languages, including C, C++, and Java, and can be used to quickly disable multiple commands. Use the ***Commands…General – Comments…/* <start comment>*** menu to start a comment above the selected command. Matching start and end comments should be 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.

4.13.3 End /* */ Comments

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

4.14 General Commands – File Handling

The ***Commands…General – File Handling*** menu provides choices to insert commands that process files. The `RemoveFile()` command is mainly used in testing but additional capability may be added.

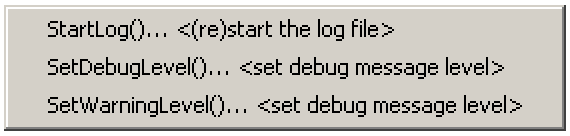


Menu_Commands_General_FileHandling

Commands…General – File Handling Menu

4.15 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.

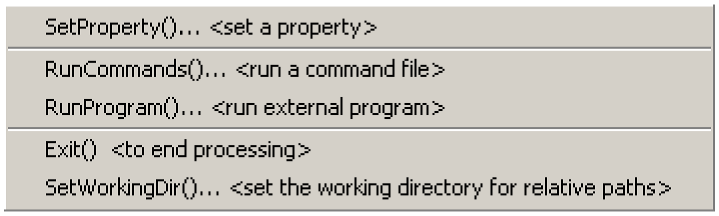

StartLog()… <(re)start the log file>
SetDebugLevel()… <set debug message level>
SetWarningLevel()… <set debug message level>

Menu_Commands_General_Logging

Commands…General – Logging Menu

4.16 General Commands – Running Commands and External Software

The ***Commands…General - Running*** menu provides choices to insert commands related to running the commands and external programs. A master command file can also be used to run other command files (this approach is used in software testing as described in the next section).

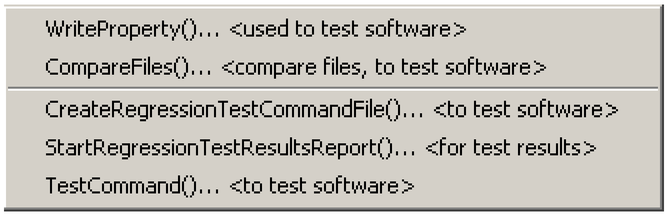

SetProperty()… <set a property>
RunCommands()… <run a command file>
RunProgram()… <run external program>
Exit() <to end processing>
SetWorkingDir()… <set the working directory for relative paths>

Menu_Commands_General_Run

Commands…General – Running Menu

4.17 General Commands – Test Processing

The ***Commands…General – Test Processing*** menu provides choices to insert commands related to 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 directory structure 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()… <used 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. …
```

This page is intentionally blank.