TSTool – Time Series Tool –

Command Reference

Version 10.21.00, 2013-04-21

This page is intentionally blank.

This document is formatted for double-sided printing.

TSTool	1
– Time Series Tool –	1
Blank Page	16
TSTool Syntax Guide	17
Commands – Basic Syntax	17
Commands – Referring to Parameters	17
Commands – Comments	18
Commands – Time Series Identifiers	18
Processor – Properties	18
Time Series – Properties	19
Time Series – Data Flags	19
Date/Time	20
Regular Expression – Notation	21
Template – Syntax	21
Configuration File – TSTool Configuration File	21
Configuration File – Datastore Properties	21
Configuration File – Time Series Product Files	22

Command Glossary	23
Command Reference: #	33
Command Reference: /*	35
Command Reference: */	37
Command Reference: Time Series Identifier (TSID)	39
Command Reference: Add()	41
Command Reference: AddConstant()	43
Command Reference: AdjustExtremes()	45
Command Reference: AnalyzePattern()	47
Command Reference: AnalyzeNetworkPointFlow()	51
Command Reference: AppendFile()	63
Command Reference: AppendTable()	65
Command Reference: ARMA()	67
Command Reference: Blend()	73
Command Reference: CalculateTimeSeriesStatistic()	75
Command Reference: ChangeInterval()	81
Irregular Time Series to Regular Time Series	81

Regular Time Series to Regular Time Series	83
ACCM (Accumulation) to ACCM (Accumulation)	83
ACCM (Accumulation) to INST (Instantaneous)	84
ACCM (Accumulation) to MEAN	84
INST (Instantaneous) to INST (Instantaneous)	85
INST (Instantaneous) to ACCM (Accumulation)	86
INST (Instantaneous) to MEAN	86
MEAN to MEAN	88
MEAN to ACCM (Accumulation)	88
MEAN to INST (Instantaneous)	88
Command Reference: ChangePeriod()	97
60_Command_CheckTimeSeries.pdf	99
Command Reference: CheckTimeSeriesStatistic()	103
60_Command_CompareFiles.pdf	107
60_Command_CompareTables.pdf	109
Command Reference: CompareTimeSeries()	111
Command Reference: ComputeErrorTimeSeries()	115

Command Reference:	ConvertDataUnits()	117
Command Reference:	Copy()	119
Command Reference:	CopyEnsemble()	121
Command Reference:	CopyTable()	123
Command Reference:	CreateEnsembleFromOneTimeSeries()	125
Command Reference:	CreateFromList()	129
Command Reference:	CreateRegressionTestCommandFile()	133
Command Reference:	Cumulate()	137
Command Reference:	Delta()	141
Command Reference:	DeselectTimeSeries()	145
Command Reference:	Disaggregate()	147
Command Reference:	Divide()	151
Command Reference:	Exit()	153
Command Reference:	ExpandTemplateFile()	155
Example Using Simple	Variable Assignment	157
Example of Passing Ti	me Series Processor Properties to Templates	157
Example of Protecting	TSTool Properties in Template with a Literal FreeMarker String	157

Example of Using a Comment in the Template, which is Omitted from Expanded Output	158
Example Using Variable Assignment and Loop Using List	158
Example Using a One-Column Table for a List for Looping	159
Example Using a Multiple-Column Table to Loop Through Two Lists	160
Example of Expanding a Template to a Processor Property	161
Example of Using ExpandTemplateFile() in a Loop to Expand Multiple Files	161
Command Reference: FillConstant()	167
Command Reference: FillDayTSFrom2MonthTSAnd1DayTS()	169
60_Command_FillFromTS.pdf	173
Command Reference: FillHistMonthAverage()	175
Command Reference: FillHistYearAverage()	177
Command Reference: FillInterpolate()	179
Command Reference: FillMixedStation()	181
Implementation in Colorado's Decision Support Systems	182
Command Reference: fillMOVE1()	187
Command Reference: FillMOVE2()	189
Command Reference: FillPattern()	193

60_Command_FillPrincipalComponentAnalysis.pdf	195
Command Reference: FillProrate()	197
Command Reference: FillRegression()	201
Command Reference: FillRepeat()	211
Command Reference: FillUsingDiversionComments()	213
Diversion Comment Not Used Flag	213
Structure Currently in Use Flag	213
Command Reference: FormatDateTimeProperty()	219
Command Reference: FormatTableString()	221
Command Reference: Free()	223
Command Reference: FreeTable()	225
60_Command_FTPGet.pdf	227
Command Reference: InsertTimeSeriesIntoEnsemble ()	229
Command Reference: LagK()	231
60_Command_LookupTimeSeriesFromTable.pdf	235
Command Reference: ManipulateTableString()	239
Command Reference: Multiply()	241

Command Reference	: NewDayTSFromMonthAndDayTS()	243
Command Reference	: NewEndOfMonthTSFromDayTS()	247
Command Reference	: NewEnsemble ()	251
Examples		254
Command Reference	e: NewStatisticTimeSeries()	255
Examples		257
Command Reference	e: NewStatisticTimeSeriesFromEnsemble()	259
Examples		262
Command Reference	e: NewStatisticYearTS()	265
Example		270
Command Reference	: NewTable ()	271
Command Reference	: NewTimeSeries()	273
Command Reference	: NewTreeView()	275
Command Reference	: Normalize()	277
Command Reference	: OpenHydroBase()	279
Command Reference	: PrintTextFile()	281
Command Reference	: ProcessTSProduct()	285

Command Reference: ProfileCommands()	289
Command Reference: ReadDateValue()	293
Command Reference: ReadDelimitedFile()	295
Command Reference: ReadHecDss()	301
Command Reference: ReadHydroBase()	303
Command Reference: ReadMODSIM()	309
Command Reference: ReadNrcsAwdb()	311
Command Reference: ReadPatternFile()	313
Command Reference: ReadPropertiesFromFile()	315
Command Reference: ReadRccAcis()	317
Command Reference: ReadReclamationHDB()	321
60_Command_ReadRiversideDB.pdf	329
Command Reference: ReadRiverWare()	331
Command Reference: ReadStateCU()	333
Command Reference: ReadStateCUB()	335
Command Reference: ReadStateMod()	337
Command Reference: ReadStateModB()	339

Command Reference: ReadTableFromDataStore()	341
Command Reference: ReadTableFromDBF()	345
60_Command_ReadTableFromDelimitedFile.pdf	347
Command Reference: ReadTableFromExcel()	351
Command Reference: ReadTimeSeries()	355
Command Reference: ReadTimeSeriesList()	357
Command Reference: ReadUsgsNwisDaily()	361
Command Reference: ReadUsgsNwisGroundwater()	365
Command Reference: ReadUsgsNwisInstantaneous()	369
60_Command_ReadUsgsNwisRdb.pdf	373
Command Reference: ReadWaterML()	375
60_Command_ReadWaterOneFlow.pdf	377
Command Reference: RelativeDiff()	379
Command Reference: RemoveFile()	383
Command Reference: RemoveTableRowsFromDataStore()	385
60_Command_ReplaceValue.pdf	387
Command Reference: ResequenceTimeSeriesData()	391

Command Reference:	RunCommands()	395
Command Reference:	RunDSSUTL()	397
Command Reference:	RunningAverage()	401
Command Reference:	RunningStatisticTimeSeries()	405
Command Reference:	RunProgram()	411
60_Command_RunPyt	thon.pdf	415
Command Reference:	Scale()	419
Command Reference:	SelectTimeSeries()	
Command Reference:	SetAutoExtendPeriod()	423
Command Reference:	SetAveragePeriod()	425
Command Reference:	SetConstant()	427
Command Reference:	SetDataValue()	429
Command Reference:	SetDebugLevel()	431
Command Reference:	SetFromTS()	433
Command Reference:	SetIgnoreLEZero()	437
Command Reference:	SetIncludeMissingTS()	_439
Command Reference:	SetInputPeriod()	_441

Command Reference:	SetOutputPeriod()	443
Command Reference:	SetOutputYearType()	445
Command Reference:	SetPatternFile()	447
60_Command_SetPro	perty.pdf	449
Command Reference:	SetTimeSeriesPropertiesFromTable()	451
Command Reference:	SetTimeSeriesProperty()	453
Command Reference:	SetToMax()	455
Command Reference:	SetToMin()	457
Command Reference:	SetWarningLevel()	459
Command Reference:	SetWorkingDir()	461
Command Reference:	ShiftTimeByInterval()	463
Command Reference:	SortTimeSeries()	465
Command Reference:	StartLog()	467
Command Reference:	StartRegressionTestResultsReport()	469
Command Reference:	StateModMax()	471
Command Reference:	Subtract()	473
Command Reference:	TableMath()	475

60_Command_TableTimeSeriesMath.pdf	477
Command Reference: TableToTimeSeries()	479
Command Reference: TimeSeriesToTable()	487
Command Reference: VariableLagK()	493
60_Command_WebGet.pdf	499
Command Reference: WeightTraces()	501
Command Reference: WriteCheckFile()	505
60_Command_WriteDateValue.pdf	507
Command Reference: WriteHecDss()	509
Command Reference: WritePropertiesToFile()	513
Command Reference: WriteProperty()	515
Command Reference: WriteReclamationHDB()	517
Command Reference: WriteRiversideDB()	525
Command Reference: WriteRiverWare()	531
Command Reference: WriteStateCU()	533
Command Reference: WriteStateMod()	535
Command Reference: WriteSummary()	_537

Command Reference: WriteTableToDataStore()	539
60_Command_WriteTableToDelimitedFile.pdf	543
Command Reference: WriteTableToHTML()	545
Command Reference: WriteTimeSeriesProperty()	547
Command Reference: WriteTimeSeriesToDataStore()	549
Command Reference: WriteTimeSeriesToJson()	553
Command Reference: WriteTimeSeriesToKmI()	557
60_Command_WriteWaterML.pdf	559
99_TSTool_Spine_CDSS_CommandReference.pdf	561

Blank Page

This page is intentionally blank.

TSTool Syntax Guide

Version 10.13.00, 2012-10-23

TSTool commands use a number of syntax (notation) conventions that have been implemented over time in response to functionality requirements. This appendix provides a summary of the syntax as a guide for users and future software development. Syntax standards listed here should be used where possible to ensure consistency in software features.

Where appropriate, notation has been selected based on other efforts. For example, date/time formatting is patterned after the C language strftime() function, which has been available for over 30 years. In cases where notation is specific to TSTool, an attempt has been made to consider common notation standards that can be adapted for TSTool. In cases where one or more existing standards are in place, the most common or relevant standard for TSTool has been selected, with an option to implement additional standards in the future.

Although standard notation is utilized into the software design, support for notation in commands may be incomplete because some commands use older code. For example, the ability to use properties to specify command parameters is implemented only for commands that have specifically required such functionality. Future software enhancements will continue to update code to universally provide standard features.

The following sections are ordered roughly in the order that topics are likely to be encountered, with headings grouped according to major TSTool design elements.

Commands – Basic Syntax

The syntax for commands adheres to the following syntax:

```
CommandName(Parameter1=Value1,Parameter2=Value2)
```

The CommandName matches a command from the TSTool **Commands** menu and as documented in the **Command Reference** documentation, which describes command parameters. Any parameter value can be surrounded by double quotes to protect whitespace and other characters (such as characters used in the command itself including equal sign, comma, and parenthesis). However, double quotes typically are used only for parameter values that are text, dates, filenames, etc., and not simple data such as numbers.

Commands currently cannot be indented, although this may be enabled in the future.

Command names and parameters generally are case-insensitive. However, "camel" notation (mixed upper and lower-case letters) is used to improve readability. In some cases this results in an acronym being converted from uppercase to missed-case (e.g., "USGS" becomes "Usgs").

Commands – Referring to Parameters

In some cases it is necessary to set one command parameter using the value of another command parameter. This capability has been implemented for a small number of commands, for example NewStatisticEnsemble(). To reference a command parameter in another parameter, use the notation:

```
CommandName(Parameter1=Value1,Parameter2="${C:Parameter1}...etc")
```

This notation uses C: to provide a "command scope", similar to how the TS: notation provides a scope for time series properties (discussed below).

Commands – Comments

Command files use comments to disable commands without deleting them. A # character at the start of a line indicates a one-line comment. A group of lines that start with /* and end with */ indicate a block of comments and all intervening commands will be ignored in processing.

Commands – Time Series Identifiers

Time series identifiers (TDIDs) uniquely identify time series and are discussed in detail in the **Introduction** chapter. TSID commands, which match the syntax discussed below, are created when using the data browsing features of the TSTool main interface, are specified by some commands, and can be edited manually if the user edits a command file with a text editor. These commands are essentially "read" commands that use default parameters (e.g., the global input period and do not assign an alias).

There are two main forms of TSIDs:

Location.DataSource.DataType.Interval[.Scenario]~DataStore[~FileName] Location.DataSource.DataType.Interval[.Scenario]

The first form of the TSID is a unique identifier for a time series, similar to a Universal Resource Indicator (URI) for a web page, and allows software to locate the data for reading. The datastore (or "input type" and corresponding filename) allow the software to find the source of the data.

The second form of the TSID is a unique identifier for a time series within TSTool and is used after reading the data. In cases where more than one time series will have the same TSID after reading, an alias can be assigned (see the **Introduction** chapter and the **Time Series – Properties** section below).

TSIDs may be more complex if, for example, the data type requires the use of multiple parts for uniqueness. In this case, a dash may be used (e.g., Streamflow-Max). The datastore appendices describe how time series properties from the original source are mapped into TSID notation.

Processor – Properties

TSTool commands are processed, and data managed, by a time series "processor". The processor interacts with all commands and is controlled with properties that initially have internal defaults (e.g., the default is to read all available data rather than a specified input period). Properties that control the processor are set with specific commands (e.g., SetInputPeriod()) and user-supplied properties can be set with the SetProperty() command (e.g., it is common to manage file locations and dates used in processing). The ReadPropertiesFromFile() and WritePropertiesToFile() commands can be used to save and manage properties outside of TSTool.

Processor properties can be used to specify parameters for commands using the following notation:

\${PropertyName}

For example, some commands that operate on files allow the property \${WorkingDir} to be used for the current working directory. Refer to command documentation to determine if properties are supported. Additional support is being phased in as resources allow and to satisfy requirements.

Properties internally have a specific data type. For example the input start and end use a "DateTime" object type supported by TSTool. All properties will convert to strings, for example when saved to a properties file. Some care may need to be taken to use properties of an appropriate type but a general rule is that properties used in file names or similar can simply be handled as strings.

Time Series – Properties

Time series properties are specific to individual time series. Some internal properties are handled as specific data values (e.g., data units are a string associated with a time series) whereas user-assigned properties are assigned to the time series as a list (see the SetTimeSeriesProperty() command). Time series properties are used by some commands to control the command functionality and output. For example, many commands that create time series allow the alias to be assigned using time series properties. The following notation is used when dealing with time series properties:

- % formatting Many commands that create time series allow the Alias or other parameters to be assigned using % formatters. For example, Alias="%L" indicates that the time series alias should be assigned to the location part of the time series identifier, which for a read command is controlled by the rules of the command. Format specifiers are provided for fundamental time series data properties that are required for each time series (units, location, data type, etc.).
- TS: Property reference Some command parameters need to specify a time series property by reference but the above formatting notation is inappropriate. In this case, the following design is being phased in (under development):
 - o TS:PropertyName
 - o \${TS:PropertyName}

The latter notation allows a time series property to be specified using a notation similar to processor properties, but the TS: prefix differentiates the property from the more generic processor notation.

Note that using time series properties in commands in some cases must be limited because TSTool uses a "discovery mode" to partially read/create time series so that they can be listed in "downstream" commands. Too much reliance on internal time series data might require reading more time series data, which can greatly decrease software performance in discovery mode.

Time Series – Data Flags

Time series data values (measurements, observations, etc.) are managed internally as lists of date/time, value, and flag data. A data flag is a string that is assigned a value based on one of the following cases:

- missing data value with a flag
- non-missing data value and no data flag
- non-missing data value with a flag

Data flags are useful for indicating the quality of a data value (e.g., E might indicate estimated) and for tracking how specific data values are manipulated (e.g., append to the data flag as specific actions are taken). TSTool generally does not implement a standard for data flags because flags used in input data may vary. However, some commands allow setting flags based on simple rules. For example fill

commands generally have a FillFlag parameter to set the data flag for filled values. The following table lists notation that is used to provide flexibility in setting data flags. The first notation option is used by most commands and the other options are being phased in (refer to command documentation to confirm available data flag functionality).

Notation	Description
x	Set the data flag to x regardless of whether it has already been set.
+X	If the flag has not been set, set to x.
	If the flag has been set, append x.
	This notation is useful when there are no concerns about the order of characters in
	multi-character flags.
+,x	If the flag has not been set, set to x.
	If the flag has been set, append, x.
	This notation is useful when flags are set for each step in a process.
Auto	Some commands allow Auto or another string as the flag. In this case, the
	command will decide the flag value that is assigned, based on some condition. For
	example, the flag may be assigned based on which time series was used to fill the
	value.

Command Parameter Notation Used When Setting Data Flag

Data points in graphs can be labeled in various ways to facilitate interpretation of the data. For example, each data point can be labeled with the data value, flag, or other information. Similar to time series property formatting, the notation %q in graph data point labels indicates that the points should be labeled with the data flag.

Date/Time

Date/time notation is ubiquitous when dealing with time series, and includes use for the following:

- date/time associated with specific data values
- date/time pair that indicates data period or subset of the full data period
- date/time pair indicating a window within each year

In most cases TSTool will default to displaying date/time using the ISO 8601 specification, which is essentially YYYY-MM-DD hh:mm:ss. Not only does this implement a global standard, but it also ensures that date/times are formatted in a way that allows sequential sorting. The precision of formatted date/times is generally consistent with the time series data interval (e.g., monthly time series will have dates that are by default formatted as YYYY-MM).

It may be desirable or necessary to specify the format of date/times, for example to indicate the format for output or parsing. When this is necessary, the notation utilizes an optional format type prefix and the format itself, as follows:

- The default is to parse the date/time string by matching ISO or other common formats (this works most of the time). The default output format is the ISO format.
- C:%m%d%y Indicates that a C-style format is being used, where the formats match the UNIX strftime() function syntax. See the FormatDateTimeProperty() command documentation.

• In the future support for Microsoft Excel or other notation may be added (e.g., MM-YYYY).

Regular Expression – Notation

Regular expressions are strings that indicate how to match patterns, for example to match file names or time series identifiers (see: http://en.wikipedia.org/wiki/Regular_expression). Many software tools and programming languages implement regular expressions to facilitate efficient data processing; however, the notation can be confusing, especially if not used on a regular basis. Within TSTool the following regular expression notations are used:

- "globbing" This notation was popularized by UNIX and in simple terms relies on the * character to indicate "match zero or more characters". For example, it can be used to match a list of comma-separated-value files using the expression *.csv.
- Regular expression syntax True regular expression syntax provides much more power than globbing notation, but also introduces complexity in notation. TSTool is written in Java and internally relies on Java's regular expression syntax.

In most cases, TSTool commands and configuration files use the simpler globbing notation because it is easier to use and explain. However, in some cases the more powerful regular expression syntax is needed. Where confusion may result, the command documentation clearly indicates the syntax that is supported, and commands may accept the notation glob:xxxx or regex:xxxx to indicate the type of regular expression that is being specified.

Template – Syntax

Template files are used when processing is automated to iterate over one or more lists of input data. For example, the same 10 commands may be executed for each of 100 time series. TSTool uses the FreeMarker template library to process templates. See the ExpandTemplateFile() command documentation for an explanation of syntax.

Configuration File – TSTool Configuration File

The TSTool configuration file uses a simple notation to assign properties:

```
[Section]
```

Property = Value

The [Section] notation is internally used as a prefix on the property name (e.g., Section.Property = Value). Comments are lines that start with #. Property values can be surrounded by double quotes.

Configuration File – Datastore Properties

Datastore property files use the simple notation:

```
Property = Value
```

Comments are lines that start with #. Property values can be surrounded by double quotes. The specific property values are described in TSTool datastore appendices.

Configuration File – Time Series Product Files

Time series product configuration file uses a simple notation to assign properties:

[Section]

Property = Value

The [Section] notation is internally used as a prefix on the property name (e.g., Section.Property = Value). Comments are lines that start with #. Property values can be surrounded by double quotes. See also the **TSView Time Series Viewing Tools** appendix.

Command Glossary

Version 07.01.00, 2007-03-02, Acrobat Distiller

The following parameter names and terms are used throughout TSTool commands. A term indicated in **bold** font is a definition. A term indicated in **bold** courier font is a parameter name. Parameters that are infrequently used are listed with the corresponding commands. Common parameters are defined but long lists of corresponding commands are not provided.

- **a1**,... Used with the ARMA() command.
- **b1**,... Used with the ARMA() command.
- Alias A (generally) short identifier for a time series, used in place of the TSID, which simplifies commands. The Alias and TSID values are interchangeable when used as parameters to commands and may both be referred to as TSID in command editors. See also TSID.
- Alias A (generally) short identifier for a time series, used in place of the TSID, which simplifies commands. When used to create/read a time series, the syntax of a command is typically similar to: TS Alias = command(...). See also TSID.
- AddTSID Time series identifiers for time series to add. See the add() command.
- AddValue A numerical value to be added to a time series. See the addConstant() command.
- **AdjustMethod** Indicates the method used when adjusting a time series. See the adjustExtremes() command.
- AllowMissingCount Indicate how many missing data values are allowed in an interval, in order to allow processing. See the changeInterval() and newStatisticYearTS() commands.
- **AnalysisEnd** A DateTime that indicates the end of an analysis.
- AnalysisMonth One or more months indicating which months should be processed in the analysis. See the fillRegression() command.
- **AnalysisStart** A DateTime that indicates the start of an analysis.
- **ARMAInterval** The data interval used in an ARMA analysis. See the ARMA() command.
- **AutoExtendPeriod** Indicate whether to autoextend the period of all time series to be the output period. See the setAutoExtendPeriod()command.
- AverageEnd A DateTime that indicates the end of an averaging analysis. See the setAveragePeriod() command.
- AverageMethod Indicate the method to use when averaging data. See the runningAverage() command.

- **AverageStart** A DateTime that indicates the start of an averaging analysis. See the setAveragePeriod() command.
- **BlendMethod** The method to use when blending time series. See the blend() command.
- **BlendTSID** Time series identifiers for time series to blend into main time series. See the blend() command.
- **Bracket** The number of days to search forward and back for a non-missing value. See the newEndOfMonthTSFromDayTS() and runningAverage() commands.
- **CalculateFactorHow** Indicate how to calculate the factor used when prorating values. See the fillProrate() command.
- **CommandLine** The command line for a program to run. See the runProgram() command.
- ConstantValue A numerical value used for filling, etc. See the fillConstant(),
 setConstant() and setConstantBefore() command.
- **DatabaseName** The name of a database, when making a database connection. See the openHydroBase() command.
- **DatabaseServer** The name of a database server, when making a database connection. See the openHydroBase() command.
- **DataSource** The data source to use when forming a TSID. See the createFromList() command.
- **DataType** The data source to use when forming a TSID. See the createFromList() command.
- **DateTime** A date/time value, typically represented as a string, which indicates a point in time. Date/time strings have a precision that is interpreted by the software. For example, the date/time string 1990 has a precision of year, whereas the string 1990-01-12 has a precision of day.
- **DayTSID** Time series identifier for a daily time series. See the newDayTSFromMonthAndDayTS() command.
- **DefaultFlow** Indicate a default flow value to be used if observations or filled values cannot be found. See the lagK() command.
- Delim The delimiter character(s) used when processing delimited files. See the
 createFromList() command.
- **DependentAnalysisEnd** A DateTime that indicates the end of an analysis of dependent time series. See the fillMOVE2() command.

- **DependentAnalysisStart** A DateTime that indicates the start of an analysis of dependent time series. See the fillMOVE2() command.
- **Description** The description (name) for a time series. See the newTimeSeries() command.
- **DeselectAllFirst** Indicate whether to deselect all time series before processing the command. See the selectTimeSeries() command.
- **DiffFlag** A character flag used to indicate when time series values are different. See the compareTimeSeries() command.
- **Divisor** Indicate which time series is the divisor. See the relativeDiff() command.
- **DivisorTSID** Time series identifier for time series to divide another time series. See the divide() command.
- **ExtremeToAdjust** Indicates whether the maximum or minimum value in a time series should be adjusted. See the adjustExtremes() command.
- FillDirection Indicate which direction (Foreward or Backward) filling should occur. See the
 fillProrate() and fillRepeat() commands.
- **FillEnd** A DateTime that indicates the end of a fill process.
- FillFlag A character flag used to indicate when time series values are filled. See the
 fillhistMonthAverage(), fillHistYearAverage(), fillMOVE2(),
 fillProrate(), and fillRegression() commands.
- FillNearest Indicate whether missing data values should be filled with the nearest non-missing
 value. See the lagK() command.
- **FillStart** A DateTime that indicates the start of fill process.
- FillUsingCIUFlag A character flag used to indicate when time series values are filled with CIU
 information (see FillUsingCIU). See the fillUsingDiversionComments()
 command.
- FillUsingDivComments Indicate whether missing data values should be filled using diversion comments from HydroBase. Additional zeros will be included in data. See the readHydroBase() and TS Alias = readHydroBase() commands. Also see the fillUsingDiversionComments() command.

- **FillUsingDivCommentsFlag** A character flag used to indicate when time series values are filled. See the readHydroBase(), and TS Alias = readHydroBase() commands.
- HandleMissingHow Indicate how to handle missing data values when processing time series. For example, when adding time series, missing values can be ignored or can result in a missing value in the result. See the add(), cumulate(), and subtract() commands.
- **HandleMissingTSHow** Indicate how to handle missing time series during processing. See the createFromList() command.
- ID The identifier to match in a file. See the createFromList() command.
- **IgnoreLEZero** Indicate whether values less than or equal to zero should be ignored when computing historical averages for time series. See the setIgnoreLEZero() command.
- IncludeMissingTS Indicate whether missing time series (e.g., from a query or read) should
 automatically be included using default information. See the setIncludeMissingTS()
 command.
- IndependentTSID Time series identifier for the independent time series being processed. See the
 fillFromTS(), fillMOVE2(), fillProrate(), fillRegression(),
 setFromTS(), and setMax() commands.
- **InflowStates** The inflow states (initial states) when routing a flow time series. See the lagK() command.
- InitialValue Indicate an initial value needed for computations. See the fillProrate() and newTimeSeries() commands.
- **InputEnd** A DateTime that indicates the end of a file read or a database query.
- **InputFile, InputFile1, InputFile2** The name of an input file to read, used by many commands.
- **InputName** The input name to use when forming a TSID. See the createFromList() command.
- **InputStart** A DateTime that indicates the start of file read or a database query.
- **InputType** The input type to use when forming a TSID. See the createFromList() command.
- Intercept The intercept to be enforced when determining a line of best fit. See the
 fillRegression() command.

- κ The attenuation factor used when routing a flow time series. See the lagk() command.
- Lag The lag term for routing a flow time series. See the lagK() command.
- Length The length of a time series trace. See the createTraces() command.
- ListFile The name of an input or output list (delimited) file to be written or read, specified using a
 relative or absolute path. See the createFromList() command.
- LogFile The name of the log file, specified using a relative or absolute path. See the
 setLogFile() command.
- LogFileLevel The level for messages printed to the log file. See the setDebugLevel() and setWarningLevel() commands.
- **MatchDataType** Indicate whether the data type part of a TSID should be matched when comparing time series identifiers. See the compareTimeSeries() command.
- MaxIntervals The maximum number of intervals to process, typically used to limit a fill or analysis
 procedure. See the adjustExtremes(), fillInterpolate(), and fillRepeat()
 commands.
- **MaxValue** The maximum value in an analysis. See the normalize() and replaceValue() commands.
- **Method** A method used when processing data, used to more specifically control how a command functions. See the analyzePattern() and disaggregate() commands.
- **MinValue** The minimum value in an analysis. See the normalize() and replaceValue() commands.
- **MinValueHow** Indicate how to determine the minimum value in an analysis. See the normalize() command.
- **MissingValue** A numerical value used for missing data in time series. See the writeStateMod() command.
- **MonthTSID** Time series identifier for a monthly time series. See the newDayTSFromMonthAndDayTS() command.
- MonthValues Monthly values used for filling, etc. See the setConstant() command.
- MultiplierTSID Time series identifier for the time series to multiply () command.

- Multiplier Value(s) to multiply time series value(s) by when processing. See the
 shiftTimeByInterval() command.
- **NewDataType** The data type for a new time series, typically used where the data type must be explicitly defined and is not determined from a TSID. See also NewTSID. See the changeInterval() command.
- **NewInterval** The data interval for a new time series, typically used where the interval must be explicitly defined and is not determined from a TSID. See also NewTSID. See the changeInterval() command.
- **NewTimeScale** The new time scale (ACCM for accumulated data, INST for instantaneous data, MEAN for mean data) for a time series. See the changeInterval() command.
- **NewTSID** The new time series identifier for a time series, used with commands that create new time series. See the copy() and newDayTSFromMonthAndDayTS() commands.
- NewUnits The new data units for a time series. See the converDataUnits(), TS Alias =
 readDateValue(), TS Alias = readMODSIM(), TS Alias = readNWSCard(),
 and TS Alias = readRiverWare() commands.
- **NewValue** The new value in an analysis. See the replaceValue() and setDataValue() commands.
- **NumEquations** Number of equations to use when analyzing data (typically one or monthly equations). See the fillMOVE2() and fillRegression() commands.
- **ObsTSID** The time series identifier for an observed time series. See the lagK() command.
- OdbcDSN The Open Database Connectivity (ODBC) Data Source Name (DNS) for a database connection. See the openHydroBase() command.
- **OldTimeScale** The old time scale (ACCM for accumulated data, INST for instantaneous data, MEAN for mean data) for a time series. See the changeInterval() command.
- **OutflowStates** The outflow states (initial states) when routing a flow time series. See the lagK() command.
- **OutputEnd** A DateTime that indicates the end of output.
- **OutputFile** The name of an output file to be written, specified using a relative or absolute path.
- **OutputStart** A DateTime that indicates the start of output.
- **OutputYearType** Indicate the type of year (e.g., calendar year, water year) for output. See the setOutputYearType() command.
- **PatternFile** The file name for a pattern file. See setPatternFile() command.

- **PatternID** An identifier for a pattern (e.g., WET, DRY, AVG). See the analyzePattern() and fillPattern() commands.
- **Pos** The position in the time series list. See the deselectTimeSeries() and selectTimeSeries() commands.
- \mathbf{pP} Used with the ARMA() command.
- Precision The precision (number of digits after the decimal point) used when comparing values or formatting values for output. See the compareTimeSeries(), writeRiverWare(), and writeStateMod() commands.
- **QueryEnd** A DateTime that indicates the end of a database query. The InputEnd parameter is preferred and is used in new commands.
- **QueryStart** A DateTime that indicates the start of database query. The InputStart parameter is preferred and is used in new commands.
- qQ-Used with the ARMA() command.
- **Read24HourAsDay** Indicate that a time series with data interval 24Hour should be automatically read as Day. See the readNwsCard() and TS Alias = readNwsCard() commands.
- **ReadEnd** A DateTime that indicates the end of a file read. See the readNWSCard() command. The InputEnd parameter is preferred.
- **ReadStart** A DateTime that indicates the start of file read. See the readNWSCard() command. The InputStart parameter is preferred.
- **RecalcLimits** Recalculate the data limits for a time series, usually when supplemental raw data are being supplied after an initial read. See the fillUsingDiversionComments() command (used with the State of Colorado's HydroBase input type).
- **ReferenceDate** The starting date for a time series trace. See the createTraces() command.
- Reset A DateTime field that indicates when to reset data values in a manipulation. For example, a
 time series may be set to zero at the start of each year when used with the cumulate()
 command. See the cumulate() command.
- RunMode Typically used to indicate whether the command should be processed in batch mode, via the GUI, or both. See the openHydroBase(), processTSProduct(), and setWorkingDir() commands.
- **Scale** A scale factor to be applied to data. See the writeRiverWare() command.

$\label{eq:scalevalue} \textbf{ScaleValue} - A \text{ numerical value used for scaling time series. See the scale() command.}$

- **Scenario** The scenario to use when forming a TSID. See the createFromList() command.
- ScreenLevel The level for messages printed to the screen (console). See the setDebugLevel()
 and setWarningLevel() commands.
- SelectAllFirst Indicate whether to select all time series before processing the command. See the
 deselectTimeSeries()command.
- SearchStart A DateTime that indicates the search start date/time in an analysis. See the
 newStatisticYearTS() command.
- **SetEnd** A DateTime that indicates the end of a set process.
- Set_scale See the writeRiverWare() command.
- **SetStart** A DateTime that indicates the start of set process.
- set_units See the writeRiverWare() command.
- **ShiftDataHow** Indicate how to shift time series traces. See the createTraces() command.
- **SpecifyWeightsHow** Indicate how to specify weights when processing time series. See the TS Alias = weighTimeSeries() command.
- **Statistic** A statistic to evaluate. See the newStatisticYearTS() command.
- **SubtractTSID** Time series identifiers for time series to subtract. See the subtract() command.
- **TestValue** A test value used in an analysis. See the newStatisticYearTS() command.
- **Timeout** The timeout when running an external program, after which processing will continue. See the runProgram() command.
- **Tolerance** A value (or values) used to indicate an allowable error/difference. See the compareTimeSeries() command.
- **TransferHow** Indicate how to transfer data during processing, either according to the date/time or sequentially. The latter can be used when time series do not align on date/time (e.g., due to a shift, leap year, etc.). See the setFromTS() command.
- Transformation Indicate whether the time series data should be transformed before processing. See the fillInterpolate(), fillMOVE2(), and fillRegression() commands.
- TSID Time series identifier, which is used to uniquely identify a time series. In full notation, this consists of a string similar to the following: Location.DataSource.DataType.Interval.Scenario~InputType~InputName. In abbreviated form, the InputType and InputName are often omitted. The InputType and InputName are typically used only by read and write commands. Because a TSID may be long (especially when file

names are used for the InputName), an Alias may be assigned to the time series. The TSID parameter is typically used in commands for the time series that is being processed. See also Alias.

- **TSID** When used as a command parameter the time series identifier indicates the time series to be processed. The TSID or alias can typically be specified. See also Alias.
- **TSID1** Time series identifier for the first daily time series in a command. See the fillDayTSFrom2MonthTSAndlDayTS() command.
- **TSID2** Time series identifier for the first daily time series in a command. See the fillDayTSFrom2MonthTSAndlDayTS() command.
- TSID_D1 Time series identifier for the first time series in a command. See the TS Alias =
 relativeDiff() command.
- **TSID_D2** Time series identifier for the second daily time series in a command. See the fillDayTSFrom2MonthTSAnd1DayTS() command.
- **TSID_M1** Time series identifier for the first monthly time series in a command. See the fillDayTSFrom2MonthTSAnd1DayTS() command.
- **TSID_M2** Time series identifier for the second monthly time series in a command. See the fillDayTSFrom2MonthTSAnd1DayTS() command.
- TSList Indicates how the list of time series is determined. Typical values are AllTS (process all time series), AllMatchingTSID (process all time series having identifiers that match the TSID parameter), SelectedTS (process all time series that have been selected with the selectTimeSeries() and deselectTimeSeries() commands). This parameter is being phased in to allow more flexibility when processing time series.
- **TSProductFile** The name of a time series product (TSProduct) file. See the processTSProduct() command.
- Units The data units for a time series. See the newTimeSeries(), TS Alias =
 readNWSRFSFS5Files(), and writeRiverWare() commands.
- **Version** Indicates the file version, to allow the software to handle different data formats. See the readStateModB() command.
- **View** Indicate whether a product should be graphically previewed (as opposed to simply writing an output file). See the processTSProduct() command.
- **UseStoredProcedures** Indicates whether stored procedures should be used (versus straight SQL calls). This is being used to transition HydroBase queries to stored procedures. See the openHydroBase() command.

- WarnIfDifferent Indicates whether a warning should be generated if data differences are detected. See the compareTimeSeries() and compareFiles() commands.
- WarnIfSame Indicates whether a warning should be generated if data differences are NOT detected. See the compareTimeSeries() and compareFiles() commands.
- Weight -Weight(s) used when processing time series. See the TS Alias =
 weighTimeSeries() command.
- Where1, Where2 Input filter information used when reading/querying data. See the readHydroBase() command.

Year - Specify year(s) of interest. See the TS Alias = weighTimeSeries() command.

Command Reference:

Comment line Version 10.12.00, 2012-09-10

The # command indicates single-line comments. Commands can be converted to and from # comments by right-clicking on a command in TSTool and selecting from the popup menu. See also the /* and */ comment block commands, which are used to comment multiple commands.

The following table lists annotation tags that can be placed in comments to provide additional information to software that processes the commands, using notation similar to the following:

#@expectedStatus Failure

Parameter	Command that Uses	Description
@expectedStatus	RunCommands()	Used to help the test framework know if an
Failure		error or warning is expected, in which case a
		passing test can occur even if the command
@expectedStatus		status is not "success".
Warning		
@os Windows	CreateRegressionTest	Used to filter out test command files that are
@os UNIX	CommandFile()	not appropriate for the operating system.
		Linux is included in UNIX.
@readOnly	TSTool main interface and	Indicates that the command file should not be
	command editors	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. This
		tag is often used with templates to protect the
		template from mistakenly being edited and
		saved in TSTool (TSTool does not currently
		allow editing templates within the interface).
@testSuite ABC	CreateRegressionTest	Used to filter out test command files that are
	CommandFile()	not appropriate for the operating system

Command # Comment Tags

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

🛃 Edit # (Cor	nments												×
Enter one of	r m	ore comme	nts (le	ading	# will be a	dded automa	atically	/ if not sh	iown).					
See also the	e /*	and */ com	mands	s for n	nulti-line co	mments, whi	ich ar	e useful	for cor	nmenti	ng out mul	tiple con	nmands.	
	0	-	LO		20	30		40		50		60	70	
	12	34567890)1234	15678	39012345	678901234	4567	890123	45678	9012	3456789	012345	56789012	234567890
	#	Uncomme	nt th	ne fo	ollowing	f command	to	regene	rate	the	expecte	d resu	ults fi	le.
	#	WriteDa	teVa.	lue((OutputFi	le="Expe	cted	Result	s∖Tes	st_Wr	iteDate	Value_	_Commen	tBlock_ou
	Į.													
Comments:														
		1												
	Ľ	I												
									-					
						Canc	el	OK						
														Common

Command Editor

The command syntax is as follows:

Some text

A sample command file is as follows:

#				
#	Some	comments		
#				

Command Reference: /*

Comment block start Version 10.12.00, 2011-09-10

The /* command starts a multi-line comment block and is useful for inserting long comments or temporarily commenting out blocks of commands. See also the */ and # commands. Commands between the /* and */ are not converted to comments but are skipped during processing. See also the # comment documentation for information about comment @ annotations.

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

🛃 Edit /* command	×
This command starts a comment block, and is used to comment out multiple commands. Use the */ command to end the block of comments.	
Command:	
Cancel OK	
	(

/* Command Editor

The command syntax is as follows:

/*

A sample command file is as follows:

/*	
SomeCommentedOutCommands()	
*/	

This page is intentionally blank.
Command Reference: */

Comment block end Version 08.17.00, 2008-10-01

The */ command ends a multi-line comment block and is useful for inserting long comments or temporarily commenting out blocks of commands. See also the /* and # commands. Commands between the /* and */ are not converted to comments but are skipped during processing.

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

፼Edit */ command	×
This command ends a multi-line comment block, which is useful for c	mmenting out multiple commands.
Use the /* command to start the comment block.	
See also the # command for commenting single lines.	
Command:	
Cancel OK	
	C

*/ Command Editor

The command syntax is as follows:

*/

A sample command file is as follows:

/ *
SomeCommentedOutCommands()
* /

This page is intentionally blank.

Command Reference: Time Series Identifier (TSID)

Read a single time series given the time series identifier

A time series identifier (TSID) command reads a single time series. In order to read the time series from a persistent format (database, file, or web site), the TSID must contain the input type, and if necessary, the input name. For example, a TSID command for the State of Colorado's StateMod model file format is of the form:

LocationID...Interval~StateMod~Filename

Refer to the **StateMod Input Type** appendix for a full description of the file format. Appendices are available for all input types. A TSID command for a StateMod file is generated as follows:

- 1. Select the StateMod input type and appropriate time step in the main TSTool window.
- 2. Press the **Get Time Series List** button to list time series. A dialog will prompt for the StateMod file and after selection the first year of data from the file will be read to get a list of identifiers. The interval that is specified (Month or Day) indicates whether the file is a monthly or daily format. The time series will be listed in the time series list in TSTool.
- 3. Select one or more time series from the list and copy to commands.

The following dialog is used to edit the command and illustrates the syntax of the command. Limited checks are done while editing the command. However, once committed, TSTool will attempt to read the time series metadata and will issue a warning if unable to read the data. Time series identifiers that include filenames should typically be adjusted to a relative path to allow the files to be moved to another location and run without errors. Use the **Remove Working Directory** button to remove the working directory (or **Add Working Directory**) to add it.

😹 Edit TSID() Command	×	
This command reads a single time series given a time series identifier that includes the input name (database, file) information.		
See also the ReadTimeSeries() command, which assigns an alias to a time series. The alias may be more convenient to use than the long time series identifie	er.	
Read commands for specific input types are also available, generally offer more options, and should be used if available.		
See also the CreateFromList() command.		
Specify a full path or relative path (relative to working directory) for a file to read.		
It is strongly recommended that relative paths be used in commands if possible.		
The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\TSID_StateMod		
Time series identifier: 0675400DAY~StateMod~Results/Test_TSID_StateMod_out.stm		
0675400DAY~StateMod~Results/Test_TSID_StateMod_out.stm		
Command:		
Add Working Directory Cancel OK		
TSID_State	Mod	

TSID Command Editor for a Time Series Read From a StateMod File

The following example is for a TSID for the State of Colorado's HydroBase database. In this case there is no filename in the identifier and therefore no need to adjust to a relative path.

🖒 Edit TSID() Comn	nand 🔀	
This command reads a	single time series given a time series identifier that includes the input name (database, file) information.	
See also the ReadTimes	Series() command, which assigns an alias to a time series. The alias may be more convenient to use than the long time series identifier.	
Read commands for sp	ecific input types are also available, generally offer more options, and should be used if available.	
See also the CreateFro	mList() command.	
Time series identifier:	06754000.DWR.Streamflow.Month~HydroBase	
6	06754000.DWR.Streamflow.Month~HydroBase	
Command:		
,		
Cancel OK		

TSID Command Editor for a Time Series Read From the HydroBase Database

After executing the command, the time series will have the identifier as originally requested, with no alias being assigned. Use the TS Alias = ReadTimeSeries() command to assign an alias to the time series, or use one of the specific read commands.

A sample command file to read time series from a StateMod file is as follows. In this case the absolute paths have been adjusted to relative paths using the command editor dialog. Note also that the data source and data type are not required because this information is not stored in the StateMod file.

```
09303000...MONTH~StateMod~whiteT.rih
09303400...MONTH~StateMod~whiteT.rih
```

A sample command file to read time series from the State of Colorado's HydroBase database is as follows:

```
# 06754000 - SOUTH PLATTE RIVER NEAR KERSEY
06754000.DWR.Streamflow.Month~HydroBase
# 06758500 - SOUTH PLATTE RIVER NEAR WELDONA
06758500.DWR.Streamflow.Month~HydroBase
```

Command Reference: Add()

Add one or more time series to a time series (or ensemble)

Version 09.09.01, 2010-10-18

The Add() command adds regular interval time series. The receiving time series will be set to the sum of itself and all indicated time series. See also the NewTimeSeries() command, which can create an empty time series to receive a sum. If an ensemble is being processed, another ensemble can be added, a single time series can be added to all time series in the ensemble, or a list of time series can be added to the ensemble (the number in the list must match the number of time series in the ensemble).

This command will generate an error if the time series do not have compatible units. If the units are compatible but are not the same (e.g., IN and FT), then the units of the part will be converted to the units of the sum before addition. Missing data in the parts can be ignored (do not set the sum to missing) or can result in missing values in the sum. The user should consider the implications of ignoring missing data. Time series being added must have the same data interval.

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

C Edit Add() Command	
Add one or more time series to a time series (or ens The time series to be added are selected using the AllMatchingTSID - add all previous time series with AllTS - add all previous time series. EnsembleID - add all time series for the ensemble. FirstMatchingTSID - add the first time series (befor LastMatchingTSID - add the last time series (befor SelectedTS - add time series selected with SelectT SpecifiedTSID - add time series selected from the l	emble of time series). The receiving time series (or ensemble) is modified. AddTSList parameter: matching identifiers. re this command) with matching identifier. e this command) with matching identifier. imeSeries() commands ist below
Time series to receive results:	0100501.DWR.DivTotal.Month
Ensemble to receive results:	
Time series to add (AddTSlist):	SpecifiedTSID Optional - indicates the time series to process (default=AllTS).
Add TELL (For TELISLEARMarching(STD))	0100503.DWP. DW Fotal Month
Add Ensemblett. (For Add) 5bol-Ensemble(0)	
Add specified TSID (for AddTSList=SpecifiedTSID):	0100501.DWR.DivTotal.Month 0100503.DWR.DivTotal.Month
Handle missing data how?:	IgnoreMissing Optional - how to handle missing values in time series (default=IgnoreMissing)
If time series list to add is empty?:	Optional - action if time series list to add is empty (default=Fail).
Command:	Add (TSID="0100501.DWR.DivTotal.Month", AddTSList=SpecifiedTSID, Ad dTSID="0100503.DWR.DivTotal.Month", HandleMissingHow="IgnoreMissi ng")
	Cancel OK

Add() Command Editor

The command syntax is as follows:

```
Add(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
TSID	The time series identifier or alias for the time series to receive the sum.	TSID or EnsembleID must be specified.
EnsembleID	The ensemble to receive the sum, if processing an ensemble.	TSID or EnsembleID must be specified.
AddTSList	 Indicates how the list of time series is specified, one of: AllTS – all time series before the command. AllMatchingTSID – all time series that match the AddTSID (single TSID or TSID with wildcards) will be added. EnsembleID – the time series from ensemble will be added. LastMatchingTSID – the last time series that matches the TSID (single TSID or TSID with wildcards) will be added. SelectedTS – the time series are those selected with the SelectTimeSeries() command. SpecifiedTSID – the specified list of time series given by the AddTSID parameter. 	AllTS (the time series receiving the sum will not be added to itself)
AddTSID	If the AddTSList parameter is SpecifiedTSID, provide the list of time series identifiers (or alias) to add, separated by commas. If the AddTSList parameter is AllMatchingTSID, FirstMatchingTSID, or LastMatchingTSID, specify a single TSID or a TSID with wildcards.	Must be specified if TSList= SpecifiedTSID, ignored otherwise.
AddEnsembleID	If the EnsembleID parameter is specified, providing an ensemble ID will add the ensembles.	Use if an ensemble is being added to another ensemble.
Handle MissingHow	 Indicates how to handle missing data in a time series, one of: IgnoreMissing - create a result even if missing data are encountered in one or more time series - this option is not as rigorous as the others SetMissingIfOtherMissing - set the result missing if any of the other time series values is missing SetMissingIfAnyMissing - set the result missing if any time series value involved is missing 	IgnoreMissing
IsEmpty	Action if time series list to add is empty.	L'ATT

A sample command file to add two time series from the State of Colorado's HydroBase is as follows:

```
# 0100501 - EMPIRE DITCH
0100501.DWR.DivTotal.Month~HydroBase
# 0100503 - RIVERSIDE CANAL
0100503.DWR.DivTotal.Month~HydroBase
Add(TSID="0100501.DWR.DivTotal.Month",TSList="SpecifiedTSID",
AddTSID="0100503.DWR.DivTotal.Month",HandleMissingHow=IgnoreMissing)
```

Command Reference: AddConstant()

Add a constant value to all data values in a time series (or ensemble)

Version 09.10.01, 2010-11-18

The AddConstant() command adds a constant value to each data value in a time series (or ensemble of time series) within the specified period. This command is useful, for example, when a time series needs to be adjusted for a constant bias. Another example is to adjust a reservoir total volume time series by the dead pool storage in order to compute the active storage (or inverse). Missing data values will remain missing in the result.

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

🔊 Edit AddConstant() Command 🛛 🛛 🔀		
Add a constant to the data values for the specified time series.		
Specify dates with precision approprial	e for the data, us:	e blank for all available data, OutputStart, or OutputEnd.
TS list:	AllMatchingTSID Optional - indicates the time series to process (default=AllTS).	
TSID (for TSList=AllMatchingTSID):	2003536.DWR.Re	esMeasStorage.Day 🛛
EnsembleID (for TSList=EnsembleID);		· · · · · · · · · · · · · · · · · · ·
Constant value to add:	5000	Required - constant value to add.
Analysis start date/time:		Optional - analysis start (default=full period).
Analysis end date/time:		Optional - analysis end (default=full period).
Command:	AddConstant(TSList=AllMatchingTSID,TSID="2003536.DWR.R esMeasStorage.Day",ConstantValue=5000)	
Cancel OK		

AddConstant() Command Editor

The command syntax is as follows:

```
AddConstant(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
TSList	 Indicates the list of time series to be processed, one of: AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will be modified. AllTS – all time series before the command. EnsembleID – all time series in the ensemble will be modified. LastMatchingTSID – the last time series that matches the TSID (single TSID or TSID with wildcards) will be modified. SelectedTS – the time series are those selected with the SelectTimeSeries() command. 	AllTS
TSID	The time series identifier or alias for the time series to be modified.	TSID or EnsembleID must be specified.
EnsembleID	The ensemble to be modified, if processing an ensemble.	TSID or EnsembleID must be specified.
ConstantValue	The data value to add to the time series.	None – must be specified.
AnalysisStart	The date/time to start analyzing data.	Full period.
AnalysisEnd	The date/time to end analyzing data.	Full period.

A sample commands file to process data from the State of Colorado's HydroBase is as follows:

2003536 - CONTINENTAL RES
2003536.DWR.ResMeasStorage.Day~HydroBase
AddConstant(TSList=AllMatchingTSID,TSID="2003536.DWR.ResMeasStorage.Day",
ConstantValue=5000)

CommandReference/AddConstant/Example_AddConstant_HydroBase.TSTool

Command Reference: AdjustExtremes()

Adjust the extreme values in time series data

Version 09.10.01, 2010-11-18

The AdjustExtremes() command adjusts extreme values in time series (e.g., to remove negative values from a time series that can only have values greater than or equal to zero), while preserving "mass".

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

👌 Edit AdjustExtremes() Com	mand	×	
Adjust extreme data values by conside	Adjust extreme data values by considering values to each side of extreme values.		
If the Extreme to Adjust is AdjustMinin	num, values < Extreme Value are a	djusted.	
If the Extreme to Adjust is AdjustMaxi	mum, values > Extreme Value are	adjusted.	
The Average Adjust Method replaces I	the extreme and values on each sig	de of the extreme	
by the average of all values, preservir	ng the total.		
The maximum intervals parameter indi	cates how many intervals on each :	side of the extreme can be modified.	
Specify dates with precision appropria	te for the data.		
TS list:	AllMatchingTSID 🛛 🗸	Optional - indicates the time series to process (default=AllTS).	
TSID (for TSList=AllMatchingTSID):	06759000.USGS.Streamflow.Day	▼	
EnsembleID (for TSList=EnsembleID):		×	
Adjust method:	Average 🔽	Required.	
Extreme to adjust:	AdjustMinimum 🔽	Required.	
Extreme value:	0	Required.	
Maximum intervals:	0	Optional (default=0, no limit).	
Analysis start:		Optional (default=full period).	
Analysis end:	Optional (default=full period).		
Command:	AdjustExtremes(TSList=AllMatchingTSID,TSID="06759000.USG S.Streamflow.Day",AdjustMethod=Average,ExtremeToAdjust=A djustMinimum,ExtremeValue=0,MaxIntervals=0)		
Cancel OK			

AdjustExtremes() Command Editor

The command syntax is as follows:

AdjustExtremes(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of:	Allts
	• AllMatchingTSID – all time series that match the TSID	
	(single TSID or TSID with wildcards) will be modified.	
	• AllTS – all time series before the command.	
	• EnsembleID – all time series in the ensemble will be modified.	
	• FirstMatchingTSID – the first time series that matches the	
	TSID (single TSID or TSID with wildcards) will be modified.	
	• LastMatchingTSID – the last time series that matches the	
	TSID (single TSID or TSID with wildcards) will be modified.	
	• SelectedTS – the time series are those selected with the	
math	SelectTimeSeries() command.	D 110
TSID	The time series identifier or alias for the time series to be modified,	Required if
	using the * wildcard character to match multiple time series.	ISLISU= *TSID
EnsembleID	The ensemble to be modified, if processing an ensemble	Required if
		TSList=
		EnsembleID.
AdjustMethod	Only the Average adjust method is implemented, in which adjusted	None – must be
	data values are set to the average over the adjusted period, necessary	specified.
	to maintain the total/mass of the original values. This method adjusts	
	extreme values by considering neighboring values equally on each	
	side of the point in question. When adjusting minimum values,	
	neighboring values are added until the average is above the allowed	
	extreme value, and an values that make up the sum are then set to the	
	command should only be applied to filled data. If a satisfactory	
	result cannot be reached within this limit, then the original values are	
	not changed. Changed values are listed in the time series history.	
	which is viewed with the time series properties. Applying the	
	command will result in the time series having periods of constant	
	value, with the length of the period being controlled by the	
	magnitude of the extreme value.	
Extreme	Indicate whether minimum (AdjustMinimum) or maximum	None – must be
TOAAJUST	(AdjustMaximum) values to be adjusted.	specified.
ExtremeValue	The extreme value that is the limit of acceptable values.	None – must be
MaxInterrole		specified.
maximuervais	indicates now many values on each side of a point are allowed to be	U, indicating no
AnalysisStart	The date/time to start analyzing date	Full period
AnalysisEnd	The date/time to start analyzing data.	Full period
	I no date, mile to end anaryzing data.	i un períou.

A sample command file using data from the State of Colorado's HydroBase is as follows:

```
# 06759000 - BIJOU CREEK NEAR WIGGINS, CO.
06759000.USGS.Streamflow.Day~HydroBase
AdjustExtremes(TSList=AllMatchingTSID,TSID="06759000.USGS.Streamflow.Day",
AdjustMethod=Average,ExtremeToAdjust=AdjustMinimum,ExtremeValue=0,MaxIntervals=0)
```

Command Reference: AnalyzePattern()

Determine historical average patterns for monthly time series

Version 09.05.01, 2009-10-28

The AnalyzePattern() command creates the pattern file for use with the FillPattern() command (see also SetPatternFile()). Each time series to be processed is analyzed as follows:

- 1. Create a time series to contain the pattern identifiers for each month (e.g., DRY, AVG, WET).
- 2. For each month, determine the monthly values for the time series being analyzed (e.g., find all of the January values).
- 3. Rank the values in ascending order.
- 4. Evaluate the percentile rank information for non-missing values and assign in the pattern time series an appropriate pattern identifier. For example, if the percentile values are .25 and .75, assign the first pattern identifier to values < 25% of the non-missing count, assign the second pattern identifier to non-missing values >= 25% and < 75%, and assign the third identifier to the non-missing values >= 75%.

The resulting pattern time series is then written to a file. **This command is enabled for monthly data only**. See below for an example of a fill pattern file. One or more patterns can be included in each pattern file, similar to StateMod time series files (see the **StateMod Input Type Appendix**), and multiple pattern files can be used, if appropriate.

Years Shown = Water Years # Missing monthly data filled by the Mixed Station Method, USGS 1989 # Time series identifier = 09034500.CRDSS_USGS.QME.MONTH.1 # Description = COLORADO RIVER AT HOT SULPHUR SPRINGS, CO. 10/1908 - 9/1996 ACFT WYR WET WET AVG AVG AVG WET WET 1909 09034500 AVG AVG AVG WET WET 1910 09034500 WET WET WET AVG AVG WET WET WET AVG AVG AVG AVG 1911 09034500 AVG AVG WET AVG AVG AVG AVG WET WET WET AVG WET 1912 09034500 WET WET WET WET WET AVG AVG WET WET WET WET WET ...ommitted...

The pattern file will by default contain all available data for the overlapping period and will be written in calendar year. The output period can be set with the SetOutputPeriod() command and the output year type can be set with the SetOutputYearType() command.

The following dialog is used to edit the <code>AnalyzePattern()</code> command and illustrates the syntax of the command.

😹 Edit AnalyzePattern() Com	mand	
This command creates the pattern file	for use with the FillPattern() command.	
Only monthly time series can be proces	ssed.	
Example percentiles are .25,.75, with (corresponding pattern identifiers DRY, AVG, WET.	
The working directory is: C:\Develop\T	STool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\AnalyzePattern	
TS list:	AllTS Optional - indicates the time series to process (default=AllTS).	
TSID (for TSList=AllMatchingTSID);		~
EnsembleID (for T5List=EnsembleID):		~
Method:	Percentile	
Percentile:	0.25,0.75 Required - comma-separated list of fractions (0 to 1) for cutoffs.	
PatternID:	DRY,AVG,WET Required - pattern identifiers corresponding to the fractions.	
Output file:	Div1.pat	Browse
Table ID:	Statistics Optional - identifier for table to create, containing statistics.	
Row(s) for data:	%L, %U Insert: %A - Alias Optional - data row name(s) for 1+ time series.	
Legacy behavior:	Optional - use legacy logic (error with some edge values shifted to lo	wer percentile).
Command:	AnalyzePattern(TSList=AllTS,Method=Percentile,Percentile="0.25,0.75",Pat RY,AVG,WET",OutputFile="Div1.pat",TableID="Statistics",DataRow="%L, %U")	ternID="D
Cancel		

AnalyzePattern() Command Editor

The command syntax is as follows:

AnalyzePattern(Parameter=Value,...)

Command Parameters

TSList Indicates the list of time series to be None – must be specified.	Parameter	Description	Default
 AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards). AllTS – all time series before the command. EnsembleID – all time series in the ensemble. FirstMatchingTSID – the first time series that matches the TSID (single TSID or TSID with wildcards). LastMatchingTSID – the last time series that matches the TSID (single TSID or TSID with wildcards). 	TSList	 Indicates the list of time series to be processed, one of: AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards). AllTS – all time series before the command. EnsembleID – all time series in the ensemble. FirstMatchingTSID – the first time series that matches the TSID (single TSID or TSID with wildcards). LastMatchingTSID – the last time series that matches the TSID (single TSID or TSID with wildcards). 	None – must be specified.

	 SelectedTS – the time series selected with the SelectTimeSeries() 	
	command.	
TSID	The time series identifier or alias for the time series to be processed, using the * wildcard character to match multiple time series.	Required if TSList=*TSID.
EnsembleID	The ensemble to be processed, if	Required if
	processing an ensemble.	TSList=EnsembleID.
Method	Method used to determine the patterns. Currently only Percentile is recognized.	Percentile
Percentile	A comma-separated list of percentiles for cutoffs, used when Method=Percentile. Values should be 0 to 1 (e.g., .25, .75)	None – must be specified.
PatternID	The pattern identifiers to use, corresponding to the percentiles. Specify one more than the number of percentiles (e.g., DRY, AVG, WET).	None – must be specified.
OutputFile	Output file to write, which will contain the pattern information. Currently only the StateMod pattern file format is supported.	None – must be specified.
TableID	The identifier for a new table to be created, containing the sample values for each month adjoining the percentile positions. Each time series will be listed in the first column as per the DataRow parameter. For N percentile values, the first N-1 values in the table will correspond to the last value below a percentile cutoff and the Nth value will be the first value above the Nth percentile value.	Optional – table will not be created by default.
DataRow	The contents of the first column, indicating the time series.	Location, data type, and units, if available.
Legacy	Indicates whether to duplicate legacy behavior (True) or use current behavior (default, False). A bug was fixed in TSTool 9.05.02 to correct a bug where the last value in each bin sometimes should have been in the larger cutoff bin.	False – use current behavior.

A sample command file to analyze streamflow data from the State of Colorado's HydroBase and save statistics in a table is as follows:

```
# 06720500 - SOUTH PLATTE RIVER AT HENDERSON
06720500.DWR.Streamflow.Month~HydroBase
# 06754000 - SOUTH PLATTE RIVER NEAR KERSEY
06754000.DWR.Streamflow.Month~HydroBase
AnalyzePattern(TSList=AllTS,Method=Percentile,
    Percentile="0.25,0.75",PatternID="DRY,AVG,WET",OutputFile="Div1.pat",
    TableID="Statistics",DataRow="%L, %U")
```

The following figure illustrates the resulting statistics:

😹 TSTool - Ta	able "Stati	stics"							
Time Series	Jan last value < 0.25	Jan first value > 0.75	Feb last value < 0.25	Feb first value > 0.75	Mar last value < 0.25	Mar first ∨alue > 0.75	Apr last value < 0.25	Apr first value > 0.75	May last value < 0.25
06720500, ACET	5048.01	17181.08	5551.82	17524.22	7001.75	21142.13	8223.59	27411.97	21653.87
06754000, ACFT	30724.42	47526.64	28760.75	44192.38	27832.47	48764.35	18222.41	55696.68	17764.23
k Displaying 2 rows	, 25 columns							Read	s V

AnalyzePatter_Table

Command Reference: AnalyzeNetworkPointFlow()

Analyze a node/link network to calculate "point flow" for nodes

The AnalyzeNetworkPointFlow() command takes as input information to define a "flow network", associates input time series with each node in the network, and computes mass balance time series at each node. Although the network is intended to represent a physical network such as a stream system, it also can represent other flow networks such as transportation or other mass/energy conservation systems.

This command differs from the functionality of other network analysis tools as follows:

- Daily administration tools, such as the State of Colorado's Colorado Water Rights Administration Tool (CWRAT) perform a point flow analysis for a single day, which only requires knowing one day's input values, whereas AnalyzeNetworkPointFlow() analyzes time series for a specified period.
- More sophisticated models, such as the State of Colorado's StateMod water allocation model, perform allocation decisions within each time step for the full period, whereas AnalyzeNetworkPointFlow() performs a sequence of basic time series manipulations that can be quickly configured.

It may be possible to utilize the network data from tools such as those mentioned above with the AnalyzeNetworkPointFlow() command.

The following figure illustrates the network connectivity and mass balance that is performed at each node. Currently "on-channel" reservoirs with storage are not supported and gain/loss can only be computed in non-branching networks – these features and others necessary to model more complex networks will be added in the future; however, this command is not intended to replace more complex models. Consequently the command currently is suitable for analysis of a main stem river with no on-channel reservoirs.



AnalyzeNetworkPointFlow() Network and Node Mass Balance

The network is defined as a table containing a list of node identifiers with associated properties, as illustrated in the following figure.

	А	В	С	D	E	F
1	NodeID	NodeName	NodeType	NodeDist	NodeWeight	DownstreamNodeID
2	06754000	KERSEY GAGE	StreamGage	2.0		0103816
3	0103816	EMPIRE RESERVOIR INLET (at reservoir)	Diversion	3.0	1.0	0100503
4	0100503	RIVERSIDE CANAL (at reservoir)	Diversion	4.0	1.0	0100507
5	0100507	BIJOU CANAL	Diversion	5.0	1.0	0100513
6	0100513	JACKSON LAKE INLET	Diversion	6.0	1.0	0100511
7	0100511	WELDON VALLEY DITCH	Diversion	7.0	2.0	0100512
8	0100512	JACKSON LAKE OUTLET DITCH	Return	8.0	2.0	0100514
9	0100514	FT MORGAN CANAL	Diversion	9.0	3.0	Instream1
10	Instream1	Instream Flow 1	InstreamFlow	10.0	3.0	06758500
11	06758500	WELDONA GAGE	StreamGage	11.0	3.0	0102900
12	0102900	WELDON VALLEY RETURN	Return	13.0	3.0	0100517
13	0100517	DEUEL AND SNYDER	Diversion	15.0	4.0	0100515
14	0100515	UPPER PLATTE BEAVER CNL	Diversion	17.0	4.0	06759500
15	06759500	FORT MORGAN GAGE	StreamGage	19.0	4.0	0100518
16	0100518	LOWER PLATTE BEAVER D	Diversion	26.0	4.0	0100519
17	0100519	TREMONT DITCH	Diversion	37.0	5.0	0100687
18	0100687	NORTH STERLING CANAL	Diversion	38.0	5.0	0100688
19	0100688	UNION DITCH	Diversion	49.0	5.0	06760000
20	06760000	BALZAC GAGE	StreamGage	50.0	5.0	
						AnalyzeNetworkPointFLow

AnalyzeNetworkPointFlow() Network Input Table

In this example the network is defined in an Excel file, the ReadTableFromExcel() command is used to read the table, and the table is used as input to the AnalyzeNetworkPointFlow() command The network definition table columns from the above figure are as follows (note, however, that the column names are user defined and are specified as parameters to the AnalyzeNetworkPointFlow() command:

Network Table	Description
Column	
NodeID	The location ID for the network node, typically corresponding to the
	location ID in time series identifiers. The column is indicated to the
	command using the NodeIDColumn parameter.
NodeName	The node name, useful because NodeID is generally terse and non-
	descriptive, used in messages. The column is indicated to the command
	using the NodeNameColumn parameter.
NodeType	The node type, needed to define node behavior (e.g., whether time series
	values get added, subtracted, reset at node). The node types are user-
	defined, although types often are defined by modeling conventions. The
	column is indicated to the command using the NodeTypeColumn
	parameter. The behavior corresponding to node types is defined by using
	command parameters (NodeAddTypes, NodeSubtractTypes,
	NodeOutflowTypes, NodeFlowThroughTypes).
NodeDist	The node distance along the flow path. Typically the distance is measured
	relative to the lowest point on the network. The distance is used to estimate
	gain/loss when GainMethod=Distance is specified as a command

AnalyzeNetworkPointFlow() Network Input Table Column Description

Network Table Column	Description
	parameter.
NodeWeight	Used when GainMethod=Weight. The weights indicate the relative weight of the reach gain/loss to be distributed between nodes on the reach. For example, specify a best estimate of the percentage of reach loss that occurs above each node. Or, specify as a rate of gain/loss when used with GainMethod=DistanceWeight (but in this case the distance*weight product will be normalized to ensure that the reach gain/loss is equalized between known point flows).
DownstreamNodeID	The location ID for the downstream node, needed to define network connectivity.

The AnalyzeNetworkPointFlow() command creates output time series with the data types indicated in the following table.

Column	Description
NodeInflow	Sum of outflows from upstream nodes, which are consequently inflows
	to the current node (lagged routing currently is not implemented).
NodeAdd	Time series added at the node (for example immediately off-channel
	reservoir release or measured return flow).
NodeSubtract	Time series subtracted at the node (for example diversion).
NodeUpstreamGain	Gain (positive) or loss (negative) between immediate upstream node(s)
	and the current node (missing if gain/loss is not computed).
NodeOutflow	Outflow from the node, which takes into account inflow and any
	additions and subtractions at the node.
NodeUpstreamReachGain	Gain (positive) or loss (negative) between upstream known flow
	node(s) and the current node (missing if gain/loss is not computed).
NodeInflowWithGain	NodeInflow + NodeUpstreamReachGain (missing if gain/loss
	are not computed).
NodeOutflowWithGain	NodeOutflow + NodeUpstreamReachGain (missing if gain/loss
	are not computed).
NodeStorage	Storage at the node after additions and subtractions (currently always
	zero, will enhance in the future to handle on-channel reservoirs).

AnalyzeNetworkPointFlow() Network Input Table Column Description

The following figure illustrates the output time series corresponding to the data types listed in the above table:

ATE	0103816, Nodeinflow, CFS	0103816, NodeAdd, CFS	0103816, Node Subtract, CFS	0103816, NodeUpstreamGain, CFS	0103816, NodeOutflow, CFS	0103816, NodeUpstreamReachGain, CFS	0103816, NodeInflowWithGain, CFS	0103816, NodeOutflowWithGain, CFS	0103816, Node Storage, CFS
975-05-24	806.00	0.00	258.00	-30.67	548.00	-30.67	775.33	517.33	0.0
75-05-25	455.00	0.00	232.00	2.44	223.00	2.44	457.44	225.44	0.0
975-05-26	291.00	0.00	205.00	11.22	86.00	11.22	302.22	97.22	0.0
75-05-27	208.00	0.00	193.00	17.33	15.00	17.33	225.33	32.33	0.0
975-05-28	222.00	0.00	181.00	15.11	41.00	15.11	237.11	56.11	0.0
75-05-29	2120.00	0.00	286.00	-165.00	1834.00	-165.00	1955.00	1669.00	0.0
75-05-30	3920.00	0.00	301.00	-229.89	3619.00	-229.89	3690.11	3389.11	0.0
975-05-31	2010.00	0.00	286.00	75.11	1724.00	75.11	2085.11	1799.11	0.
75-06-01	1710.00	0.00	286.00	4.33	1424.00	4.33	1714.33	1428.33	0.
75-06-02	1480.00	0.00	284.00	-17.44	1196.00	-17.44	1462.56	1178.56	0.
75-06-03	1340.00	0.00	290.00	-26.11	1050.00	-26.11	1313.89	1023.89	0.
75-06-04	1370.00	0.00	281.00	-30.22	1089.00	-30.22	1339.78	1058.78	0.
75-06-05	1070.00	0.00	111.00	-15.78	959.00	-15.78	1054.22	943.22	0.
75-06-06	943.00	0.00	91.00	-5.22	852.00	-5.22	937.78	846.78	0.
75-06-07	924.00	0.00	91.00	-20.11	833.00	-20.11	903.89	812.89	0.
75-06-08	1120.00	0.00	91.00	-52.11	1029.00	-52.11	1067.89	976.89	0.
75-06-09	2200.00	0.00	91.00	-157.00	2109.00	-157.00	2043.00	1952.00	0.
975-06-10	2310.00	0.00	99.00	-98.00	2211.00	-98.00	2212.00	2113.00	0.
75-06-11	3750.00	0.00	107.00	-200.78	3643.00	-200.78	3549.22	3442.22	0.
75-06-12	2990.00	0.00	111.00	5.22	2879.00	5.22	2995.22	2884.22	0.
975-06-13	2570.00	0.00	97.00	7.89	2473.00	7.89	2577.89	2480.89	0.
975-06-14	2240.00	0.00	42.00	19.44	2198.00	19.44	2259.44	2217.44	0.

AnalyzeNetworkPointFlow() Output Time Series Table

The following logic is used to analyze the network. Currently this logic is performed by navigating the network from most upstream to downstream and processing all timesteps for a node before moving to the next node.

- 1. The network is navigated from top to bottom. When a confluence is found (a node with more than one upstream node), each confluence is processed from the top down to the confluence point. Of particular importance is the concept of a "stream reach", which is the reach between known flow points, because mass balance is enforced at known flow points and gain/loss can be estimated between the known flow points.
 - a. The data type for the node (see *DataType command parameters) is used to retrieve the relevant time series for the node. The first time series that matches the location ID, data type, and interval is used as input for the node. The time series must have been read prior to the AnalyzeNetworkPointFlow() command. For example, use the CopyTable() command to copy a subset of the network table's NodeID values and then use the ReadTimeSeriesList() command with the list of identifiers.
 - b. Calculate the node's inflow:
 - i. Node types that set outflow, indicated by the NodeOutflowDataTypes parameter (e.g., StreamGage):
 - NodeInflow = input time series for node
 - ii. All other node types:
 - NodeInflow = sum of upstream node outflows
 - c. Calculate the node's outflow:
 - i. Node types that add, indicated by the NodeAddDataTypes parameter (e.g., Return, Import):
 - NodeOutflow = NodeInflow + added time series
 - ii. Node types that subtract, indicated by the NodeSubtractDataTypes parameter (e.g., Diversion):
 - NodeOutflow = NodeInflow subtracted time series

- iii. Node types that set outflow, indicated by the NodeOutflowDataTypes parameter (e.g., StreamGage):
 - NodeOutflow = NodeInflow
- iv. Node types that let flow through, indicated by the
 - NodeFlowThroughDataTypes parameter (e.g., InstreamFlow):
 - NodeOutflow = NodeInflow.
- d. For known flow points (e.g., StreamGage node type), set the reach gain/loss:
 - i. NodeUpstreamReachGain = difference between upstream node outflow and known flow at downstream node in reach
- e. If gain/loss is being estimated and a known flow node encountered (e.g., StreamGage), gain/loss between this node and the nearest upstream node(s) is compute. This has only been implemented for the case where all intervening nodes are in a non-branching reach.
 - i. First calculate the distribution factor by which the reach gain (see previous step) will be distributed to each node in the reach:
 - If the GainMethod=None, no adjustment to flows is made and the gain/loss upstream of the know flow node will result in a discontinuous jump because no gain/loss adjustment is made.
 - If the GainMethod=Distance, use the node distance data from the network table to prorate the gain/loss in the stream reach. The difference in distance between the upstream node and the current node is set to weight for prorating the reach gain/loss. Use this method if the gain/loss rate is the same throughout the reach and therefore only the distance between nodes controls the gain/loss.
 - If the GainMethod=Weight, the gain/loss is prorated by the weights specified by the NodeWeightColumn parameter (or weight equally if the weights are not specified in the network table). The weight of the upstream known flow node is not used. Use this method if the relative gain/loss for each node within the reach can be specified.
 - If the GainMethod=DistanceWeight, the gain/loss is prorated by the product of the weights specified by the NodeWeightColumn parameter (or weight equally if the weights are not specified in the network table) and by the values from the NodeDistanceColumn. The weight of the upstream known flow node is not used. Use this method if the relative rate of gain/loss for each node can be specified, but overall gain/loss is also a function of the distance. Even though a rate is specified, the calculated gain/loss may be slightly different because the overall reach gain/loss must be balanced at known flow points for each time step.
 - Multiply NodeUpstreamReachGain by the gain/loss distribution factor to calculate NodeReadGain for each node.
 - Compute the cumulative gain/loss for the node by summing NodeUpstreamNodeGain for each upstream node and set to NodeUpstreamReachGain for the current node.
- 2. Analysis statistics optionally are written to an output table, which contains a row for each network node. Statistics include information such as the number of missing values in the input time series. This information can be used to evaluate the quality of the analysis. **This feature has not yet been implemented.**

Issues that need to be considered include:

- 1. Missing data in input result in missing data in calculated values. Use TSTool features to fill missing data in time series before using as input to the analysis. Because this may be a major effort, especially for a long analysis period, it may be appropriate to read time series from model data sets. It is envisioned that the output table will provide feedback on how much missing data there is and how it impacts the analysis.
- 2. TSTool's graphing tool currently does not allow graphing lines as a step function in the case where no gain/loss is computed. Instead, the line connects the data points. An enhancement to the graphing tool is needed.
- 3. TSTool does not provide a way to graph a stream reach where the graph values are pulled from each time series for a point in time. Ideally a visualization tool would allow "scrolling" through dates and showing the river reach with flow on the Y axis and node distance on the X axis, although it would be tedious to have to scroll through the period.
- 4. There may be cases where a subtraction at the node takes all of the flow resulting in a zero or negative value, essentially causing the node to be a known zero point flow. For example, in Colorado, a river call may result in a river drying up during the call. It is possible to estimate when this occurs, but the data quality may be low. Currently TSTool allows negative flows in this case, which indicates that input time series or the simple gain method calculations do not accurately represent the system. One option in this case is to use the TSTool AdjustExtremes() command, which maintains mass balance around the extreme values.

It is important to understand that such a point flow analysis represents a snapshot of the system at any point in time, but does not route flows through the network. Known flows at stream gages are used as fixed values from which other data are estimated. Gains and losses are representative of the network system, essentially interpolating over time and distance. This type of analysis introduces errors in cases where the lag time between nodes would result in significant differences if lagging were considered. In the physical system, changing an upstream flow would result in lagged impacts due to routing; however, the point flow analysis shows the impacts to downstream nodes in the same time step. A more sophisticated model with routing would be needed to represent actual conditions. However, the point flow analysis will be reasonably accurate if gains and losses are occurring because of fairly static phenomena (e.g., groundwater interactions that do not change rapidly within the network travel time). One way to work around these limitations is to use a longer interval, for example monthly instead of daily, in input time series or convert the point flow analysis results.

The following dialog is used to edit the command and illustrates the syntax of the command:

Edit AnalyzeNetw	orkPointFlo	w() Command				23
This command anal It is assumed that r Consequently, the	yzes a flow r math operati analysis prov	etwork to comp ons can occur in vides an approxi	ute poi the sa	nt flows at me timeste of system	all nodes in the network. p (no routing). behavior at a point in time.	
-Specify how to m	ap table colu	mns to the netw	ork-			-
	Table ID:	Network1			 Required - table containing network node infor 	mation.
Node	ID column:	NodeID			Required - column name for node IDs.	
Node na	ame column:	NodeName			Optional - column name for node names.	
Node t	ype column:	NodeType			Required - column name for node types.	
Node dista	nce column:	NodeDist			Optional - used if GainMethod requires distance	e.
Node wei	ght column:	NodeWeight			Optional - used if GainMethod requires weight.	
Downstream node	ID column:	DownstreamNo	deID		Required - column name for downstream node	IDs.
-Specify node type	e behavior fo	r the point flow	analysi	s		
a hunder	Node	types that add:	Return	1	Optional - node types that add.	
Node time serie	es data type:	that add flow:	DivTo	tal	Optional - node time series data types that add.	
No	de types tha	t subtract flow:	Diversion		Optional - node types that subtract.	
Node time serie	es data type	that subtract: DivTotal		tal	Optional - node time series data types that subt	types that subtract.
N	lode types th	nat set outflow:	Stream	nGage	Optional - node types that set outflow.	
Node time series of	data types th	nat set outflow:	Stream	nflow	Optional - node time series data types that set of	outflow.
1	Node types v	with no change:	Instre	amFlow	Optional - node types where inflow=outflow.	
Data interval:	ay 👻			Require	d - data interval (time step) for time series.	
Analysis start: 1	950-01-01			Optiona	I - analysis start date/time (default=full time serie	s period
Analysis end: 2	012-12-31			Optiona	I - analysis end date/time (default=full time series	period)
Data units: C	FS			Optiona	I - units for output time series (default=no units).	
Gain method:)istance 👻			Optiona	I - how to compute gains (default=None).	
Output table ID: R	esults			Optiona	I - identifier for output summary table.	
Command: ^E E	',NodeOu streamFl Ind="201 eID="Re	tflowDatal ow",Interv 2-12-31",U sults")	Types val=D Jnits	="Stre ay,Ana ="CFS"	amflow",NodeFlowThroughTypes=" lysisStart="1950-01-01",Analys ,GainMethod="Distance",OutputT	'In sis Tab

AnalyzeNetworkPointFlow() Command Editor

AnalyzeNetworkPointFLow

The command syntax is as follows:

AnalyzeNetworkPointFlow(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TableID	The identifier for the table defining the	None – must be specified.
	network.	
NodeIDColumn	The name of the column in the network	None – must be specified.
	table containing node identifiers. Node	
	identifiers will be used for the location	
	ID part of time series identifiers.	
NodeNameColumn	The name of the column in the network	
	table containing node names.	
NodeTypeColumn	The name of the column in the network	None – must be specified.
	table containing node types. The node	
	type is used to specify what	
	calculations will occur for the node.	
NodeDistanceColumn	The name of the column in the network	Must be specified when
	table containing node distance. The	GainMethod=
	distance is the measure from the most	Distance or
	downstream node and is used when	GainMethod=
	GainMethod=Distance or	DistanceWeight.
	GainMethod=	
	DistanceWeight.	
NodeWeightColumn	The name of the column in the network	If not specified when
	table containing node weights, which is	GainMethod=Weight,
	used to distribute gain/loss when	gain/loss will be
	GainMethod=Weight or	distributed evenly for the
	GainMethod=	nodes. Must be specified
	DistanceWeight (in the latter case	when GainMethod=
	the weight is the rate to use).	DistanceWeight.
DownstreamNodelDColumn	The name of the column in the network	None – must be specified.
	table containing downstream node	
	identifiers. This information defines	
	the connectivity of the network.	NT 111 111
NodeAddTypes	Node types for which time series are	No additions will occur.
	added to the node's inflow to compute	
	outflow, for example the Return	
	node type in the above table example.	
	The Node Type Column table column	
	is checked to determine the type for	
NodoAddDatamma	The time series date time to match for	No additiona will accur
ModeAddDataType	the node. The data type to match for	The additions will occur.
	the Node TD as the location ID to	
	metab available time series to use a	
	input. This may be appeared to allow	
	TSID nottom like % Distribute like	
	a ISID pattern like &L-DIVTOTAL, to	

Parameter	Description	Default
	allow more flexibility in matching time series.	
NodeSubtractTypes	Node types for which time series are subtracted from the node's inflow, for example the Diversion node type in the above table example. The NodeTypeColumn table column is checked to determine the type for each node in the network.	No subtractions will occur.
NodeSubtractDataType	The time series data type to match for the node. The data type is used with the NodeID as the location ID to match available time series to use as input. This may be enhanced to allow to a TSID pattern like %L-DivTotal, to allow more flexibility in matching time series.	No subtractions will occur.
NodeOutflowTypes	Node types for which time series outflows are set to the node's time input time series, for example the Streamflow node type in the above table example. The NodeTypeColumn table column is checked to determine the type for each node in the network.	No known flows will be set – gain/loss cannot be computed.
NodeOutflowDataType	The time series data type to match for the node. The data type is used with the NodeID as the location ID to match available time series to use as input. This may be enhanced to allow a TSID pattern like %L- Streamflow, to allow more flexibility in matching time series.	No subtractions will occur.
NodeFlowThroughTypes	Node types for which time series outflows are set to the node's inflow, for example the InstreamFlow node type in the above table example. The NodeTypeColumn table column is checked to determine the type for each node in the network.	No known flows will be set – gain/loss cannot be computed.
Interval	The time series interval to process. The interval is used with the node identifier and data type to match input time series.	None – must be specified.
AnalysisStart	The analysis start, which defines the period for output time series. Specify to a precision consistent with Specify to a precision consistent with Interval.	Global output period.

Parameter	Description	Default
AnalysisEnd	The analysis end, which defines the period for output time series. Specify to a precision consistent with Interval.	Global output period.
Units	Units for output time series. Warnings will be generated if input time series for the analysis are not consistent with these units.	
GainMethod	 The method used to prorate the gain/loss between known point flow nodes to other nodes in the reach. Currently this can be used only on nonbranching networks. Distance – prorate the gain/loss using distance between nodes (as a portion of the total distance). Use this method if a constant gain/loss rate applies over each reach in the network. None – no gain/loss is estimated, resulting in a discontinuity in an outflow jump above each known point flow. DistanceWeight – prorate the gain/loss using distance*weight as the weight for each node, where the rate is specified in the weight network table column. Use this method when the gain/loss rate varies by location and should be represented as a rate. Weight – prorate the gain/loss using the weights specified for each node. Use this method if the gain/loss fraction in a reach is overlicitly specified 	None
OutputTable	The identifier for the output table to receive analysis results statistics	No output table will be created

The following command files illustrate how to implement a point flow analysis. In this case the first command file prepares daily time series using the network as input. The time series could similarly be provided by other processing procedures, or read from other model input files.

```
# Read time series needed to perform the AnalyzeNetworkPointFlow() tests.
# Use data from HydroBase to provide realistic input.
# First read the network table
ReadTableFromExcel(TableID="Network1",InputFile="Network1.xlsx",ExcelColumnNames=FirstRowInRange)
# Get the list of streamflow gages and associated time series
# Free()
CopyTable(TableID="Network1",NewTableID="StreamflowStationList",IncludeColumns="NodeID",
```

ColumnMap="NodeID:StreamGageID",ColumnFilters="NodeType:StreamGage") ReadTimeSeriesList(TableID="StreamflowStationList",LocationColumn="StreamGageID",DataSource="DWR,USGS", DataType="Streamflow", Interval="Day", DataStore="HydroBase", IfNotFound=Warn) WriteDateValue(OutputFile="Network1-StreamGage-Streamflow.dv",MissingValue=NaN,TSList=AllMatchingTSID, TSID="*.*.Streamflow.Day.*") # Get the list of diversion stations and associated time series Free() CopyTable(TableID="Network1", NewTableID="DiversionStationList", IncludeColumns="NodeID", ColumnMap="NodeID:DiversionID", ColumnFilters="NodeType:Diversion") ReadTimeSeriesList(TableID="DiversionStationList",LocationColumn="DiversionID",DataSource="DWR", DataType="DivTotal", Interval="Day", DataStore="HydroBase", IfNotFound=Warn) WriteDateValue(OutputFile="Network1-Diversion-DivTotal.dv",MissingValue=NaN,TSList=AllMatchingTSID, TSID="*.*.DivTotal.Day.*") # Get the list of diversion return stations and associated time series Free() CopyTable(TableID="Network1", NewTableID="DiversionReturnStationList", IncludeColumns="NodeID", ColumnMap="NodeID:DiversionID", ColumnFilters="NodeType:Return") ReadTimeSeriesList(TableID="DiversionReturnStationList",LocationColumn="DiversionID",DataSource="DWR", DataType="DivTotal", Interval="Day", DataStore="HydroBase", IfNotFound=Warn) WriteDateValue(OutputFile="Network1-Return-DivTotal.dv", MissingValue=NaN, TSList=AllMatchingTSID, TSID="*.*.DivTotal.Day.*")

The second command file performs the point flow analysis. This example is from a TSTool test and fills missing data with a simple approach in order to ensure that no missing values are included in the analysis. A single command file that combines the two command file examples also could be used.

```
# Test analyzing a simple network for point flows
StartLog(LogFile="Results/Test_AnalyzeNetworkPointFlow.TSTool.log")
# Read the network
ReadTableFromExcel(TableID="Network1",InputFile="Data\Network1.xlsx",Worksheet="Network1",
 ExcelColumnNames=FirstRowInRange)
# Read the time series associated with network nodes (pregenerated)
# Fill diversion time series with zeros so there is something to analyze
# Fill stream gage time series with repeat forward and backward
SetInputPeriod(InputStart="1950-01-01",InputEnd="2013-12-31")
ReadDateValue(InputFile="Data\Network1-Diversion-DivTotal.dv")
ReadDateValue(InputFile="Data\Network1-Return-DivTotal.dv")
FillConstant(TSList=AllMatchingTSID,TSID="*.*.DivTotal.*.*",ConstantValue=0)
ReadDateValue(InputFile="Data\Network1-StreamGage-Streamflow.dv")
FillRepeat(TSList=AllMatchingTSID,TSID="*.*.Streamflow.*.*",FillDirection=Backward)
FillRepeat(TSList=AllMatchingTSID,TSID="*.*.Streamflow.*.*",FillDirection=Forward)
CheckTimeSeries(CheckCriteria="Missing")
# Analyze the network point flow.
AnalyzeNetworkPointFlow(TableID="Network1", NodeIDColumn="NodeID", NodeNameColumn="NodeName",
 NodeTypeColumn="NodeType", NodeDistanceColumn="NodeDist", NodeWeightColumn="NodeWeight",
  DownstreamNodeIDColumn="DownstreamNodeID", NodeAddTypes="Return", NodeAddDataTypes="DivTotal",
 NodeSubtractTypes="Diversion", NodeSubtractDataTypes="DivTotal", NodeOutflowTypes="StreamGage",
 NodeOutflowDataTypes="Streamflow", NodeFlowThroughTypes="InstreamFlow", Interval=Day,
  AnalysisStart="1950-01-01", AnalysisEnd="2012-12-31", Units="CFS", GainMethod="Distance",
  OutputTableID="Results")
```

Command Reference: AppendFile()

Append 1+ files to another file

Version 10.12.00, 2012-10-12

The AppendFile() command appends one or more files to another file. All or only matching lines from input files can be transferred. This command is useful for appending multiple data files into a single file that can be read by TSTool.

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

👌 Edit Appe	ndFile() command	2		
Append the cont	tents of one or more files to anoth	ner file.		
The input file can	n be a single file, all files in a folde	r (*), or all files matching an extensio (*.csv).		
Use the IncludeT	Text parameter to append only lin	es that match a pattern using the Java regular expressions.		
It is recommende	ed that the file name be relative to	o the working directory, which is:		
C:\Develop\TS	5Tool_SourceBuild\TSTool\test\reg	gression\commands\general\AppendFile		
Input file(s): D	Data/*.csv	Browse		
Output file(s):	Results/Test_AppendFile_Extension_IncludeText_out.csv Browse			
Include text:	*\Q-\E.*	Optional - include lines matching regular expression (default=include all)		
If not found?:	✓	Optional - action if input file is not found (default=Warn)		
Command: X	AppendFile(InputFile: xtension_IncludeText_	="Data/*.csv",OutputFile="Results/Test_AppendFile_E _out.csv",IncludeText=".*\Q-\E.*")		
	Add Working Directory (Inpul	t) Add Working Directory (Output) Cancel OK		

AppendFile() Command Editor

The command syntax is as follows:

```
AppendFile(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
InputFile	The name of one or more files to delete, using the following conventions:	None – must be specified.
	• No * in name – match one file.	
	• Filename of *- match all files in input	
	directory (working directory by default).	
	• Filename of *.ext – match all files with	
	extension	
	More options may be supported in the future when TSTool is updated to use Java 1.7+.	
OutputFile	The output file that will be appended to. The file	None – must be specified.
	is created if it does not exist. Use the	
	RemoveFile() command to remove the old	
	file.	
IncludeText	A regular expression pattern to include text. This	Transfer all lines.
	uses the Java regular expressions syntax (see	
TfNotFound	<u>http://en.wikipedia.org/wiki/Regular_expression).</u>	Waxa
IINOCFOUND	indicate action if the file is not found, one of:	Walli
	• Ignore – Ignore the missing file (do not warn).	
	• Warn – generate a warning (use this if the	
	file truly is expected and a missing file is a	
	cause for concern).	
	• Fail – generate a failure (use this if the file	
	truly is expected and a missing file is a cause	
	tor concern).	

The following table lists regular expression examples:

InputText Regular Expression	Description
.*\Q-\E.*	Match lines that start with any character, end with any character, and contain a dash.
	The \Q and \E characters are special characters to start and end a quoted character,
	and are necessary because the dash has special meaning in a regular expression.

Command Reference: AppendTable() Append one table to another table

Version 10.21.00, 2013-06-28

The AppendTable() command appends rows from one table to another table. For appended rows:

- values in columns that are not matched are set to null in the receiving table
- values in columns where the data types do not match are set to null in the receiving table

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

Edit AppendTable() Command				
This command appends records from one table Columns in the second table must be mapped	e to the bottom of another table. to columns in the first table and match in type.			
Table ID to modify:	Table1	Required - original table.		
Table ID to append from ("append table"):	Table2	Required - table to append from.		
Column names to append from append table:	String2	Optional - names of columns to append (default=append all matching columns).		
Column map:	String2:String	Optional - to change append table names (default=no changes).		
Column filters:	String2:F*	Optional - filter appended rows by matching column pattern (default≕append all rows).		
Command:	<pre>AppendTable(TableID="Table1", AppendTableID="Table2", IncludeColumns="String2", ColumnMap ="String2:String", ColumnFilters="String2:F*")</pre>			
Cancel OK				

AppendTable() Command Editor

The command syntax is as follows:

AppendTable(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TableID	The identifier for the original table, to which records	None – must be
	will be appended.	specified.
AppendTableID	The identifier for the table from which to append.	None – must be
		specified.
IncludeColumns	The names of columns to append from	Append all of the
	AppendTableID, separated by commas. See also	columns from
	ColumnMap to indicate how to map column names in	AppendTableID
	the append table to the first table (necessary if the	that match columns in
	column names don't match).	TableID.
ColumnMap	The map of the append table columns to the first	If no map, append
	table's columns, necessary when column names are not	table column names in

Parameter	Description	Default
	the same:	IncludeColumns
	AppendColumn1:OriginalColumn1,	must have the same
	AppendColumn2:OriginalColumn2	name in the first table.
ColumnFilters	Filters that limit the number of rows being appended	No filtering.
	from the append table, using the syntax:	
	FilterColumn1:FilterPattern1,	
	FilterColumn2:FilterPattern2	
	Patterns can use * to indicate wildcards for matches.	
	Only string values can be checked (other data types are	
	converted to strings for comparison). Comparisons are	
	case-independent. All patterns must be matched in	
	order to append the row. In the future a command may	
	be added to perform queries on tables, similar to SQL	
	for databases.	

The following figures show the input tables and results (modified first table) corresponding to the parameters shown in the editor dialog figure above. Note that the column names for "Table2" have a "2".

🕚 TSTool - Table "Table1" 🛛 🗖 🔀				
DateTime	String	Double	Integer	
2000-01-01	First day	1.0		1
2000-01-02	Second day	2.0		2
2000-01-03	Third day	3.0		3
2000-01-04	Fourth day	4.0		4
2000-01-05	Fifth day	5.0		5
2000-01-06	Sixth day	6.0		6
Displaying 6 rows, 4 columns. Ready				

💧 TSTool - Table "Table2" 🛛 🔲 🔀				
DateTime2	String2	Double2	Integer2	
2000-01-01	First day	1.0	1	
2000-01-02	Second day	2.0	2	
2000-01-03	Third day	3.0	3	
2000-01-04	Fourth day	4.0	4	
2000-01-05	Fifth day	5.0	5	
2000-01-06	Sixth day	6.0	6	
Displaying 6 rows, 4 columns. Ready				

AppendTable_Table2

AppendTable_Table1 Table Corresponding to TableID in Command Editor



👌 TSTool - 1	Table "Table1"			
DateTime	String	Double	Integer	
2000-01-01	First day	1.0	1	
2000-01-02	Second day	2.0	2	
2000-01-03	Third day	3.0	3	
2000-01-04	Fourth day	4.0	4	
2000-01-05	Fifth day	5.0	5	
2000-01-06	Sixth day	6.0	6	
	First day			
	Fourth day			
	Fifth day			
		·	,	
Displaying 9 row	vs, 4 columns.		Ready	/

AppendTable_Table1

Table Corresponding to Results from Parameters in Command Editor

Command Reference: ARMA()

Lag and attenuate a time series using AutoRegressive Moving Average

Version 10.13.00, 2012-10-25

The ARMA() command lags and attenuates a time series (e.g., to route a streamflow time series downstream). This approach preserves the "mass" of the data. The general equation for ARMA is:

 $O_t = a_1 * O_{t-1} + a_2 * O_{t-2} + \dots + a_p * O_{t-p} + b_0 * I_t + b_1 * I_{t-1} + \dots + b_q * I_{t-q}$

Where:

t = time step

 O_t = output value at time t

 I_t = input value at time t

a, b = ARMA coefficients

and the *p* and *q* values indicate the degree of the equation: ARMA(p,q).

The ARMA coefficients are determined by analyzing historical data and may be developed using a data interval that is different than the data interval of the time series that is being manipulated. The coefficients are typically computed by an external analysis program (TSTool does not perform this function).

The time series to process can have any interval. The *a* and *b* coefficients are listed in the dialog from left-most to right-most in the equation. Note that there are *p a*-coefficients and (q + 1) *b*-coefficients (because there is a *b*-coefficient at time t_0). The interval used to compute the ARMA coefficients can be different from the data interval but the data and ARMA intervals must be divisible by a common interval. The ARMA algorithm is executed as follows:

- 1. The data and ARMA intervals are checked and if they not the same, the data are expanded by duplicating each value into a temporary array. For example, if the data interval is 6Hour and the ARMA interval is 2Hour, each data value is expanded to three data values (2Hour values). If the data interval is 6Hour and the ARMA interval is 10Hour, each data value is expanded to three data value is expanded to three data values (2Hour values).
- 2. The ARMA equation is applied at each point in the expanded data array. However, because the ARMA coefficients were developed using a specific interval, only the data values at the ARMA interval are used in the equation. For example, if the expanded data array has 2Hour data and the ARMA interval is 10Hour, then every fifth value will be used (e.g., *t* corresponds to the "current" value and t I corresponds to the fifth value before the current value). Because the ARMA algorithm depends on a number of previous terms in both the input and output, there will be missing terms at the beginning of the data array and in cases where missing data periods are encountered. Ideally ARMA will be applied to filled data and only the initial conditions will be an issue. In this case the output period should ideally be less than the total period so that the initial part of the routed time series can be ignored. In cases where *O* values are missing, the algorithm first tries to use the *I* values. If any values needed for the result are missing, the result is set to missing.
- 3. The final results are converted to a data interval that matches the original input, if necessary. If the original data interval and the ARMA interval are the same, no conversion is necessary. For example, if the original data interval is 6Hour and the ARMA interval is 10Hour, then the expanded data

interval will be 2Hour. Consequently, three sequential expanded values are averaged to obtain the final 6Hour time series.

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

🛃 Edit ARMA() Command	<u>×</u>		
Lag and attenuate a time series using the ARMA (AutoRegressive Moving Average) method.			
The adjusted output time series O is o	omputed from the original input I using:		
$O[t] = a_1*O[t-1] + a_2*O[t-2] + + a_1$	_p*O[t-p] + b_0*l[t] + b_1*l[t-1] + + b_q*l[t-q]		
where t = time, p = number of outflow	s to consider, and q = number of inflows to consider		
ARMA a and b coefficients must be c	omputed externally and should sum to 1.0.		
The values for p and q will be determine	ned from the number of coefficients.		
Specify the interval used to compute a	ARMA coefficients as 1Day, 6Hour, 2Hour, etc.		
The ARMA interval must be <= the tim	e series interval.		
The resulting value is set to missing if	one or more input values are missing		
(typically only filled data should be us	ed). The period will not automatically be extended.		
TS list:	AllMatchingTSID 🔽 Indicates the time series to process (default=AlITS		
TSID (for TSList=AllMatchingTSID):	Route		
EnsembleID (for TSList=EnsembleID):			
"a" coefficients:	0.7325,-0.3613,0.1345,0.5221,-0.2500,0.1381,-0.2643,0.0558		
"b" coefficients:	0.0263,0.0116,-0.0146,-0.0081,0.0127,0.0798,0.0727,0.0523,0.0599		
ARMA interval:	2Hour Required (e.g., 2Hour, 15Minute).		
Command:	ARMA(TSList=AllMatchingTSID,TSID="Route",ARMAInterval=2Ho ur,a="0.7325,-0.3613,0.1345,0.5221,-0.2500,0.1381,-0.2643 ,0.0558",b="0.0263,0.0116,-0.0146,-0.0081,0.0127,0.0798,0 .0727,0.0523,0.0599")		
	Cancel OK		

ARMA() Command Editor

The command syntax is as follows:

ARMA(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of:	Allts
	• AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will be modified.	
	• AllTS – all time series before the command.	
	• EnsembleID – all time series in the ensemble will be modified.	
	• FirstMatchingTSID – the first time series that matches the	

Parameter	Description	Default
	TSID (single TSID or TSID with wildcards) will be modified.	
	• LastMatchingTSID – the last time series that matches the	
	TSID (single TSID or TSID with wildcards) will be modified.	
	• SelectedTS – the time series are those selected with the	
	SelectTimeSeries() command.	
TSID	The time series identifier or alias for the time series to be modified,	Required if
	using the * wildcard character to match multiple time series.	TSList=*TSID.
EnsembleID	The ensemble to be modified, if processing an ensemble.	Required if
		TSList=
		EnsembleID.
ARMA	The ARMA interval to use in the analysis	None – must be
Interval		specified.
a	a coefficients.	Optional.
b	b coefficients.	None – must be
		specified.

A sample command file to process streamflow data from the USGS is as follows:

```
SetOutputPeriod(OutputStart="1936-01-01",OutputEnd="1936-03-31")
ReadUsgsNwisRdb(InputFile="Data/G03596000.rdb",Alias=Original)
Copy(TSID="Original",NewTSID="03596000.USGS.Streamflow.Day.Routed",Alias=Routed)
ARMA(TSList=AllMatchingTSID,TSID="Routed",ARMAInterval=2Hour,a="0.7325,
-0.3613,0.1345,0.5221,-0.2500,0.1381,-0.2643,0.0558",b="0.0263,0.0116,
-0.0146,-0.0081,0.0127,0.0798,0.0727,0.0523,0.0599")
```



The following figure shows the original and routed time series.

Example Graph Showing Original and ARMA-Routed Time Series

The Cumulate() command can be used to verify mass balance of the original and routed time series (see the Cumulate() command discussion below). For example, insert a Cumulate() command near the end of a command file.



The following figure shows the time series from the previous graph, this time as cumulative time series.

Example Graph Showing Original and ARMA-Routed Time Series as Cumulative Values

This page is intentionally blank.
Command Reference: Blend()

Append a Time Series to the End of Another Time Series

Version 08.15.00, 2008-05-01

The Blend() command blends one time series into another, extending the first time series period if necessary. This is typically used for combining time series for a station that has been renamed or to blend historic and real-time data. The second (independent time series) will ALWAYS override the first time series. See also the SetFromTS() and Add() commands. The Blend() command ensures that single data values are used whereas Add() will add values if more than one value is available at the same date/time. The SetFromTS() does not extend the period.

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

💊 Edit Blend() commar	nd 🗙						
Blend one time series into the	Blend one time series into the start or end of another.						
The BlendAtEnd blend meth	nod will use data from the second (independent) time series at the end of the first.						
The overall period will be th	hat of both time series.						
Additional blend methods (e	e.g., interpolating the blend) may be added in the future.						
See also the SetFromTS() a	and Add() commands.						
Time series to modify:	obatos_current						
Independent time series: lo	bbatos_likely						
Blend method: B	3lendAtEnd						
B	lend(TSID="lobatos_current",IndependentTSID="lobatos_likely",Ble dMethod=BlendAtEnd)						
Command:							
	Cancel OK						

Blend() Command Editor

The command syntax is as follows:

Blend(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSID	The time series identifier or alias for the	None – must be specified.
	time series to be modified.	
IndependentTSID	The time series identifier or alias for the	None – must be specified.
	time series to be blended to the first time	
	series.	
BlendMethod	The method used to blend the data, one	None – must be specified.
	of:	Currently only BlendAtEnd is
	• BlendAtEnd, resulting in the main	recognized.
	time series having the other time	
	series attached to the end of its	
	period.	

A sample command file to blend two time series from the State of Colorado's HydroBase database is as follows:

```
# 08236000 - ALAMOSA RIVER ABOVE TERRACE RESERVOIR
08236000.DWR.Streamflow.Month~HydroBase
# 08236500 - ALAMOSA RIVER BELOW TERRACE RESERVOIR
08236500.DWR.Streamflow.Month~HydroBase
Blend(TSID="08236000.DWR.Streamflow.Month",
IndependentTSID="08236500.DWR.Streamflow.Month",
BlendMethod="BlendAtEnd")
```

Command Reference: CalculateTimeSeriesStatistic()

Calculate time series statistic

Version 10.18.00, 2013-02-21

The CalculateTimeSeriesStatistic() command calculates a statistic for a time series (typically a single value, but may have multiple output values) and optionally adds the result to a table. Multiple time series can be processed. The sample from each time series consists of data values for the full period or a shorter analysis period if specified for the command. Missing values typically are ignored unless significant for the statistic (e.g., Statistic=MissingCount).

The following dialog is used to edit the command and illustrates the command syntax. Most statistics do not require additional input; however, those that do utilize the Value* parameters to specify additional information.

Edit CalculateTimeSeriesSt	tatistic() Command	X								
Calculate a statistic for time series and optionally save in a table.										
The table and its columns will be created if not found.										
Statistics results may include 1+ values and may include the date/time of the result.										
Specify dates with precision appropriate for the data, use blank for all available data, OutputStart, or OutputEnd.										
TS list:		Optional - indicates the time series to process (default=AllTS).								
TSID (for TSList=AllMatchingTSID):		\checkmark								
EnsembleID (for TSList=EnsembleID):		\checkmark								
Statistic to calculate:	NqYY	Required - may require other parameters.								
Value1:	7	Optional - may be needed as input to calculate statistic.								
Value2:	10	Optional - may be needed as input to calculate statistic.								
Value3:	0	Optional - may be needed as input to calculate statistic.								
Analysis start:		Optional - analysis start date/time (default=full time series period).								
Analysis end:		Optional - analysis end date/time (default=full time series period).								
🗌 Analysis window:	Month: Day: Hour:	Optional - analysis window within input year (default=full year).								
Table ID for output:	Table1	Optional - if statistic should be saved in table.								
Table TSID column:	TSID	Required if using table - column name for TSID.								
Format of TSID:	Select Specifier 💙 =>	Optional - use %L for location, etc. (default=alias or TSID).								
Table statistic column:	7q10	Required if using table - column name(s) for statistic(s).								
Command:	CalculateTimeSeriesStatistic(Stat: ,TableID="Table1",TableTSIDColumn=	istic="NqYY",Value1=7,Value2=10,Value3=0 ="TSID",TableStatisticColumn="7q10")								
	Cancel OK	ColoulotoTimeSeriesStatistic								

CalculateTimeSeriesStatistic() Command Editor

The command syntax is as follows:

```
CalculateTimeSeriesStatistic(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one	Allts
	of:	
	• AllMatchingTSID – all time series that	
	match the TSID (single TSID or TSID with	
	wildcards).	
	• AllTS – all time series before the command.	
	• EnsembleID – all time series in the ensemble.	
	• FirstMatchingTSID - the first time series	
	that matches the TSID (single TSID or TSID	
	with wildcards).	
	• LastMatchingTSID – the last time series	
	that matches the TSID (single TSID or TSID	
	with wildcards).	
	• SelectedTS – the time series selected with	
	the SelectTimeSeries() command.	
TSID	The time series identifier or alias for the time series	Required if
	to be processed, using the * wildcard character to	TSList=*TSID.
	match multiple time series.	
EnsembleID	The ensemble to be processed, if processing an	Required if
	ensemble.	TSList=EnsembleID.
Statistic	Statistic to compute as shown in the Statistic	None – must be specified.
	Details table below.	
Value1	Input data required by the statistic. Currently the	See Statistic Details
	dialog does not check the value for correctness – it	table below.
	is checked when the statistic is computed.	
Value2	Input data required by the statistic. Currently the	See Statistic Details
	dialog does not check the value for correctness – it	table below.
	is checked when the statistic is computed.	-
Value3	Input data required by the statistic. Currently the	See Statistic Details
	dialog does not check the value for correctness – it	table below.
	is checked when the statistic is computed.	
AnalysisStart	The date/time to start analyzing data.	Full period is analyzed.
Analysisend	The date/time to end analyzing data.	Full period is analyzed.
Analysis	The calendar date/time for the analysis start within	Analyze the full year.
WINdOwstart	each year. Specify using the format MM, MM–DD,	
	MM-DD hh, or MM-DD hh:mm, consistent with the	
	time series interval precision. A year of 2000 will	
	be used internally to parse the date/time. Use this	
	parameter to limit data processing within the year,	
	tor example to analyze only a season. The analysis	
	window has only been enabled for Count,	
	GECount, GTCount, LECount, LTCount, Max,	

Parameter	Description	Default
	Min,MissingCount,MissingPercent,	
	NonmissingCount, and	
	NonmissingPercent statistics.	
Analysis	Specify date/time for the analysis end within each	Analyze the full year.
WindowEnd	year. See AnalysisWindowStart for details.	
TableID	Identifier for table that receives the statistic. An	Optional – table output is
	existing table can be specified. If not found, a new	not required.
	table will be created.	
TableTSIDColumn	Table column name that is used to look up the time	Optional – table output is
	series. If a matching TSID is not found, a row will	not required.
	statistic cell value for the time series is modified.	
TableTSIDFormat	The specification to format the time series identifier	Time series alias if
	to insert into the TSID column. Use the format	available, or the time
	choices and other characters to define a unique	series identifier.
	identifier.	
TableStatistic	Table column name to receive the statistic value. If	Optional – table output is
Column	not found in the table, a new column is added	not required.
	automatically.	

The following table provides additional information about specific statistics, in particular to describe how the statistic is computed and whether additional input needs to be provided with Value command parameters.

Statistic Details

Statistic	Description	Required Values
Count	Number of data values total, including missing and non-missing.	
DeficitMax	Maximum deficit value (where deficit is mean minus value).	
DeficitMean	Mean deficit value (where deficit is mean minus value).	
DeficitMin	Minimum deficit value (where deficit is mean minus value).	
DeficitSeqLengthMax	Maximum number of sequential intervals where each value is less than the mean (for example maximum drought length).	
DeficitSeqLengthMean	Mean number of sequential intervals where each value is less than the mean (for example mean drought length).	
DeficitSeqLengthMin	Minimum number of sequential intervals where each value is less than the mean (for example minimum drought length).	
DeficitSeqMin	Maximum sum of sequential values where each value is less than the mean (for example maximum drought water volume).	
DeficitSeqMean	Mean of the sum of sequential values where each	

Statistic	Description	Required Values
	value is less than the mean (for example mean drought water volume).	
DeficitSeqMin	Minimum sum of sequential values where each value is less than the mean (for example minimum drought water volume).	
GECount	Count of values greater than or equal to Value1.	Value1 – criteria to check
GTCount	Count of values greater than Value1.	Value1 – criteria to check
Lag-1AutoCorrelation	Autocorrelation between values and the those that follow in the next time step, given by: $r_{k} = \frac{\sum_{i=1}^{N-k} (Y_{i} - Y_{mean})(Y_{i+k} - Y_{mean})}{\sum_{i=1}^{N} (Y_{i} - Y_{mean})^{2}}$	
Last	Last non-missing value.	
LECount	Count of values less than or equal to Value1.	Value1 – criteria to check
LTCount	Count of values less than Value1.	Value1 – criteria to check
Max	Maximum value.	
Mean	Mean value.	
Min	Minimum value.	
MissingCount	Number of missing values.	
MissingPercent	Percent of values that are missing.	
MissingSeqLengthMax	Maximum number of sequential values that are missing.	
NonmissingCount	Number of non-missing values.	
NonmissingPercent	Percent of values that are not missing.	
NqYY	 This statistic is typically used to evaluate the return period of low flows and is implemented only for daily data. The N indicates the number of daily values to be averaged and YY indicates the return interval. For example, 7q10 indicates the flow corresponding to the 10-year recurrence interval for minimum average daily flow (for 7 days) in a year. This statistic is computed as follows, using 7q10 as an example: 1. Determine the number of years to be analyzed (from analysis period command parameters or time series data). 	Value1 – specify the number of daily values to be averaged. Currently this must be an odd number to allow bracketing the current day.
	2. For each year, loop through each day from January 1 to December 31. Compute an average flow by averaging 7 days, in this case with 3 values on each side of the current day and including the current day. If at the end of the year, use 3 values from adjoining years. The number of missing data allowed is controlled by the Value3 command parameter.	Value2 – specify the return interval (e.g., 10). Value3 – specify the number of

Statistic	Description	Required Values
	 For the year, save the minimum 7-day average. Utilize the minimum values for all years, with log- Pearson Type III distribution, to determine the value for the 10-year recurrence interval. See <u>http://pubs.usgs.gov/sir/2008/5126/section3.html</u> for a description of NqYY and "Hydrology for Engineers, 3rd Edition," Linsley, Kohler, Paulhus for a description of log-Pearson Type III distribution. 	missing values allowed in the average (e.g., 0 for most rigorous analysis). It may be useful to set this value if, for example, a single daily value is available in the time series, for example entered on the first day of the month.
Skew	Skew coefficient, as follows: $Cs = \frac{N \sum_{i=1}^{N} (Y_i - Y_{mean})^3}{(n-1)(n-2)s^3}$	
	where $s = \text{standard deviation}$	
StaDev	Standard deviation.	
SurplusMin	Maximum surplus value (where surplus is value minus mean).	
SurplusMean	Mean surplus value (where surplus is value minus mean).	
SurplusMin	Minimum surplus value (where surplus is value minus mean).	
SurplusSeqLengthMax	Maximum number of sequential intervals where each value is greater than the mean (for example maximum water surplus length).	
SurplusSeqLengthMean	Mean number of sequential intervals where each value is greater than the mean (for example mean water surplus length).	
SurplusSeqLengthMin	Minimum number of sequential intervals where each value is greater than the mean (for example minimum water surplus length).	
SurplusSeqMin	Maximum sum of sequential values where each value is greater than the mean (for example maximum water surplus volume).	
SurplusSeqMean	Mean of the sum of sequential values where each value is greater than the mean (for example mean water surplus volume).	
SurplusSeqMin	Minimum sum of sequential values where each value is greater than the mean (for example minimum water surplus volume).	
Total	Total of values.	
TrendOLS	Ordinary least squares analysis is used to compute results that are named TableStatisticColumn	

Statistic	Description	Required Values
	with appended _Intercept, _Slope, and _R2.	
Variance	Variance.	

The following example illustrates how to use the command to compute the 7q10 statistic for daily flow:

```
ReadDateValue(Alias="linsley", InputFile="Data\linsley.dv")
NewTable(TableID="Table1", Columns="TSID, string; 7q10, double")
CalculateTimeSeriesStatistic(Statistic="NqYY", Value1=7, Value2=10, Value3=6,
TableID="Table1", TableTSIDColumn="TSID", TableStatisticColumn="7q10")
WriteTableToDelimitedFile(TableID="Table1",
OutputFile="Results/Test_CalculateTimeSeriesStatistic_7q10_linsley_out.csv")
```

Command Reference: ChangeInterval()

Create new time series by changing the input time series data interval

Version 10.10.01, 2011-04-18

The ChangeInterval() command creates new time series by changing the data interval of each input time series. A list of one or more time series or an ensemble of time series can be processed. The majority of the original header data (e.g., description, units) are copied to the new time series; however, the new interval will be used for data management and in the new time series identifier. Time series data values have a time scale of instantaneous, accumulated (e.g., volume), or mean. Changing the interval also can result in a change in the time scale (e.g., converting instantaneous values to a mean value). Currently, the time scale for input and output time series is NOT automatically determined from the data type and interval and must be specified as ACCM, MEAN, or INST. Instantaneous values are recorded at the date/time of the value and typically apply to small intervals (e.g. minute and hour). For mean and accumulated time series, the date/time for each value is at the end of the interval for which the value applies.

Irregular time series have a date/time precision and a scale appropriate for the data. For example, irregular minute time series may be used for instantaneous temperature or accumulated precipitation. Irregular day time series may be used for "instantaneous" reservoir level. For regular time series, the data intervals must align so that each larger interval aligns with the end-points of the corresponding smaller intervals (e.g., the ends of 6-hour intervals align with the daily interval).

The following conversions are currently supported, with a description of the conversion process. Refer to the command parameter reference for an explanation of parameters. The conversion from daily and monthly interval to yearly interval (for ACCM and MEAN) utilizes a simpler algorithm.

Irregular Time Series to Regular Time Series

An irregular time series can be converted to a regular time series. The ability to change from an irregular or regular time series to an irregular time series currently is not implemented. Missing data is handled in different ways depending on the old and new time scales. Each of the follow examples demonstrates how missing data is interpreted.

The following conversion combinations are allowed.

Small Interval ACCM to Large Interval ACCM

When converting from small interval accumulated data to large interval accumulated data, values from the old time series are summed for the new interval-ending date/time from the values in the old intervals prior to this date/time.

The following illustrates the conversion from NHour to NHour (1Hour to 3Hour example):

Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,
Hour 0	Hour 1	Hour 2	Hour 3	Hour 4	Hour 5	Hour 6
(A)	(B)	(Missing)	(C)	(Missing)	(Missing)	(Missing)
Day 1,		Day	1, Hour 3		Day	y 1, Hour 6
Hour 0			=B+C			= Missing
=A						-

Large Interval ACCM to Small Interval ACCM

When converting from large interval accumulated data to small interval accumulated data, values from the old time series are equally divided by the number of intervals prior to this date/time in the new time series since the previous non-missing data.

	1	_		1	_		1	_	
Day 1,	Day 1, Hour 3		Day 1, Hour 3 Day 1, Hour 6		Day 1, Hour 9				
Hour 0	(B)		(B) (Missing)		(C)		(C)		
(A)						-			
Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,
Hour 0	Hour 1	Hour 2	Hour 3	Hour 4	Hour 5	Hour 6	Hour 7	Hour 8	Hour 9
=A	=B/3	=B/3	=B/3	=C/6	=C/6	=C/6	=C/6	=C/6	=C/6

The following illustrates the conversion from NHour to NHour (3Hour to 1Hour example):

Small Interval MEAN or INST to Large Interval MEAN

When converting from instantaneous or mean data to mean data, mean values are calculated for the new interval-ending date/time from the values in the old intervals prior to this date/time.

The following illustrates the conversion from NHour to NHour (1Hour to 3Hour example):

Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,
Hour 0	Hour 1	Hour 2	Hour 3	Hour 4	Hour 5	Hour 6
(A)	(B)	(Missing)	(C)	(Missing)	(Missing)	(Missing)
Day 1,		Day	1, Hour 3		Day	y 2, Hour 6
Hour 0		:	=(B+C)/2			= Missing
=A						-

Large Interval MEAN or INST to Small Interval MEAN

When converting from large interval mean or instantaneous data to small interval mean data, values from the old time series are copied to the new interval-ending date/time time series.

The following illustrates the conversion from NHour to NHour (3Hour to 1Hour example):

Day 1,	Day 1, Hour 3			Day 1, Hour 6			Day 1, Hour 9		
Hour 0 (A)	(B)			(Missing)					(C)
Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,
Hour 0	Hour 1	Hour 2	Hour 3	Hour 4	Hour 5	Hour 6	Hour 7	Hour 8	Hour 9
=A	=B	=B	=B	=C	=C	=C	=C	=C	=C

Small Interval INST to Large Interval INST

When converting from small interval instantaneous data to large interval instantaneous data, the data is copied directly from the old time series when available. If the data is missing, the most recent previous valid data is used.

The following illustrates the conversion from NHour to NHour (1Hour to 3Hour example):

Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,
Hour 0	Hour 1	Hour 2	Hour 3	Hour 4	Hour 5	Hour 6	Hour 7	Hour 8
(A)	(B)	(Missing)	(C)	(Missing)	(D)	(Missing)	(E)	(F)
Day 1, Hour 0			Day 1, Hour 3			Day 1, Hour 6		
=A			=C			=D		

Large Interval INST to Small Interval INST

When converting from large interval instantaneous data to small interval instantaneous data, values from the old time series are linearly interpolated to calculate values for the new time series.

The following illustrates the conversion from NHour to NHour (3Hour to 1Hour example):

Day 1,		Day 1, Hour 3			Day 1, Hour 6			Day 1, Hour 9		
Hour 0 (A)	(B)			(Missing)			(C)			
Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	
Hour 0	Hour 1	Hour 2	Hour 3	Hour 4	Hour 5	Hour 6	Hour 7	Hour 8	Hour 9	
=A	=A+	=A+	=B	=B+	= B+	= B+	$= \mathbf{B} +$	$= \mathbf{B} +$	=C	
	(B-A)*	(B-A)*		(C-B)*	(C-B)*	(C-B)*	(C-B)*	(C-B)*		
	(1/3)	(2/3)		(1/6)	(2/6)	(3/6)	(4/6)	(5/6)		

Regular Time Series to Regular Time Series

ACCM (Accumulation) to ACCM (Accumulation)

Small Interval ACCM (Accumulation) to Large Interval ACCM (Accumulation)

Changing the interval for small interval accumulated data to large interval accumulated data involves summing the small interval data values for the period that overlaps the large interval.

Accumulated data have a timestamp corresponding to the interval-end for the accumulation. Conversions involving time intervals that have zero values (e.g., Hour 0, Minute 0) result in a perceived shift in time because the zero occurs on the boundary between larger intervals. The following examples illustrate the accumulation for common cases. In cases where an accumulation jumps over two or more interval categories (e.g., minute to day), the accumulation occurs as if the two intermediate accumulations occurred in succession. In the following examples, the general representation is shown first, followed by an example where appropriate.

The following illustrates the conversion from NHour to Day (6Hour to Day example, i equals the hour multiplier):

Day 1,	Day 1,	Day 1,	Day 1,	Day 2,				
Hour 0	Hour <i>i</i>	Hour 2i	Hour 3i	Hour 0				
Day 1,	Day 1,	Day 1,	Day 1,	Day 2,				
Hour 0	Hour 6 (A) Hour 12 (B) Hour 18 (C) Hour 0 (D)							
	Day 1 accumulation (A+B+C+D)							

The following illustrates the conversion from NDay to Month (example for a month with 30 days):

Month 1,				Month 1, Day					
Day 1 (A1)				30 (A30)					
Month 1 accumulation $(A1 + + A30)$									

Large Interval ACCM (Accumulation) to Small Interval ACCM (Accumulation)

Changing from large interval accumulation data to small interval mean data involves dividing each accumulated value by the number of new values for that same period of record.

The following illustrates the conversion from Day to 6Hour (Day to 6Hour example, *i* equals the hour multiplier):

Day 1 acc	umulate (A	4)	
Day 1,	Day 1,	Day 1,	Day 1,
Hour 0	Hour <i>i</i>	Hour 2i	Hour 3i
Day 1,	Day 1,	Day 1,	Day 1,
Hour 0	Hour 6	Hour 12	Hour 18
= A/4	= A/4	= A/4	= A/4

ACCM (Accumulation) to INST (Instantaneous)

Accumulated to instantaneous is not currently supported.

ACCM (Accumulation) to MEAN

Small Interval ACCM to Large Interval MEAN

See Small Interval INST (Instantaneous) to Large Interval MEAN.

Interval ACCM to Same Interval MEAN

Changing the interval from accumulation data to the same interval mean data involves copying the data from the old time series to the new time series (no changes to date values occur).

The following illustrates the conversion from 6Hour to 6Hour (6Hour to 6Hour example, *i* equals the hour multiplier):

Day 1,	Day 1,	Day 1,	Day 1,
Hour 0	Hour <i>i</i>	Hour 2i	Hour 3i
Day 1,	Day 1,	Day 1,	Day 1,
Hour 0	Hour 6	Hour 12	Hour 18
(A)	(B)	(C)	(D)
Day 1,	Day 1,	Day 1,	Day 1,
Hour 0	Hour 6	Hour 12	Hour 18
=A	=B	=C	=D

Large Interval ACCM to Small Interval MEAN

See Large Interval ACCM to Small Interval ACCM.

INST (Instantaneous) to INST (Instantaneous)

Small Interval INST (Instantaneous) to Large Interval INST (Instantaneous)

Changing the interval for small interval instantaneous data to large interval instantaneous data involves assigning each date in the new time series a value from the corresponding date in the old time series. The HandleMissingInputHow parameter indicates how to interpret a missing value in the old time series. HandleMissingInputHow=KeepMissing will simply assign a missing value for that date/time. HandleMissingInputHow=SetToZero will set the value to 0. Repeat fills the date with data from the last non-missing value. Interpolation and using a non-missing future value may be added in the future.

A special case is the ability to compute a statistic from the sample of values from the input time series, using the Statistic parameter. For example, instantaneous 5 minute temperature data can be converted to 1 day maximum values. In this case, each 1 day sample of values from the input time series is used to compute the statistic. The initial handling of missing data described above is supported and additionally the AllowMissingCount parameter is recognized to control computation of the statistic.

The following illustrates the conversion from NHour to Day (6Hour to Day example where HandleMissingInputHow = Repeat, *i* equals the hour multiplier):

Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,	Day 1,
Hour 0	Hour <i>i</i>	Hour 2i	Hour 3i	Hour 0	Hour <i>i</i>	Hour 2i	Hour 3i
Day 1,	Day 1,	Day 1,	Day 1,	Missing	Day 1,	Day 1,	Day 1,
Hour 0	Hour 6	Hour 12	Hour 18	data	Hour 6	Hour 12	Hour 18
(A)	(B)	(C)	(D)		(E)	(F)	(G)
Day 1 instantaneous = A				Day 2 instantaneous = D			

Large Interval INST (Instantaneous) to Small Interval INST (Instantaneous)

Small interval instantaneous data is created from larger interval instantaneous data by linearly interpolating between the previous and current large interval data to fill each value in the new time series during that same period of time. If the value in the old time series is missing, the method specified by the user in the HandleMissingInputHow parameter is used.

The following illustrates the conversion from Day to NHour (Day to 6Hour example, *i* equals the hour multiplier):

Day 1 ins	tantaneous (A	Day 2 in	stantaneou	is (B)		Day 3		
Day 1,	Day 1,	Day 1,	Day 1,	Day 2,	Day 2,	Day 2,	Day 2,	
Hour 0	Hour <i>i</i>	Hour 2 <i>i</i>	Hour <i>3i</i>	Hour 0	Hour i	Hour 2i	Hour 3i	
Day 1,	Day 1,	Day 1,	Day 1,	Day 2,	Day 2,	Day 2,	Day 2,	
Hour 0	Hour 6	Hour 12	Hour 18	Hour 0	Hour 6	Hour 12	Hour 18	
=A	=A+	=A+	=A+	=B				
	(B-A)*	(B-A)*	(B-A)*					
	(6/24)	(12/24)	(18/24)					
These val	ues are an int	erpolated value	ie between	These va				
the Day 1 instantaneous value and the Day 2				between the Day 2 instantaneous value				
instantaneous value using a time of 24 hours.				and the Day 3 instantaneous value using				
					a time of 24 hours.			

In the future, the ability to repeat input values may be added.

INST (Instantaneous) to ACCM (Accumulation)

Instantaneous to accumulated is not currently supported.

INST (Instantaneous) to MEAN

Small Interval INST (Instantaneous) to Large Interval MEAN

Changing from small interval instantaneous data to large interval mean data involves adding together all the values from the small interval time series over the larger interval for the corresponding time period and then dividing by the number of data values used within this calculation. As in other conversions, HandleMissingInputHow is first used to interpret missing data. If HandleEndpointHow = AverageEndpoints, the values at each end of the interval are averaged for minute and hour inputs (the parameter does not apply to day, month or year input).

The following illustrates the conversion from NHour to Day (6Hour to Day example with HandleEndpointHow = IncludeFirstOnly, *i* equals the hour multiplier):

Day 1,	Day 1,	Day 1,	Day 1,	Day 2,	Day 2,	Day 2,	Day 2,
Hour θ	Hour <i>i</i>	Hour 2i	Hour 3i	Hour 0	Hour <i>i</i>	Hour 2i	Hour 3i
Day 1,	Day 1,	Day 1,	Day 1,	Day 2,	Day 2,	Day 2,	Day 2,
Hour 0	Hour 6	Hour 12	Hour 18	Hour 0	Hour 6	Hour 12	Hour 18
Value A	В	С	D	Е	F	G	Н
Day 1 mean= $(A+B+C+D)/4$				Day 2 mean= $(E+F+G+H)/4$			

Day 1,	Day 1,	Day 1,	Day 1,	Day 2,	Day 2,	Day 2,	Day 2,	1
Hour 0	Hour <i>i</i>	Hour 2i	Hour 3i	Hour 0	Hour <i>i</i>	Hour 2i	Hour 3i	
Day 1,	Day 1,	Day 1,	Day 1,	Day 2,	Day 2,	Day 2,	Day 2,]
Hour 0	Hour 6	Hour 12	Hour 18	Hour 0	Hour 6	Hour 12	Hour 18	
Value A	В	С	D	Е	F	G	Н	Ι
Day 1 mean= $((A+E)/2 + B+C+D)/4$ Day 2 mean= $((E+I)/2 + F+G+H)/4$								

The following illustrates the conversion from NHour to Day (6Hour to Day example with HandleEndpointHow = AverageEndpoints, *i* equals the hour multiplier):

Interval INST (Instantaneous) to Same Interval MEAN

If OutputFillMethod = Interpolate, see Large Interval INST (Instantaneous) to Small Interval INST (Instantaneous). Otherwise, the values are duplicated from the old time series directly to the new time series.

The following illustrates the conversion from 6Hour to 6Hour (6Hour to 6Hour example with OutputFillMethod = Repeat, *i* equals the hour multiplier):

Day 1,	Day 1,	Day 1,	Day 1,	Day 2,
Hour 0	Hour <i>i</i>	Hour 2i	Hour 3i	Hour 0
Day 1,	Day 1,	Day 1,	Day 1,	Day 2,
Hour 0	Hour 6	Hour 12	Hour 18	Hour 0
(A)	(B)	(Missing)	(D)	(E)
Day 1,	Day 1,	Day 1,	Day 1	Day 2,
Hour 0	Hour 6	Hour 12	Hour 18	Hour 0
=A	=B	=B	=D	=Е

Large Interval INST (Instantaneous) to Small Interval MEAN

If the OutputFillMethod = Interpolate, see Large Interval INST (Instantaneous) to Small Interval INST (Instantaneous). The time series are handled in the same way. Otherwise, the values are duplicated from the old time series directly to the new time series.

The following illustrates the conversion from Day to 6Hour (Day to 6Hour example with OutputFillMethod = Repeat, *i* equals the hour multiplier):

Day 1 instantaneous = A					
Day 1,	Day 1, Day 1, Day 1,				
Hour 0	Hour <i>i</i>	Hour 2i	Hour 3i		
Day 1,	Day 1,	Day 1,	Day 1,		
Hour 0	Hour 6	Hour 12	Hour 18		
=A	=A	=A	=A		
Each of these values is equal to the					
instantaneous value for that day.					

MEAN to MEAN

Small Interval MEAN to Large Interval MEAN

See Small Interval INST (Instantaneous) to Large Interval MEAN.

Large Interval MEAN to Small Interval MEAN

Changing from large interval mean data to small interval mean data involves copying values from the old time series into the new time series for that same period of record.

The following illustrates the conversion from Month to Day (Example for a month with 30):

Month Mean (A)			
Day 1	Day 2		Day 30
=A	=A		=A

MEAN to ACCM (Accumulation)

Small Interval MEAN to Large Interval ACCM (Accumulation)

See Small Interval INST (Instantaneous) to Large Interval MEAN.

Interval MEAN to Same Interval ACCM (Accumulation)

See Interval ACCM to Same Interval MEAN.

Large Interval MEAN to Small Interval ACCM (Accumulation)

See Large Interval ACCM to Small Interval ACCM.

MEAN to INST (Instantaneous)

Small Interval MEAN to Large Interval INST (Instantaneous)

Not currently supported.

Interval MEAN to Same Interval INST (Instantaneous)

Not currently supported. The data can be treated equivalently by most commands.

Large Interval MEAN to Small Interval INST (Instantaneous)

Changing the interval for large interval mean to small interval instantaneous data involves calculating a value for each new interval based on trends found in the mean data. This approach has been adapted from the NWSRFS CHANGE-T operation (see

<u>http://www.nws.noaa.gov/oh/hrl/nwsrfs/users_manual/part5/_pdf/533changet.pdf</u>). The following example demonstrates how the data is converted from the old interval to the new interval. A general representation is shown first followed by an example.

The following illustrates the conversion from Day to NHour (Day to 6Hour example, *i* equals the hour multiplier):



In computing instantaneous values, the volume of the original mean time series needs to be maintained. However, the value of the instantaneous endpoints affects the calculated mean value for the previous and subsequent long intervals, since the mean over a longer interval uses both endpoints in its calculation. In the above example, Average_{new Day 1} = ((A+E)/2 + B+C+D)/4

As a result, an iterative technique is required to adjust the initially computed instantaneous values to produce a time series with a volume that is within a specified tolerance of the input mean volume for each time step. The following paragraphs describe how the initial instantaneous time series values are computed, followed by a description of the volume adjustment.

Initial Instantaneous Time Series Calculations

Prior to converting from a large interval mean to small interval instantaneous, special cases are handled associated with missing data:

- Missing data is initially converted using the method specified by the user in the HandleMissingInputHow parameter.
- If the current input value is still missing, the instantaneous time series is also filled with missing data for each interval that falls in the larger interval.
- If the previous or next mean values are missing, the current mean value for that interval is copied directly to the instantaneous time series.

The output instantaneous values for each input interval are computed using the current, next, and previous mean values. All three values are useful because together they indicate whether the current value is part of a continuous rise or fall, a peak or trough or simply a continuation of a steady value. These conditions are illustrated in the following figure.



Mean data illustration

- The first condition that may exist is a **peak or trough**. A peak exists when the current value is greater than the previous and next values. A trough is when the current value is less than the next and previous values.
 - 1. In this case, an *instantaneous peak* (or trough) is calculated. Referring to the above illustration, the magnitude of the peak is calculated by adding (or subtracting for a trough) $\frac{1}{4} (a+b)/2$ to the current mean.
 - 2. The *time of the instantaneous peak* is initially set to the start date/time of the current interval then shifted forward in time using the following calculation. The number of instantaneous intervals per larger interval is multiplied by b/(a+b). That result is added to the start date/time. The time of the instantaneous peak will not necessarily correspond to the output interval.
 - 3. The value for the starting endpoint of the interval is set to the *current value minus* $\frac{1}{4}a$.
 - 4. The value for the ending endpoint of the interval is set to the *current value minus* $\frac{1}{4}b$.
 - 5. The remaining intermediate instantaneous values for the interval are linearly interpolated between the peak (or trough) and both endpoints.
- The second condition that may exist is a **continuous rise or fall**. A continuous rise or fall exists when the current value is between the previous and next values.
 - 1. In the case of a continuous rise, the starting endpoint of the interval is set to the *current* value minus $\frac{1}{4}c$ (again using the above illustration). In the case of a continuous fall, $\frac{1}{4}c$ is added to the current value.

- 2. The ending endpoint of the interval of a continuous rise is set to the *current value plusl*/4 c (minus ¹/₄ c for a continuous fall)
- 3. The remaining intermediate instantaneous values are calculated based on the following.
 - a. The difference between the starting and ending endpoints is computed.
 - b. The values *c* and *a* are calculated. The ratio of mean differences is computed: If c > a, the mean ratio = c / a. If a > c, the mean ratio = a / c.
 - c. If c is less than a, then the intermediate instantaneous values are computed by adding small but increasing increments to the starting endpoint until the last point of the interval is reached. If a is less than c then the output values are computed by subtracting small but increasing increments to the last endpoint until the first point of the interval is reached. For intermediate interval n from the appropriate endpoint:

 $Increment_n = (1/([number intervals] - n) * [endpoint difference] * [mean ratio])$

Instantaneous Time Series Volume Adjustment

After the instantaneous values are estimated using the above set of rules, they are adjusted so that the volume over each interval is within a specified tolerance of the input mean volume for each time step. This tolerance is specified with the Tolerance parameter. The volume calculation for each interval uses the average of the first and last endpoint.

For each time step of the original mean time series, the error of each original interval is computed as:

([mean of current instantaneous values] – [original mean]) / [original mean]

If this volume error is within the tolerance, no adjustment is made for that time interval. If the error is larger than the tolerance, the ratio: [original mean] / [mean of current instantaneous values] is computed. For the intermediate instantaneous values, the instantaneous values are adjusted by multiplying each value by this ratio. For the instantaneous endpoint values, since these values affect the mean value of the current and the previous or following time intervals, the endpoint values are not adjusted to the above calculated ratio. Instead, the new endpoint value is computed as the average of the original endpoint value and the original endpoint value multiplied by the above ratio.

The volume error is checked for each original time step. If any adjustments were made, the process is repeated, up to 15 iterations. If the adjustment technique cannot adjust the instantaneous time series such that the corresponding mean volume is within the specified tolerance of the input mean volume within 15 iterations, a warning will be written to the log file.

The following dialog is used to edit the command and illustrates the syntax for the command. This example is converting a monthly volume time series to annual water year (October to September) volumes.

👌 Edit ChangeInterv	al() Command			
Create new time series by changing the data interval of each input time series.				
The output time series will have an identifier that is the same as the input, but with the new interval.				
The conversion process dep	ends on whether the original and	new time series contain	n accumulated, mean, or instantaneous data.	
The time scales must be spe	cified (they are not automatically	determined from the da	ata type).	
Many of the advanced p and are mainly for inter	parameters depend on the inj	out data interval, wi ers are enabled/di	hich may only be confirmed when commands are run, sabled as much as possible but see the documentation for details	
Time series to process-		ers are enabled, al		
	TS list: LastMatchingTSID	•	Optional - indicates the time series to process (default=AlITS).	
TSID (for TSI	ist=*TSID: Original		×	
EnsembleID (for TSList=Er	nsembleID):		×	
New ensemble ID:]	Optional - to create ensemble when input is an ensemble.	
New ensemble name:			Optional - name for new ensemble.	
Alias to assign:	New Insert:	Select Specifier	Required - use %L for location, etc.	
New interval:	Year 🔽		Required - data interval for result.	
Old time scale:	ACCM - Accumulated 🔽		Required - indicates how to process data.	
New time scale:	ACCM - Accumulated 🔽		Required - indicates how to process data.	
Statistic to calculate:	~		Optional - limited support for INST to INST (default statistic is from old/new time scale).	
Output year type:	Water 💌		Optional - use only when old interval is day or month and new interval is Year (default=Calendar).	
New data type:			Optional - data type (default = original time series data type).	
New units:			Optional - data units (default = original time series units).	
Tolerance:			Optional - convergence tolerance (default = 0.01).	
Handle endpoints how?:	~		Optional - for INST to MEAN, small to large, new interval=day or less (default=AverageEndpoints).	
Allow missing count:			Optional - number of missing values allowed in input interval (default=0).	
Allow consecutive missing:			Optional - number of consecutive missing values allowed in input interval (default=AllowMissingCount).	
Output fill method:	×		Optional - use when converting from INST to MEAN, large to small interval (default=Repeat).	
Handle missing input how?:	~		Optional - how to handle missing values in input (default=KeepMissing).	
	ChangeInterval(Alias	="New",TSList=	=LastMatchingTSID,TSID="Original",NewInterval=Year,OldTimeScale	
Compandi	=ACCM,NewTimeScale=A	CCM,OutputYear	rType=Water)	
Command:				
Cancel				
		l		

ChangeInterval() Command Editor

ChangeInterval

The command syntax is as follows:

ChangeInterval(Parameter=Value,...)

The following older command syntax is updated to the above syntax when a command file is read:

TS Alias = ChangeInterval(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one	AllTS
	of:	
	• AllMatchingTSID – all time series that match	
	the TSID (single TSID or TSID with wildcards)	
	• AllTS – all time series generated before the	
	command	

Parameter	Description	Default
	• EnsembleID – all time series in the ensemble	
	• FirstMatchingTSID – the first time series	
	that matches the TSID (single TSID or TSID	
	with wildcards)	
	• LastMatchingTSID – the last time series that	
	matches the TSID (single TSID or TSID with	
	wildcards)	
	• SelectedTS – the time series selected with the	
	SelectTimeSeries() command	
TSID	The time series identifier or alias for the time series to	Required if
	be processed, using the * wildcard character to match	TSList=*TSID.
	multiple time series.	
EnsembleID	The ensemble to be processed, if processing an	Required if TSList=
	ensemble.	EnsembleID.
Alias	The alias to assign to the time series, as a literal string	None – must be specified.
	or using the special formatting characters listed by the	
	command editor. The alias is a short identifier used	
	by other commands to locate time series for	
	processing, as an alternative to the time series	
	identifier (TSID).	
NewInterval	The data interval for the new time series, from the	None – must be specified.
	provided choices. For example: 6Hour, Day,	
	Month, Year.	
OldTimeScale	The time scale for the original time series, one of:	None – must be specified.
	ACCM – accumulated data	
	INST – instantaneous data	
	MEAN – mean data	
	In the future, this parameter may be made optional if	
NormimoGralo	the time scale can be determined from the data type.	NT (1 °C' 1
NewlineScale	The time scale for the new time series (see	None – must be specified.
	Old'l'imeScale for possible values). In the future,	
	this parameter may be made optional if the time scale	
Statistic	List in the asses where TNGT (small internal) to	The statistic is
Statistic	Used in the case where INST (sman interval) to	determined from the old
	sample of values from the input time series	and now time scales
	corresponding to the output interval is determined	and new time scales.
	and used to compute a statistic instead of a simple	
	value transfer. Statistics that are currently supported	
	are MAX and MIN. The	
	HandleMissingInput How parameter is initially	
	used to adjust missing data and then the	
	AllowMissingCount parameter is used to check	
	whether the statistic can be computed	
OutputYearType	The output year type if the output time series has an	Calendar
	interval of Year The output ver type can only be	
	specified for input time series having an interval of	

Parameter	Description	Default
	Day or Month and the output can have a time scale	
	of ACCM (sum the input values) or MEAN (average the	
	input values). The AllowMissingCount and	
	AllowMissingConsecutive parameters are	
	recognized.	
NewDataType	The data type for the new time series. This will be set	Use the data type from
	in the identifier of the new time series.	the original time series.
NewUnits	The units for the new time series. This will be set in	Use the units from the
	the identifier of the new time series.	original time series.
Tolerance	Currently used when converting large interval MEAN	0.01
	data to small interval INST data. After the new time	
	series is created, the volume of the new time series	
	over each old interval is compared to the old time	
	series for that same interval. If the difference	
	between the two is outside the specified tolerance	
	percentage, then each value in the new time series is	
	adjusted so the totals will match. The endpoints are	
	averaged for this comparison. Additionally, when the	
	adjustment is made, the new starting value is	
	averaged with the ending value of the previous	
	interval so that the previous interval is not overly	
	affected by this calculation.	
Handle	Indicates how endpoints should be handled when	Average Endpoints
EndpointsHow	changing from INST to MEAN, small interval to	
	larger interval (daily output or finer), one of:	
	AverageEndpoints – use both endpoint values	
	for new single value	
	IncludeFirstOnly – only use earlier endpoint	a 1
AllowMissing	The number of missing values allowed in the input	0 - do not allow any
counc	interval in order to produce a result. For example, if	missing data in the source
	converting daily data to monthly, a value of 5 would	data when computing a
	allow <= 5 missing daily values and still compute the	result.
	heading it may regult in data that are not	
	because it may result in data that are not	
	considered after the HandloMiggingHow	
	parameter	
AllowMissing	The number of consecutive missing values allowed in	If not specified the
Consecutive	the input interval in order to produce a result. For	default for the number of
	example if converting daily data to monthly a value	allowed consecutive
	of 3 would allow ≤ 3 consecutive missing daily	missing values is set to
	values and still compute the result. The value must	AllowMissingCount.
	be less than or equal to Allow MissingCount.	
	This parameter is considered after the	
	HandleMissingHow parameter.	
OutputFill	Use to fill output when converting from INST to	Repeat
Method	MEAN, large interval time series to small interval	
	time series, one of:	

Parameter	Description	Default
	Interpolate – linearly interpolate Repeat – repeat values for the output	
HandleMissing InputHow	Indicate how to handle missing values in input, one of:	KeepMissing
	KeepMissing – leave data missing Repeat – repeat last non-missing value SetToZero – set values to 0	
	The missing data is handled on input and the replacement value, if any, is applied to input and used for calculations just as if it was the actual value. The following cases do not use this parameter:	
	 Irregular data Day and Month input converted to ACCM and MEAN. 	

Several example command files follow. The following commands creates a Day ACCM time series from a Month ACCM time series:

```
0109.NOAA.Precip.Day~HydroBase
ChangeInterval(Alias="0109Month",TSList=AllMatchingTSID,
TSID="0109.NOAA.Precip.Day",NewInterval=Month,OldTimeScale=ACCM,NewTimeScale=ACCM)
```

The following commands create a 6Hour INST time series from a Day MEAN time series:

```
NewPatternTimeSeries(Alias="DayMEAN",NewTSID="ts1..SQME.Day",Description="Test data",
SetStart="2006-12-01",SetEnd="2007-01-31",
Units="CMSD",PatternValues="20,30,55,40,30,40,50,45,45,80,80,80,80")
ChangeInterval(Alias="6HourINST",TSID="DayMEAN",NewInterval=6Hour,OldTimeScale=MEAN,
NewTimeScale=INST,NewDataType=CMS)
```

The following commands create a Day MEAN time series from a 6Hour INST time series:

```
NewPatternTimeSeries(Alias="6HourInst",NewTSID="ts2..Flow.6Hour",IrregularInterval=6Hour,
    Description="Test data",SetStart="2006-12-15 12",SetEnd="2007-01-29 00",
    Units="CFS",PatternValues="20,23,56,62,35,42")
ChangeInterval(Alias="DayMean2",TSID="6HourInst",NewInterval=Day,OldTimeScale=INST,
    NewTimeScale=MEAN,HandleEndpointsHow=IncludeFirstOnly)
```

The following commands create a 3Hour INST time series from an Irregular (1Hour) INST time series:

```
NewPatternTimeSeries(Alias="IrregularINST",NewTSID="tsl..Temp.Irregular",IrregularInterval=1Hour,
    Description="Test data",SetStart="2006-12-15 00",SetEnd="2007-01-31 23",Units="DEGF",
    PatternValues="20,23,-999,45,-999,-999,56,62,0,-3")
ChangeInterval(Alias="3HourINST",TSID="IrregularINST",NewInterval=3Hour,OldTimeScale=INST,
    NewTimeScale=INST)
```

This page is intentionally blank.

Command Reference: ChangePeriod()

Change period of record for time series

Version 09.10.01, 2010-11-18

The ChangePeriod() command changes the period for the given time series, for example to extend the time series. A longer period will be filled with missing values.

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

👌 Edit ChangePeriod() Comm	and	X			
Change the time series period. The time series to process are indicated using the TS list. If TS list is "AllTS", pick a single time series, or enter a wildcard time series identifier pattern. Specify period start and end date/times using a precision consistent with the data interval.					
TS list:	Alits	Optional - indicates the time series to process (default=AllTS).			
TSID (for TSList=AllMatchingTSID);		▼			
EnsembleID (for TSList=EnsembleID):		~			
New start date/time:	1900-01	Optional - specify to change start date/time.			
New end date/time:	Optional - specify to change end date/time.				
Command:	ChangePeriod(TSList=AllTS,NewStart="1900-01") d:				
Cancel OK					

ChangePeriod() Command Editor

The command syntax is as follows:

```
ChangePeriod(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
TSList	 Indicates the list of time series to be processed, one of: AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will be modified. AllTS – all time series before the command. EnsembleID – all time series in the ensemble will be modified. FirstMatchingTSID – the first time series that matches the TSID (single TSID or TSID with wildcards) will be modified. LastMatchingTSID – the last time series that matches the TSID (single TSID or TSID with wildcards) will be modified. SelectedTS – the time series are those selected with the SelectTimeSeries() command. 	AllTS
TSID	The time series identifier or alias for the time series to be modified, using the * wildcard character to match multiple time series.	Required if TSList=*TSID.
EnsembleID	The ensemble to be modified, if	Required if
NewStart	The new period start, specified to precision that matches the time series data interval.	Start will remain the same.
NewEnd	The new period end, specified to precision that matches the time series data interval.	End will remain the same.

A sample command file to change the period of a time series from the State of Colorado's HydroBase is as follows:

```
# 08236000 - ALAMOSA RIVER ABOVE TERRACE RESERVOIR
08236000.DWR.Streamflow.Month~HydroBase
ChangePeriod(TSList=AllTS,NewStart="1900-01")
```

Command Reference: CheckTimeSeries()

Check time series data values against criteria and optionally take action

Version 10.03.00, 2011-12-19

The CheckTimeSeries() command checks time series data values against criteria, for example to identify missing, erroneous, or extreme data values. A warning is generated for each match and time series values optionally can be flagged, which allows annotation on graphs and reports. Values that meet the check criteria also can be removed (if irregular interval), or set to missing. The WriteCheckFile() command can be used to write a summary of the warnings. The CheckTimeSeriesStatistic() command checks a statistic for the entire time series (e.g., missing value count). See also the Delta() command, which creates new time series as the change between each value – this command may be necessary in cases where data periodically reset to a starting value, prior to using a performing a Change> check, for example.

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

👌 Edit CheckTimeSeries() Command 🛛 🛛 🔀				×	
Check time series data values for critic	Check time series data values for critical values (see also the CheckTimeSeriesStatistic() command).				
A warning will be generated for each o	ase where a value matc	hes the sp	ecified condition(s).		
Use the WriteCheckFile() command to	save the results of all ch	necks.			
Specify dates with precision appropriat	te for the data, use blan	nk for all av	ailable data, OutputStart, or OutputEnd.		
TS list:	AllMatchingTSID		Optional - indicates the time series to process (default=AlITS).		
TSID (for TSList=AllMatchingTSID):	ts2			*	
EnsembleID (for TSList=EnsembleID):				\sim	
Check criteria:	Missing	*	Required - may require other parameters.		
Value1:			Optional - minimum (or only) value to check.		
Value2:			Optional - maximum value in range, or other input to check.		
Analysis start:			Optional - analysis start date/time (default=full time series peri	od).	
Analysis end:			Optional - analysis end date/time (default=full time series perio	d).	
Problem type:			Optional - problem type to use in output (default=check criteria	a).	
Maximum warnings:			Optional - maximum # of warnings/time series (default=no limit).	
Flag:			Optional - flag to mark detected values.		
Flag description:			Optional - description for flag.		
Action:	Remove	*	Optional - action for matched values (default=no action).		
	CheckTimeSerie	s(TSLi	st=AllMatchingTSID,TSID="ts2",CheckCrit	5 -	
Command:	eria="Missing"	,Actio	n=Remove)		
Cancel OK					
ChackTimeSeries					

CheckTimeSeries() Command Editor

The command syntax is as follows:

CheckTimeSeries(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of:	AllTS
	• AllMatchingTSID – all time series that match	
	the TSID (single TSID or TSID with wildcards)	
	will be processed.	
	• AllTS – all time series before the command will be	
	processed.	
	• EnsembleID – all time series in the ensemble will be processed	
	• FirstMatchingTSID - the first time series that	
	matches the TSID (single TSID or TSID with	
	wildcards) will be processed	
	 LastMatchingTSID – the last time series that 	
	matches the TSID (single TSID or TSID with	
	wildcards) will be processed.	
	• SelectedTS – the time series selected with the	
	SelectTimeSeries() command will be	
	processed.	
TSID	The time series identifier or alias for the time series to	Required if
	be processed, using the * wildcard character to match	TSList=*TSID.
	multiple time series.	
EnsembleID	The ensemble to be modified, if processing an ensemble.	Required if
		TSLIST=
CheckCriteria	The criteria that is checked one of the following	None must be
CHECKCIICEIIa	Missing values are skipped except for cases where the	specified
	statistic is specific to missing values.	specificu.
	• AbsChange> – check for absolute change from	
	one value to the next value > Value1	
	 AbsChangePercent> – check for absolute 	
	change in percent from one value to the next value >	
	Valuel.	
	• Change> - check for change > Value1.	
	• Change< - check for change < Value1.	
	 InRange – check for value >= Value1 and <= 	
	Value2.	
	 OutOfRange – check for value < Value1 or > 	
	Value2.	
	• Missing – check for missing values.	
	• Repeat - check for Value1 repeating values (i.e.,	
	if Value1=2, then the check will detect 2 adjacent	
	values that are the same). If the flag or action are	

Parameter	Description	Default
	specified, values Value1+ in the sequence are	
	modified (i.e., if Value1=2, the 2 nd and subsequent	
	repeating values will be modified by the action).	
	• <- check for values < Value1.	
	• <= - check for values <= Value1.	
	• > - check for values > Value1.	
	• >= - check for values >= Value1.	
	• == - check for values equal to Value1.	
Valuel	A parameter that is used for specific CheckCriteria	
	values.	
Value2	A parameter that is used for specific CheckCriteria	
	values.	
AnalysisStart	The date/time to start analyzing data.	Analyze full period.
AnalysisEnd	The date/time to end analyzing data.	Analyze full period.
ProblemType	The problem type that will be shown in warning	CheckCriteria
	messages.	
MaxWarnings	The maximum number of warnings to list for each time	List all warnings.
	series, useful if analysis results in many warnings.	
Flag	A string to use for a flag on values that are detected	No flag.
	during the check, which will be shown in the HTML	
	summary report.	
FlagDesc	Description for the flag.	No description.
Action	Action to take for matched values, in addition to	No action is taken.
	generating warnings:	
	• Remove – remove the values. For irregular interval	
	time series the values will be removed. For regular	
	interval time series the values will be set to missing.	
	• SetMissing – set the values to missing.	

This page is intentionally blank.

Command Reference: CheckTimeSeriesStatistic()

Check time series statistic against criteria

The CheckTimeSeriesStatistic() command checks a time series statistic against criteria, for example to perform quality control using full-period statistics. This command is essentially a combination of the CalculateTimeSeriesStatistic() command with features similar to the CheckTimeSeries() command; however, the latter checks individual data values and this command checks a statistic computed from the entire time series. The WriteCheckFile() command can be used to write a summary of the warnings.

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

Edit CheckTimeSeriesStatis	stic() Command			
Check time series statistic for critical values (see also the CheckTimeSeries() command, which checks data values).				
A warning will be generated if the stat	istic matches the specified condition(s).			
use the write_neck-lie() command to save the results or all checks. Specify dates with precision appropriate for the data, use blank for all available data, OutputStart, or OutputEnd,				
Time series to process				
TS list	: AllMatchingTSID Optional - indicates the time series to process (default=AllTS).			
TSID (for TSList=AllMatchingTSID)	: 516122-precip-year-level3			
EnsembleID (for TSList=EnsembleID)				
ſ ^{Statistic}				
Statistic to calculate: Count	Required - may require other parameters.			
Statistic value1:	Optional - may be needed as input to calculate statistic.			
Statistic value2:	Optional - may be needed as input to calculate statistic.			
Statistic value3:	Optional - may be needed as input to calculate statistic.			
Analysis start:	Optional - analysis start date/time (default=full time series period).			
Analysis end:	Optional - analysis end date/time (default=full time series period).			
Table ID for output: InclusionChee	ks Optional - if statistic should be saved in table.			
Table TSID column: Alias	Required if using table - column name for TSID.			
Format of TSID: %A	Insert: Select Specifier 💙 Optional - use %L for location, etc. (default=alias or TSID).			
Table statistic column: Count	Required if using table - column name for statistic.			
Check and actions				
Check criteria: <	💌 Required - may require other parameters.			
Check value1: 15	Optional - minimum (or only) statistic value to check.			
Check value2:	Optional - maximum value in range, or other statistic value to check.			
Problem type:	Optional - problem type to use in output (default=Statistic-CheckCriteria).			
If criteria met?: Ignore ⊻	Optional - should warning/failure be generated (default=Warn).			
Property name: PeriodLevel3	Optional - name of property to set when criteria are met.			
Property value: No	Optional - value of property to set when criteria are met.			
CheckTimeSeriesS	tatistic(TSList=AllMatchingTSID,TSID="516122-precip-year-level			
Command: 3", Statistic="Count", TableID="InclusionChecks", TableTSIDColumn="Alias", TableTS				
IDFormat="%A", Ta	ableStatisticColumn="Count",CheckCriteria="<",CheckValue1=15,If			
Criteriamet=ignore, PropertyName="PeriodLevel3", PropertyValue="No")				
Cancel OK				
	ChackTimeSeriesStatistic			

CheckTimeSeriesStatistic() Command Editor

The command syntax is as follows:

CheckTimeSeriesStatistic(Parameter=Value,...)

Parameter Description Default TSList Allts Indicates the list of time series to be processed, one of: • AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will be processed. AllTS – all time series before the command will • be processed. EnsembleID – all time series in the ensemble will be processed. FirstMatchingTSID - the first time series that matches the TSID (single TSID or TSID) with wildcards) will be processed. LastMatchingTSID – the last time series that matches the TSID (single TSID or TSID with wildcards) will be processed. SelectedTS – the time series selected with the • SelectTimeSeries() command will be processed. TSID The time series identifier or alias for the time series to Required if be processed, using the * wildcard character to match TSList=*TSID. multiple time series. EnsembleID The ensemble to be modified, if processing an Required if TSList= ensemble. EnsembleID. Statistic Statistic to compute. Refer to the None – must be CalculateTimeSeriesStatistic() specified. command documentation. StatisticValuel Input data required by the statistic. Currently the See the Calculate dialog does not check the value for correctness – it is TimeSeries Statistic() checked when the statistic is computed. command documentation. StatisticValue2 Input data required by the statistic. Currently the to the Calculate dialog does not check the value for correctness – it is TimeSeries Statistic() checked when the statistic is computed. command documentation. StatisticValue3 Input data required by the statistic. Currently the to the Calculate TimeSeries dialog does not check the value for correctness - it is Statistic() checked when the statistic is computed. command documentation. AnalysisStart The date/time to start analyzing data. Full period is analyzed. AnalysisEnd Full period is The date/time to end analyzing data. analyzed. TableID Identifier for table that receives the statistic. Optional – table

Command Parameters

output is not

Parameter	Description	Default
	•	required.
TableTSIDColumn	Table column name that is used to look up the time series. If a matching TSID is not found, a row will be added to the table. If a TSID is found, the statistic cell value for the time series is modified.	Optional – table output is not required.
TableTSIDFormat	The specification to format the time series identifier to insert into the TSID column. Use the format choices and other characters to define a unique identifier.	Time series alias if available, or the time series identifier.
TableStatistic Column	Table column name to receive the statistic value. If not found in the table, a new column is added automatically.	Optional – table output is not required.
CheckCriteria	 The criteria that is checked, one of: InRange - check for value >= Value1 and <= Value2. OutOfRange - check for value < Value1 or > Value2. < - check for values < CheckValue1. <= - check for values <= CheckValue1. > - check for values > CheckValue1. >= - check for values >= CheckValue1. = - check for values >= CheckValue1. = - check for values and the checkValue1. 	None – must be specified.
CheckValue1	A parameter that is used for specific CheckCriteria values.	
CheckValue2	A parameter that is used for specific CheckCriteria values, currently only needed for InRange and OutOfRange criteria.	
ProblemType	The problem type that will be shown in warning messages.	Statistic- CheckCriteria
IfCriteriaMet	 Indicate whether to set the command status if the statistic meets the criteria, one of: Ignore - do not set the command status Warn - set the command status to Warning Fail - set the command status to Failure 	The command status will not be changed.
PropertyName	If the statistic meets the criteria, set the property identified by PropertyName to PropertyValue.	No property is set.
PropertyValue	If the statistic meets the criteria, set the property identified by PropertyName to PropertyValue.	No property is set.

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.

\delta 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 n	names be relative to the working directory, which is:			
C:\Develop\TSTool_Source	eBuild\TSTool\test\regression\commands\general\CompareFiles			
First file to compare:	Data/A1.txt	Browse		
Second file to compare:	Data/B1.txt	Browse		
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)			
Add Working Directory (File 1) Add Working Directory (File 2) Cancel OK				

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	None – the file name
	double quotes to protect whitespace and special	is required.
	characters. Global properties can be used with the	
TroutEileo	S{Propercy} syntax.	None the file name
InputFIlez	in double quotes to protect whitespace and special	is required
	characters. Global properties can be used with the	is required.
	\${Property} syntax	
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	False
	and trailing whitespace characters (spaces and tabs)	
	before doing the comparison. If False, then	
	whitespace is retained for the comparison.	
AllowedDiff	The number of lines allowed to be different, when	0
	checking for differences. This is useful, for example,	
	when a non-comment line contains the date/time when	
	the file was generated.	
liDifferent	Indicates the action to be taken if the files are different:	Do not generate a
	• Ignore – do not generate warning	warning if the files
	• Warn – generate a warning message	Differences are
	• Fail – generate a failure message	printed to the log
		file.
IfSame	Indicates the action to be taken if the files are the same:	Do not generate a
	• Ignore – do not generate warning	warning if the files
	• Warn – generate a warning message	are the same.
	• Fail – generate a failure message	

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)
Command Reference: CompareTables()

Compare tables Version 10.03.00, 2012-01-07

The CompareTables() command compares columns from two tables, saving the results in a new table. Comparisons are made using the data values formatted as strings based on the precision shown in tables. If the table was read with ReadFromDelimitedFile(), the precision for floating point numbers is set based on the largest number of digits after the decimal encountered in the input. Optionally, a precision and tolerance can be specified to control the comparison of floating point values. Values that are the same are shown in the new table without modification. Values that are different result in both table values being shown (as strings) to allow comparison. The command also allows the comparison table to be output to an HTML file, in which case different values are shown as red.

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

Edit CompareTables() Command					
This command compares two tables and optionally creates a new comparison table. By default, all columns (and rows) from the specified tables are compared; however, the columns to compare can be specified. The table is compared by formatting cell values as strings. If necessary, specify precision and tolerance for floating point comparisons. The results table, if written as HTML, indicates differences as colored cells (the simple table view will not color the differences).					
Table1 ID:	table1 💌	Required - first table to compare.			
Table2 ID:	table2 💌	Required - second table to compare.			
Table 1 columns to compare:		Optional - default is to compare all.			
Table 2 columns to compare:		Optional - default is to compare all.			
Precision:	2	Optional - digits after decimal to compare (default=use precision set for column).			
Tolerance:	.2	Optional - tolerance(s) to indicate difference (e.g., .01, .1, default=exact comparison).			
Allowed # of different values:	Optional - when checking for differences (default=0)				
New table ID:	comparison	comparison Optional - unique identifier for the comparison table.			
Output file to write:	<pre>[Fest_CompareTables_Differen</pre>	itTables_Precision_Tolerance_IfDifferent=Warn_out.html Browse			
Action if different:	Warn 💌	Optional - action if tables are different (default=Ignore).			
Action if same:	Optional - action if tables are the same (default=Ignore).				
Command:	CompareTables(Table1ID="table1",Table2ID="table2",Precision=2,Toleran ce=.2,NewTableID="comparison",OutputFile="Results/Test_CompareTables_ DifferentTables_Precision_Tolerance_IfDifferentWarn_out.html",IfDiffe rent=Warn)				
Remove Working Directory Cancel OK					

CompareTables() Command Editor

The command syntax is as follows:

CompareTables(Parameter=Value,...)

CompareTables

Parameter	Description	Default
TableID1	The identifier for the first table to be compared.	None – must be
		specified.
TableID2	The identifier for the second table to be compared.	None – must be
		specified.
CompareColumns1	The names of columns to be compared from the first	All columns will be
	table, separated by columns.	compared.
CompareColumns2	The names of columns to be compared from the second	All columns will be
	table, separated by columns.	compared.
Precision	The number of digits after the decimal to consider	Format floating point
	when comparing floating point values. If values are	numbers as strings for
	different to the specified (or default) precision, both	comparison according
	values are shown in the comparison table.	to the table column
		precision.
Tolerance	A value indicating the allowed difference between	Floating point values
	floating point values. The tolerance should be	must exactly match,
	consistent with the precision (i.e., don't specify a	according to the
	coarse precision and fine tolerance). If the difference is	precision.
	less than the tolerance, the values will not be marked as	
	different.	-
AllowedDiff	The allowed number of differences before triggering a	0
	Warn/Fail message (see IfDifferent). A value >=	
	0 indicates that the number of differences must be the	
	same as the specified value. A negative value indicates	
	that the number of differences can be less than or equal	
	to the specified value. This parameter is useful for	
	constructing tests where a specified number of	
	differences is expected.	
NewTableID	The identifier for the new comparison table.	TablelID-
		Table21D-
		comparison
OutputFile	If specified, an HTML table will be created for the	No HTML output file
	comparison table, in which different values are	will be created.
TEDifferent	highlighted in red.	Demotore
IIDIIIerent	indicates the action to be taken if the tables are	Do not generate a
		warning if the tables
	• Ignore – do not generate warning	are different.
	• Warn – generate a warning message	
	Fail – generate a failure message	
IfSame	Indicates the action to be taken if the tables are the	Do not generate a
	same:	warning if the tables
	• Ignore – do not generate warning	are the same.
	• Warn – generate a warning message	
	• Fail – generate a failure message	

Command Reference: CompareTimeSeries()

Compare time series to find data value differences

Version 08.15.00, 2008-05-04

The CompareTimeSeries() command compares time series to determine data differences. Currently time series header information is NOT compared – only data values are compared. It is designed to process many time series in bulk fashion. For example, read commands can be used to read time series from two different versions of a database, or from two files. Time series to compare are determined by trying to match each available time series with another time series in the list (ignoring itself); consequently, the list of time series should contain only pairs of time series.

Time series that are matched by TSID location and/or data type are compared value by value, with the differences computed as the value from the second time series minus the value from the first time series. The values can be rounded based on a specified precision. It may be important to read each set of time series from files to ensure that final round off is consistent. The checks occur by comparing the difference to one or more specified tolerances. Differences and simple statistics are printed to the log file. Values that are different can optionally be tagged with a character flag, for use with the graphing package. Time series of the differences can optionally be created. A warning can be generated if a difference is detected, or if no differences are detected (see also the CompareFiles() and CompareTables() commands).

🌌 Edit CompareTimeSer	🛿 Edit CompareTimeSeries() command 🛛 🔀				
This command compares time se	his command compares time series. Currently all available time series are evaluated,				
comparing time series that have	the same time series ider	tifier location and/or data type.			
The command is useful for comp	aring two time series (e.g	., a simple test) or two similar lists of time series (e.g., two data files).			
Specify one or more tolerances,	separated by commas. D)ifferences greater than these values will be noted.			
Match location:	~	Optional - match location to find time series pair? (default=True).			
Match data type:	*	Optional - match data type to find time series pair? (default=False).			
Precision:		Optional - digits after decimal to compare (default=available digits are used).			
Tolerance:		Optional - tolerance(s) to indicate difference (e.g., .01, .1, default=exact comparison).			
Analysis period:		to			
Difference flag:		Optional - 1-character flag to use for values that are different.			
Create difference time series?:	×	Optional - create a time series TS1 - TS2? (default=False).			
Warn if different?:	×	Optional - generate a warning if different? (default=False).			
Warn if same?:	True 🔽	Optional - generate a warning if same? (default=False).			
	CompareTimeSeries(WarnIfSame=True)				
Command:					
Cancel OK					
		CompareTimeSerie			

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

CompareTimeSeries() Command Editor

The command syntax is as follows:

CompareTimeSeries(Parameter=Value,...)

Parameter	Description	Default
MatchLocation	Match the location part of time series identifiers when	True
	matching time series to compare.	
MatchDataType	Match the data type part of time series identifiers when	False
	matching time series to compare.	
Precision	When comparing data values, round the values to the	Compare the
	given precision. For example, a precision of 2 will round	available values
	to the hundredths place. This can be used to do	without rounding.
	comparisons on the lowest precision of the available time	
	series.	
Tolerance	Specify a comma-separated list of values. The difference	A tolerance of zero
	in the time series values will be compared to the	will be used to detect
	tolerances and messages printed to the log file.	differences.
AnalysisStart	The starting date/time to analyze for differences. Specify	Analyze all available
	a date/time of appropriate precision for the time series or	data.
	OutputStart to use the output start.	
AnalysisEnd	The ending date/time to analyze for differences. Specify	Analyze all available
	a date/time of appropriate precision for the time series or	data.
	OutputEnd to use the output end.	
DiffFlag	Specify as a single character to append a flag to the data	Do not flag data.
	flags for the time series. Each value that is different is	
	flagged in both time series that are compared. The flag	
	can be displayed by the graphing package. This is useful	
	for verification processes. New time series will be	
	created with the original identifier preceded by Diff	
CreateDiffTS	Indicate whether a time series should be created	False
	containing the differences between time series. This is	
	useful to visually evaluate the differences and process	
	the results with other commands.	
WarnIfDifferent	If True and at least one difference is detected, a warning	Do not generate a
	will be generated by the command, which will result in	warning if time
	software like TSTool displaying a warning. If False,	series are different.
	only status messages are written to the log file. The	Differences are
	warning is useful if it is critical to detect any change in	printed to the log
	the time series.	file.
WarnIfSame	If True and no differences are detected, a warning will	Do not generate a
	be generated by the command, which will result in	warning if time
	software like TSTool displaying a warning. If False,	series are the same.
	only status messages are written to the log file. The	
	warning is useful if it is critical to detect that time series	
	are the same.	

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

```
# Example to compare files. Since they are different, a warning will be generated.
ReadDateValue(InputFile="RawData1.dv")
ReadDateValue(InputFile="RawData1Scaled.dv")
CompareTimeSeries(Precision=2,WarnIfDifferent=True)
```

The following example compares matching time series for the full available period, doing checks for several tolerances:

```
CompareTimeSeries(Precision=2,Tolerance="0,.1,.5,1",DiffFlag="x")
```

The following example compares data only within the output period, as specified by the SetOutputPeriod() command:

```
CompareTimeSeries(Precision=2,Tolerance="0,.1,.5,1",
AnalysisStart="OutputStart",AnalysisEnd="OutputEnd",DiffFlag="x")
```

This page is intentionally blank.

Command Reference: ComputeErrorTimeSeries()

Compute the error between time series and create new time series for the results Version 10.00.01, 2011-05-12

The ComputeErrorTimeSeries() command computes the error between two time series as absolute value or percent, creating a new time series for each pair of time series that is compared. This is useful for comparing observed and simulated time series. The time series that are created have the simulated time series' metadata but an alias can be assigned. The command can be used to process multiple pairs of time series, each determined using the appropriate *TSList parameter.

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

\delta Edit ComputeErrorTimeSeries() Command 🛛 🔀					
Compute the error between simulated time series and observed time series, and generate time series of the specified error measure. This command is useful for calibrating models and evaluating predictions after observed data are available. If one observed time series is specified, it will be analyzed against all simulated time series. If multiple observed time series are specified (e.g., for ensembles), the same number of simulated time series must be specified.					
Observed TS list:	AllMatchingTSID 🛛 👻]			Optional - indicates the time series to process (default=AllTS).
Observed TSID (for ObservedTSList=AllMatchingTSID):	ts1				▼
Observed ensembleID (for ObservedTSList=EnsembleID):					×
Simulated TS List:	AllMatchingTSID Optional - indicates the time series to process (default=AllTS).				
Simulated TSID (for Simulated TSList=AllMatchingTSID):	ts2 💌				
$\label{eq:simulated} \texttt{EnsembleID} \ (\texttt{for Simulated TSList} {=} \texttt{EnsembleID});$					×
Error measure:	PercentError 💌				Required - indicates how to compute error.
Alias to assign:		Insert: -	Select Specifier	~	Required - use %L for location, etc.
Command:	ComputeErrorTimeSeries(ObservedTSList=AllMatchingTSID,ObservedTSID="ts1" ,SimulatedTSList=AllMatchingTSID,SimulatedTSID="ts2",SimulatedEnsembleID ="PercentError",ErrorMeasure=PercentError)				
Cancel OK					

ComputeErrorTimeSeries() Command Editor

The command syntax is as follows:

ComputeErrorTimeSeries(Parameter=Value,...)

Parameter	Description	Default
Observed	Indicates the list of observed time series to be processed,	Allts
TSList	one of:	
	• AllMatchingTSID – all time series that match the	
	TSID (single TSID or TSID with wildcards).	
	• AllTS – all time series before the command.	
	• EnsembleID – all time series in the ensemble.	
	• FirstMatchingTSID – the first time series that	
	matches the TSID (single TSID or TSID with	
	wildcards).	

Parameter	Description	Default
	 LastMatchingTSID - the last time series that matches the TSID (single TSID or TSID with wildcards). SelectedTS - the time series are those selected with the SelectTimeSeries() command. 	
Observed TSID	The time series identifier or alias for the observed time series, using the * wildcard character to match multiple time series.	Use when ObservedTSList= *MatchingTSID.
Observed EnsembleID	The observed ensemble to be compared, if processing an ensemble.	Use when ObservedTSList= EnsembleID.
Simulated TSList	Indicates how to determine the list of simulated time series (see the explanation of ObservedTSList).	Allts
Simulated TSID	The time series identifier or alias for the simulated time series (see the explanation of ObservedTSID).	Use when SimulatedTSList= *MatchingTSID.
Simulated EnsembleID	The ensemble identifier for the simulated time series (see the explanation of SimulatedEnsembleID).	Use when SimulateddTSList= EnsembleID
ErrorMeasure	 The error measure to compute, one of: PercentError – Simulated minus observed, divided by observed. AbsoluteError – not yet implemented. 	
Alias	The alias to assign to the time series, as a literal string or using the special formatting characters listed by the command editor. The alias is a short identifier used by other commands to locate time series for processing, as an alternative to the time series identifier (TSID).	Alias will not be assigned.

A sample command file is as follows (in this case using contrived data):

```
RemoveFile(InputFile="Results\Test_ComputeErrorTimeSeries_1_out.dv",WarnIfMissing=False)
NewPatternTimeSeries(Alias="ts1",NewTSID="ts1..test.Day",Description="Test data",
    SetStart="1950-01-01",SetEnd="1951-03-12",Units="CFS",PatternValues="5,10,12,13,75")
NewPatternTimeSeries(Alias="ts2",NewTSID="ts2..test.Day",Description="Test data",
    SetStart="1950-01-01",SetEnd="1951-03-12",Units="CFS",PatternValues="6,12,14,11.5,80")
ComputeErrorTimeSeries(ObservedTSList=AllMatchingTSID,ObservedTSID="ts1",
    SimulatedTSList=AllMatchingTSID,SimulatedTSID="ts2",ErrorMeasure=PercentError)
# Uncomment the following command to regenerate the expected results file.
# WriteDateValue(OutputFile="Results\Test_ComputeErrorTimeSeries_1_out.dv")
WriteDateValue(OutputFile="Results\Test_ComputeErrorTimeSeries_1_out.dv",
    InputFile2="ExpectedResults\Test_ComputeErrorTimeSeries_1_out.dv",WarnIfDifferent=True)
```

Command Reference: ConvertDataUnits()

Convert time series data units

Version 09.10.01, 2010-11-18

The ConvertDataUnits() command converts the data units for a time series (e.g., before output to a file). Some read and write commands also may allow units to be converted.

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

S Edit ConvertDataUnits() Command					
Convert the data units of the selected time series to new data units. The old and new data units must have the same dimension (e.g., both are length). However, the dimension is not checked until time series are actually processed. If desired units are not recognized, try using the Scale() command.					
TS list:	AllMatchingTSID 🛛 😽		Optional - indicates the time series to process (default=AllTS).		
TSID (for TSList=AllMatchingTSID):	08236000.DWR.Streamflow.Month		▼		
EnsembleID (for TSList=EnsembleID):			×		
Dimension:	L3 - L3	*	Select the dimension to list corresponding units.		
New data units:	CFSD - CUBIC FOOT PER SECOND-DAY	4	Required - new units for data.		
Command:	ConvertDataUnits(TSList=AllMatchingTSID,TSID="08236000.DWR.Stre amflow.Month",NewUnits="CFSD")				
Cancel OK					
			ConvertDataUnits		

ConvertDataUnits() Command Editor

The *Dimension* choice should be selected to narrow the list of available units to the appropriate dimension. Next, select the *New Data Units* for the time series. The list of available data units is taken from the information described in the TSTool *DATAUNIT* file (see the TSTool **Installation and Configuration Appendix** for more information). If desired units are not available, contact the TSTool developers to suggest adding units to the *DATAUNIT* file or edit the command manually after initial creation. See also the TSTool *View... Data Units* menu to view the current data units.

The dialog cannot display the current units for the time series because the units are not available until time series are actually processed – commands are edited before processing.

The command syntax is as follows:

ConvertDataUnits(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSList	 Indicates the list of time series to be processed, one of: AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will be modified. AllTS – all time series before the command. EnsembleID – all time series in the ensemble will be modified. LastMatchingTSID – the last time series that matches the TSID (single TSID or TSID with wildcards) will be modified. SelectedTS – the time series are those selected with the SelectTimeSeries() command. 	AllTS
TSID	The time series identifier or alias for the time series to be modified, using the * wildcard character to match multiple time series.	TSID or EnsembleID must be specified if identifiers are being matched.
EnsembleID	The ensemble to be modified, if processing an ensemble.	TSID or EnsembleID must be specified if identifiers are being matched.
NewUnits	The new data units.	None – must be specified.

A sample commands file to convert the units of a time series from the State of Colorado's HydroBase is as follows:

08236000 - ALAMOSA RIVER ABOVE TERRACE RESERVOIR 08236000.DWR.Streamflow.Month~HydroBase ConvertDataUnits(TSList=AllMatchingTSID, TSID="08236000.DWR.Streamflow.Month",NewUnits="CFSD")

Command Reference: Copy()

Create a new time series as a copy of an existing time series

The Copy() command creates a copy of an existing time series, assigning an alias to the result. The copy is an exact copy except that the alias is different (the TSID must also specified and should be defined to be unique). The alias can then be used for further time series manipulation. A copy of a time series is useful when data filling or other manipulation will occur and time series that is unique from the original is needed. For example, if adding two time series, a copy of one time series can be made, and the second time series added to the copy – this ensures that there is not confusion with the original time series. Parameters are available to control how much of the original data are copied.

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

Edit Copy() C	Command	×
Make a copy of a tim The copy is exactly t Specify new time ser	e series, giving the copy an alias. :he same and can be referenced by its alias in other comman :ies identifier (TSID) information for the copy to avoid errors	ds. with the copy being mistaken for the original.
Time series to copy:	08223000.DWR.Streamflow.Month	*
New time series ID:	08223000.DWR.Streamflow.Month.Fill ed	Required - specify to avoid confusion with TSID from original time series.
Alias to assign:	Filled Insert: Select Specifier 👻	Required - use %L for location, etc.
Copy data flags?:	*	Optional - should data flags be copied (default=True).
Copy history?:	×	Optional - should history be copied (default=True).
Command:	Copy(Alias="Filled",TSID="08223000. .Streamflow.Month.Filled")	DWR.Streamflow.Month",NewTSID="08223000.DWR
	Cancel	ок
		Cop

Copy() Command Editor

The command syntax is as follows:

TS Alias = Copy(Parameter=Value,...)

The following older command syntax is updated to the above syntax when a command file is read:

TS Alias = Copy(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSID	The time series identifier or alias of the	None – must be specified.
	time series to copy. The time series will	
	be found by searching backwards from	
	the copy command.	
NewTSID	A new time series identifier to assign to	Copy the original time series
	the copy. This is useful to avoid	TSID. If NewTSID is specified
	confusion with the original time series.	but does not have a valid interval,
	Use the <i>Edit</i> button to edit the time series	copy the interval from TSID.
	identifier parts. The data interval must	The default cannot be determined
	match that of the original time series.	if an alias is used for the input
		time series.
Alias	The alias to assign to the time series, as a	None – must be specified.
	literal string or using the special	
	formatting characters listed by the	
	command editor. The alias is a short	
	identifier used by other commands to	
	locate time series for processing, as an	
	alternative to the time series identifier	
	(TSID).	
CopyDataFlags	Indicates whether data flags are copied.	True
	Specify as False or True.	
CopyHistory	Indicates whether the time series	True
	manipulation history is copied. Specify	
	as False or True.	

A sample command file to read a time series and make a copy is as follows:

Command Reference: CopyEnsemble()

Create a new ensemble as a copy of an ensemble

Version 08.15.00, 2008-05-04

The CopyEnsemble() command creates a copy of an ensemble, copying all time series in the ensemble and assigning a new identifier to the result. The copy is an exact copy except that the ensemble identifier is different (the TSIDs for each ensemble time series should also specified to be unique).

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

Edit CopyEnsemble() command				
Make a copy of a time seri	ies ensemble, giving the copy a new identifier.			
The copy is exactly the sa	ame and can be referenced by its identifier in o	ner commands.		
Because time series in the	e ensemble are copies of time series from the o	iginal ensemble, the	Ensemble ID should be used for processing time series.	
Optionally, specify new tir	ne series identifier (TSID) information for the tin	e series in the copy	location, source, data type, and/or scenario.	
This is highly recommende	ed if there is any chance that the copy will be n	staken for the origin	al.	
New ensemble identifier:	Ensemble_2			
New ensemble name:	Test ensemble 2		Optional name for copy.	
Ensemble to copy:	Ensemble_1			
New time series ID parts:	09019500.USGS.Streamflowcopy		Specify to avoid confusion with TSID from original TS.	
			Edit Clear	
	CopyEnsemble (NewEnsembleID="Ense	mble_2",NewEns	embleName="Test ensemble	
Command:	2",NewTSID="09019500.USGS.Stream	flowcopy",En	sembleID="Ensemble_1")	
Continuity.				
Cancel OK				
			ConvEnsemble	

CopyEnsemble() Command Editor

The command syntax is as follows:

```
CopyEnsemble(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
EnsembleID	The ensemble to copy.	None – must be specified.
NewEnsembleID	The ensemble identifier for the new	None – must be specified.
	ensemble	
NewEnsembleName	The name for the new ensemble.	Blank.
NewTSID	A new time series identifier to assign to	Copy the original time series
	time series in the new ensemble. This is	TSID.
	useful to avoid confusion with the	
	original time series. Use the <i>Edit</i> button	
	to edit the time series identifier parts.	
	The data interval and sequence number	
	will be determined from the original time	
	series.	

A sample commands file to read a time series from the State of Colorado's HydroBase, create an ensemble from the time series, and make a copy is as follows:

```
# 09019500 - COLORADO RIVER NEAR GRANBY
09019500.USGS.Streamflow.Day~HydroBase
CreateEnsemble(TSID="09019500.USGS.Streamflow.Day",
   TraceLength=1Year,EnsembleID="Ensemble_1",EnsembleName="Test
Ensemble",ReferenceDate="2008-01-01",ShiftDataHow=ShiftToReference)
CopyEnsemble(NewEnsembleID="Ensemble_2",
   NewEnsembleName="Test ensemble 2",
   NewEnsembleName="Test ensemble 2",
   NewTSID="09019500.USGS.Streamflow..copy",EnsembleID="Ensemble_1")
```

Command Reference: CopyTable()

Create a table as a (partial) copy of a table

Version 10.21.00, 2013-05-17

The CopyTable() command copies all or a subset of the columns from one table to create a new table. For example, this is useful to create one-column lists that can be used to expand template files with the ExpandTemplateFile() command, or to create a subset of columns to output to a file or write to a database.

The following dialog is used to edit the command and illustrates the syntax of the command (in this case illustrating how values in a column named LocationID are copied to a new table).

The second second second	Contraction for constants of safety with	
For example, single colu	new table by copying another table. mn tables can be created from a larger table to use as a lis	t with a template.
Table ID:	Table 1	Required - original table.
New table ID:	Table 1Copy	Required - unique identifier for the new table.
Column names to copy:	DateTime,String	Optional - names of columns to copy (default=copy all).
Distinct column names:		Optional - names of columns to filter distinct combinations (default=copy all)
Column map:		Optional - to change names (default=names are same).
Column filters:	String:first day	Optional - filter output by matching column pattern (default=copy all rows).
Command:	CopyTable(TableID="Table1",NewTable g",ColumnFilters="String:first day"] ID="Table1Copy", IncludeColumns="DateTime, Strin)
	Cancel	OK.

CopyTable() Command Editor

The command syntax is as follows:

CopyTable(Parameter=Value,...)

Parameter	Description	Default
TableID	The identifier for the original table.	None – must be specified.
NewTableID	The identifier for the new table.	None – must be specified.
IncludeColumns	The names of columns to copy, separated by commas.	Copy all of the columns.
DistinctColumns	The names of columns to copy, separated by commas. Only distinct values from the specified column will be copied. For example, if column A contains strings X, Y, Z, Y, C, the resulting distinct value column will have rows with X, Y, Z, C. Currently only one column name can be specified but in the future more than one column may be allowed.	Don't do a distinct comparison.
	This parameter overrides IncludeColumns.	
ColumnMap	The new names for the output columns, using syntax: OriginalColumn1:NewColumn1, OriginalColumn2:NewColumn2	The column names in the copy will be the same as in the original table.
ColumnFilters	Filters that limit the number of rows being copied, using the syntax: FilterColumn1:FilterPattern1, FilterColumn2:FilterPattern2 Patterns can use * to indicate wildcards for matches. Only string values can be checked (other data types are converted to strings for comparison). Comparisons are case-independent. All patterns must be matched in order to copy the row. In the future a command may be added to perform queries on tables, similar to SQL for databases.	No filtering.

Command Reference: CreateEnsembleFromOneTimeSeries()

Create a new ensemble from a single time series

Version 10.14.00, 2012-12-12

The CreateEnsembleFromOneTimeSeries() command creates an ensemble by splitting up a single time series into traces. For example, a historical time series can be split into 1-year overlapping traces that are shifted to start in the current year. The sequence number part of the time series identifier for each trace is set to the input starting year and will be shown as [Year] at the end of the time series identifier.

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

🗅 Edit CreateEnsembleFromOneTimeSeries() command 🛛 🛛 🔀				×
Create an ensemble of time series traces fro	Create an ensemble of time series traces from a single time series, for example to split the time series into overlapping historical traces.			
Each trace will start on the reference date a	nd will be as long as specified (Tra	ceLength).		
Each trace will have the properties of the or	iginal time series with sequence nu	mbers set to the	input year.	
Specify the period to limit the number of tra-	ces generated from the original tim	ie series.		
Specify the reference date using standard o	late formats to a precision appropr	iate for the data	, or use "CurrentToDay", etc., notation.	
If shifted, each trace will start on the refere	nce date (use to align time series f	for display and an	halysis).	
Ir NOT shirted, each trace will start on the r	ererence date, but year will vary v	with the data.		
Time series from which to create ensemble:	09019500.USGS.Streamflow.Day	7		<u> </u>
Input start:			Optional (default=full period).	
		Input end:	Optional (default=full period).	
Ensemble ID:	Ensemble_1]	Required - identifier for ensemble.	
Ensemble name:	Test Ensemble		Optional - name for output.	
Alias to assign:	Select Specifier 💉 =>		Optional - alias for each trace (default=%L_%z).	
Trace length:	1Year		Optional (default=1Year).	
Reference date:	2008-01-01		Optional (default=Jan 1 of first year).	
Output year type:	¥		Optional - causes sequence number to agree with year t	ype.
Shift data how?:	ShiftToReference 🐱		Optional (default=NoShift).	
Command:	CreateEnsembleFromOneTimeSeries(TSID="09019500.USGS.Streamflow.Day ",TraceLength=1Year,EnsembleID="Ensemble_1",EnsembleName="Test Ensemble",ReferenceDate="2008-01-01",ShiftDataHow=ShiftToReference)			
Cancel OK				

CreateEnsembleFromOneTimeSeries() Command Editor

The command syntax is as follows:

CreateEnsembleFromOneTimeSeries(Parameter=Value,...)

Parameter	Description	Default
TSID	The time series identifier or alias for the time series used to	None – must
	create the ensemble.	be specified.
InputStart	The date/time to start transferring data from the time series.	Use all data.
InputEnd	The date/time to end transferring data from the time series.	Use all data.
EnsembleID	The new ensemble identifier.	None – must
		be specified.
Ensemble	The name for the new ensemble.	Blank.
Name		0.70
Allas	The alias to assign to the time series, as a literal string or using	%L_%Z
	the special formatting characters listed by the command editor.	(location_
	The alias is a short identifier used by other commands to locate	sequence
	identifier (TSID).	Number)
TraceLength	An interval for the trace length (e.g., 1Year, #Month or,	1Year
	#Day).	
ReferenceDate	The reference date indicates the starting date for each trace.	January 1 of
	Each trace optionally can be shifted (see ShiftDataHow), in	the first year in
	which case the year in the ReferenceDate is used for the	the source time
	common starting date. The reference date can be one of:	series.
	• Blank, indicating that January 1 of the current year will be	
	used.	
	• A date/time string (use the format 01/01/YYYY or YYYY-	
	MM-DD).	
	• CurrentToYear, CurrentToMonth,	
	CurrentToDay,CurrentToHour,	
	CurrentToMinute, indicating the current date/time to	
	the specified precision.	
	• A Current * value +- an interval, for example:	
	CurrentToMinute - 7Day	
OutputYearType	The output year type for the ensemble traces. The only impact	Calendar
	from this parameter is that sequence number for the time series	
	will be set to the start of the output year. This is useful because	
	legends on graphs that use the sequence number (%z format	
	specifier) will use the appropriate year type. The	
	ReferenceDate should normally be specified as the first day	
	of the output year (e.g., ReferenceDate=2012-10-01 for	
	OutputYearType=Water).	
ShiftDataHow	Indicates whether the traces should be shifted. Possible values	NoShift
	arc.	
	• SILLUTORELELENCE – cach uace will be sinned to the reference date, resulting in overlapping time series	
	No Chieft plotting the traces will result in a total line that	
	• NOSHILL - plotting the traces will result in a total line that	
	be manipulated individually	
TraceLength ReferenceDate OutputYearType ShiftDataHow	 The anas to assign to the time series, as a interfal string of Using the special formatting characters listed by the command editor. The alias is a short identifier used by other commands to locate time series for processing, as an alternative to the time series identifier (TSID). An interval for the trace length (e.g., 1Year, #Month or, #Day). The reference date indicates the starting date for each trace. Each trace optionally can be shifted (see ShiftDataHow), in which case the year in the ReferenceDate is used for the common starting date. The reference date can be one of: Blank, indicating that January 1 of the current year will be used. A date/time string (use the format 01/01/YYYY or YYYY-MM-DD). CurrentToYear, CurrentToMonth, CurrentToMinute, indicating the current date/time to the specified precision. A Current* value +- an interval, for example: CurrentToMinute - 7Day The output year type for the ensemble traces. The only impact from this parameter is that sequence number (%z format specifier) will use the appropriate year type. The ReferenceDate should normally be specified as the first day of the output year (e.g., ReferenceDate=2012-10-01 for OutputYearType=Water). Indicates whether the traces should be shifted. Possible values are: NoShift - plotting the traces will result in a total line that matches the original time series, except that each trace can be manipulated individually. 	old_old (location_sequence Number) 1Year January 1 of the first year in the source time series. Calendar NoShift

A sample command file to read a time series from the State of Colorado's HydroBase and create an ensemble from the time series is as follows:

```
# 09019500 - COLORADO RIVER NEAR GRANBY
09019500.USGS.Streamflow.Day~HydroBase
CreateEnsembleFromOneTimeSeries(TSID="09019500.USGS.Streamflow.Day",
TraceLength=1Year,EnsembleID="Ensemble_1",EnsembleName="Test
Ensemble",ReferenceDate="2008-01-01",ShiftDataHow=ShiftToReference)
```

The following figure illustrates a graph of the resulting ensemble:



CreateEnsembleFromOneTimeSeries() Example Graph

This page is intentionally blank.

Command Reference: CreateFromList()

Create one or more time series from a file containing a list of identifiers

Version 10.21.00, 2013-05-17

See also the **ReadTimeSeriesFromTable()** command, which may replace this command in the future. The CreateFromList() command creates one or more time series using identifiers from a list file, an example of which is shown below:

```
# Example list file. Comments start with the # character.
# Column headings can be specified in the first non-comment row using quotes.
"Structure ID", "Structure Name"
500501,Ditch 501
500502,Ditch 502
# Invalid ID (see IfNotFound parameter)
509999,Ditch 9999
```

The command is typically used when reading time series from a database or binary file and can streamline processing in the following situations:

- A list of identifiers may have been generated from a database query and saved to a file.
- A list of identifiers may have been extracted from a model data set.

TSTool reads the list file and internally creates a list of time series identifiers. The time series are of the standard form:

Location.DataSource.DataType.Interval[.Scenario]~InputType[~InputName]

where the brackets indicate optional information. TSTool then queries each time series, which can be processed further.

Although it is possible to specify an input type that reads from files by also using the InputName, this is not generally recommended because the CreateFromList() command can only specify one input file name and the file will be reopened for each read. Instead, read commands for specific file formats should be used because these commands are typically optimized to read multiple time series from the files. In summary, the CreateFromList() command is useful with databases but performance may suffer when used with file input types.

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

Edit CreateFromLis	st() Command	×	
Create a list of time series	s from a list of location identifiers in	n a file.	
The information specified	below is used with the identifiers	to create time series identifiers,	
which are then used to re	ead the time series. The identifiers	are of the form:	
ID.DataSource.DataType	e.Interval.Scenario~InputType~Inpu	dName	
This command is useful for	or automating time series creation	where lists of identifiers are being processed.	
The list file can contain co	omment lines starting with #.		
It is recommended that the	e path to the file be specified using	g a relative path.	
The working directory is:	C:Develop\1S1ool_SourceBuild\1:	STooltestvregression/UserManualExamples\TestCases\CommandReference\CreateFromList	
List file to read:	Data\Diversions.txt	Browse	
ID column:	1 🔽	Required - the ID column in the list file (1+).	
Delimiter:		Optional - delimiter(s) between data columns (default is " ,").	
ID filter pattern:		Optional - IDs to use from list file (default is all). For example, use X*.	
Data source:	DWR	Optional or required depending on input type.	
Data type:	DivTotal	Optional or required depending on input type.	
Data interval:	Month	Required - for example, 5Minute, 6Hour, Day, Month, Year, or Irregular.	
Scenario:		Optional.	
Input type:	HydroBase	Required - needed to identify format of input (e.g., HydroBase).	
Input name:		Optional (e.g., use for file name for input type).	
If time series not found?:	Default 💌	Required - how to handle time series that are not found.	
Default units:		Optional - units when IfNotFound=Default.	
	CreateFromList(ListFile	="Data\Diversions.txt",IDCol=1,DataSource="DWR",DataType="DivT	
	otal", Interval="Month", InputType="HydroBase", IfNotFound=Default)		
Command:			
	,		
	Remove W	/orking Directory Cancel OK	
		CreateFromList	

CreateFromList() Command Editor

The command syntax is as follows:

```
CreateFromList(Parameter=Value, ...)
```

Parameter	Description	Default
ListFile	The name of the list file to read, surrounded	None – must be specified.
	by double quotes.	_
IDCol	The column $(1+)$ in the list file containing	1
	the location identifiers to use in time series	
	identifiers.	
Delim	The delimiter characters that separate	Comma
	columns in the list file. If a space is used as	
	the delimiter, surround with another	
	delimiter characters or a character that is	
	unlikely to be found so that the space is not	
	discarded as white space (e.g., "~ ~").	
ID	Indicate a pattern to filter the identifiers in	Process all identifiers.
	the list file. For example, use A* to only	

Parameter	Description	Default
	process identifiers in the list file that start	
	with A.	
DataSource	The data source in the time series identifier, appropriate for InputType. For example, if using the State of Colorado's HydroBase, USGS indicates that data are from the United States Geological Survey. See the input type appendices for more information on	May or may not be required, depending on the input type. Refer to the input type appendices.
	available data types.	
DataType	The data type in the time series identifier, as appropriate for InputType. For example, if using the State of Colorado's HydroBase, DivTotal is used for diversion totals. See the input type appendices for more information on available data types.	Usually required for an input type. Refer to the input type appendices.
Interval	Data interval in the time series identifier, using standard values such as 15Minute, 6Hour, Day, Month, Year.	None – must be specified.
Scenario	Scenario in the time series identifier.	Usually not required.
InputType	The input type in the time series identifier. For example, use HydroBase for the State of Colorado's HydroBase database. Refer to the input type appendices or the TSTool main GUI for options.	None – must be specified.
InputName	The input name in the time series identifier.	Typically only required if the input type requires a file name.
IfNotFound	 Indicates how to handle missing time series, one of: Warn – generate fatal warnings and do not include in output. Ignore – generate non-fatal warnings and do not include in output. Default – generate non-fatal warnings and create empty time series for those that could not be found. This requires that a SetOutputPeriod() command be used before the command to define the period for default time series. 	Warn
DefaultUnits	Default units when IfNotFound=Default.	Blank – no units.

A sample command file to process monthly diversion data from the State of Colorado's HydroBase database is as follows:

```
# Read monthly diversion total from HydroBase for the structures in the list
# file. The data source is set to DWR because data source is saved in
# HydroBase.
CreateFromList(ListFile="Data\Diversions.txt",IDCol=1,DataSource=DWR,
```

DataType=DivTotal, Interval=Month, InputType=HydroBase, IfNotFound=Default)

Command Reference: CreateRegressionTestCommandFile()

Create a command file to run software regression tests

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

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.
<pre>@expectedStatus Failure @expectedStatus Warning</pre>	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	The test is designed to work only on the specified platform and will
@os UNIX	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

Command # Comment Tags

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.

• Edit CreateRegressionTestCom	mandFile() command			
This command creates a regression test con	his command creates a regression test command file, for use in software testing.			
Test command files should follow documente	ed standards.			
A top-level folder is specified and will be sea	arched for command files matching the specified pat	tern.		
The resulting output command file will includ	le RunCommands() commands for each matched file	e, and can be independently loaded and run.		
A "setup" command file can also be included	l at the top of the output, for example to initialize d	latabase connections.		
It is recommended that file names be relativ	e to the working directory, which is:			
C:\Develop\TSTool_SourceBuild\TSTool\b	est\regression\TestSuites\commands_general\creat	te		
Folder to search for TSTool command files:	\\commands\general		Browse	
Command file to create:	\run\RunRegressionTest_commands_general_Inc	cludeOS=Windows.TSTool	Browse	
Setup command file:	Create_RunTestSuite_commands_general_Include	OS=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				
		CreateRegressionTest	CommandFile	

CreateRegressionTestCommandFile() Command Editor

The command syntax is as follows:

CreateRegressionTestCommandFile(Parameter=Value,...)

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

Parameter	Description	Default
	typically specifies False and additional commands	
	specify True.	
IncludeTestSuite	If *, all tests that match FilenamePattern and	* – include all test
	IncludeOS are included. If a test suite is specified,	cases.
	only include tests that have @testSuite tag values	
	that match a value in IncludeTestSuite. One or	
	more tags can be specified, separated by commas.	
IncludeOS	If *, all tests that match FilenamePattern and	* – include all test
	IncludeTestSuite are included. If an OS is	cases.
	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.	

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:

# Tile memorates	al lass				
# File generated	u Dy mom1 10 00	00 (0010 0	1 1 0 1		
# program:	TSTool 10.20.00 (2013-04-10)				
# user:	sam				
# date:	Sat Apr 20 1	3:36:05 MD1	2013		
# host:	AMAZON				
# directory:	C:\Develop\T	STool_Sourc	eBuild\TSToc	ol\test\regression\TestSuites\commands_general\run	
# command line:	TSTool				
# -home test/op	perational/CD	SS			
#					
# Command file :	regression te	st report f	rom StartRec	gressionTestResultsReport() and RunCommands()	
#		-			
# Explanation of	f columns:				
#					
# Num: count of	the tests				
# Enabled: blan	k if test ena	bled or FAI	SE if "#@ena	whed false in command file	
# Bup Time: rup	time in mill	isegonds	ion in lieene		
# Tegt Dagg/Fai	1.	19000108			
# The test at	1. totug bolow m	arr be DACC	or EATT (or	blank if disabled)	
# Ine test s	lacus below m	ay be PASS	OF FAIL (OF	Dialik II disabled).	
# A Lest WII.	I pass II the	Command 11	ie actual st	ally matches the expected status.	
# Disabled Le	ests are not	run and do	not count as	PASS OF FAIL.	
# Search for	"FAIL" LO II.	na taitea u	ests.		
# Commands Expe	cted Status:				
# Default is	assumed to b	e SUCCESS.			
# "#@expected	dStatus Warni	ng Failure"	comment in	command file overrides default.	
# Commands Actua	al Status:				
# The most se	evere status	(Success Wa	rning Failur	re) for each command file.	
#					
#	Test	Commands	Commands		
# R1	un Pass/	Expected	Actual		
# Num Enabled T	ime Fail	Status	Status	Command File	
#++		+	+	•+	
1	141 PASS	SUCCESS	SUCCESS	$\verb C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ARMA\Test_ARMA_Day.TSTool\general\ARMA\Test_ARMA_Day.TSTool\general\ARMA\Test_ARMA_Day.TSTool\general\ARMA\Test_ARMA_Day.TSTool\general\general\ARMA\Test_ARMA_Day.TSTool\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\general\ge$	
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	LINKNOWN		
C:\Develop\TST	ool SourceBui	ld\TSTool\t	est\rearest	on\commande\ceneral\WritePeclamationHDR\Test WritePeclamationHDR	
#+	+	+	+		
FATL count = 0.0.000					
DASS dount	- 17 100 000	9			
Disabled count	- 1, 100.000	0			
#	- +				
Total	- 19				
IULAI	- +0				

This page is intentionally blank.

Command Reference: Cumulate()

Convert time series data values to cumulative values

Version 10.12.00, 2012-07-25

The Cumulate() command converts a time series into cumulative values, which is useful for:

- comparing the cumulative trends of related time series (e.g., nearby gages or precipitation gages) and can serve as a substitute for the double-mass graph, which has difficulty handling missing data
- checking mass balance when routing time series (the cumulative values before and after routine will track closely)
- computing year-to-date totals such as cumulative precipitation

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

\delta Edit Cumulate() Command	X	
The selected time series will be converted to cumulative values over the period. The units remain the original. Specify a Reset value to reset the total for each year. The date/time should be specified to a precision matching the time series.			
Use a format MM for r The year's data will be	e set to missing if AllowingMissingCount and MinimumSampleSize criteria are not met.		
	TS list: AllTS	AllTS).	
TSID (for TSList=AllM	AatchingTSID);	~	
EnsembleID (for TSList=	=EnsembleID);	~	
Handle missi	ing data how?: CarryForwardIfMissing 🔜 Optional (default=SetMissingIfMissing).		
Parameters to reset v	value every year		
Reset date/time:	Optional - date/time on which to reset total (default=no reset).		
Reset value:	Optional - value for reset (default=0).		
Allow missing count:	Optional - number of missing values allowed in year (default=no limit).		
Minimum sample size:	Optional - minimum required sample size in year (default=no minimum).		
Command:			
	Cancel OK		

Cumulate() Command Editor

The command syntax is as follows:

```
Cumulate(Parameter=Value,...)
```

Parameter	Description	Default
TSList	Indicates the list of time series to be processed,	AllTS
	one of:	
	• AllMatchingTSID – all time series that	
	match the TSID (single TSID or TSID	
	with wildcards) will be modified.	
	• AllTS – all time series before the	
	command.	
	• EnsembleID – all time series in the	
	ensemble will be modified.	
	 LastMatchingTSID – the last time 	
	series that matches the TSID (single TSID	
	or TSID with wildcards) will be modified.	
	• SelectedTS – the time series are those	
	selected with the	
	SelectTimeSeries() command.	
TSID	The time series identifier or alias for the time	Required for
	series to be modified, using the * wildcard	TSList=*TSID.
	character to match multiple time series.	
EnsembleID	The ensemble to be modified, if processing an	Required for
	ensemble.	TSList=EnsembleID.
HandleMissingHow	Indicate how to handle missing data, one of:	SetMissingIfMissing
	 CarryForwardIfMissing_carry 	
	forward the last non-missing value	
	 SetMissingTfMissing - set the 	
	result to missing if the original value is	
	missing.	
	The only difference in output is that the period	
	of missing data will either be blank or a	
	horizontal line in graphs.	D
Reset	A date to the precision of the time series (e.g.,	Do not reset.
	01-01 for January 1 in a daily time series) that	
	the initial value, before beginning to cumulate	
	again Specifying the reset effectively defines	
	the first timestep in a new year, whether	
	calendar or some other year is being used for	
	the cumulative values. Use the format MM-DD,	
	MM-DD hh, or MM-DD hh:ss.	
ResetValue	When Reset is specified: the value to	0 (zero)
	initialize the total at the Reset date/time, one	· · · ·
	of:	
	• DataValue – the data value from the	
	original time series	
	• Number – a number to use for the reset	

Parameter	Description	Default
AllowMissingCount	When Reset is specified: the number of	No limit on the number of
	values allowed to be missing in a year. If more	missing values.
	values are missing, the entire year is set to	
	missing. The missing value count for the first	
	year includes the period from analysis start to	
	Reset. A partial year at the end of the	
	analysis period will not count as missing	
	beyond the analysis end.	
MinimumSampleSize	When Reset is specified: the minimum	No minimum sample size is
	number of non-missing values required in a	required.
	year to perform the computation. If fewer	
	values are in the sample, the entire year is set	
	to missing. The missing value count for the	
	first year includes the period from analysis	
	start to Reset. A partial year at the end of the	
	analysis period will result in the sample size	
	being less than the full year.	

A sample command file to cumulate times from the State of Colorado's HydroBase is as follows:

```
# 1458 - CENTER 4 SSW
1458.NOAA.Precip.Month~HydroBase
# 2184 - DEL NORTE 2 E
2184.NOAA.Precip.Month~HydroBase
Cumulate(TSList=AllTS,HandleMissingHow=CarryForwardIfMissing)
```

The following graph illustrates cumulative data for two precipitation gages in the same region, where missing data results in carrying forward the last known value.



Example Graph Showing Results of cumulate() Command

This page is intentionally blank.

Command Reference: Delta() Create new time series where values are the difference between each value in original time series

Version 9.07.00, 2010-08-05

The Delta() command creates a new time series from an input time series. The resulting values are computed as the difference between each value and the previous value. Consequently, the delta result is the change from the previous value. The CheckTimeSeries() command can be used to check time series for changes that exceed a threshold; however, the Delta() command handles the complexity of time series that reset to a new starting value – the output can be used in conjunction with CheckTimeSeries(). The Delta() command will create as many output time series as there are input time series.

The output value is simply the current value minus the previous value. The result is set to missing if this value cannot be computed due to missing values, or in cases where a transition across a reset has errors.

If the data do reset, then the expected trend should be specified to allow the ResetMin and ResetMax parameters to be properly interpreted. For example, if Trend=Increasing and a decrease is detected, it is assumed that the values have circled past the reset values. In this case the command will attempt to compute the change across the reset values. If this is not possible, then warnings will be generated and the result will be set to missing. Specific cases that are handled are:

- The previous value is out of range in this case the contribution from the out of range previous value is added to the delta and default flag value is assigned (see Flag parameter description). A warning will be generated.
- The current value is out of range in this case the difference will be decreased because the reset value has not be achieved. A warning will be generated.

The above special cases result in somewhat arbitrary difference values because the inputs do not conform to expected values. Out of range values indicate erroneous data that should be corrected before being used in further analysis.

Irregular-interval time series that result in differences not being computed will have missing values inserted at appropriate locations to maintain consistent data point spacing with the original data.

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

💧 Edit Delta() Command				X
Create new time series as a delta (difference) between the current value and the previous value.				
Use the ResetMax and ResetMin parameters for cumulative time series that periodically reset to a new starting value (will compute differences across resets).				
Specify dates with precision appropriate for the data or use blank for all available data.				
The new time series identifier is	deraulted to the	e ola, with -D	eica appended to the dat	a type (may allow specifying as a parameter in the future).
I S list:				Optional - Indicates the time series to process (derault=AIITS).
TSID (for TSList=AllMatchingTSID);				×
EnsembleID (for TSList=EnsembleID):				×
Expected trend:	Increasing 💌			Optional - specify with reset limits (default=Variable)
Reset value (minimum):	0			Optional - minimum value that indicates reset of values.
Reset value (maximum):	20			Optional - maximum value that indicates reset of values.
Analysis start:]	Optional - analysis start date/time (default=full time series period).
Analysis end:]	Optional - analysis end date/time (default=full time series period).
Flag:	Auto			Optional - flag to mark problem values (use Auto for defaults).
Alias to assign:	%L-delta	Insert:	Select Specifier 💉	Optional - use %L for location, etc. (default=no alias).
	Delta (Trend	d=Increasi	ng,ResetMin=O,Rese	tMax=20,Flag="Auto",Alias="%L-delta")
Command:				
Cancel OK				
				Delta

Delta() Command Editor

The command syntax is as follows:

Delta(Parameter=Value,...)

Parameter	Description	Default
TSList	 Indicates the list of time series to be processed, one of: AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards). AllTS – all time series before the command. EnsembleID – all time series in the ensemble specified by TSID. FirstMatchingTSID – the first time series that matches the TSID (single TSID or TSID with wildcards). LastMatchingTSID – the last time series that matches the TSID (single TSID or TSID with wildcards). SelectedTS – the time series are those selected with the SelectTimeSeries() command 	AllTS
TSID	The time series identifier or alias for the time series to be modified, using the * wildcard character to match multiple time series.	Must be specified if TSList=*TSID.
EnsembleID	The ensemble to be modified, if processing an ensemble.	Must be specified

Parameter	Description	Default
		if TSList=
		EnsembleID.
ResetMin	The minimum expected data value, used when data are	Data are not
	expected to increase (or decrease) to a threshold and then	expected to reset.
	reset, for example raw precipitation values that reset to zero	
	when a container fills.	
ResetMax	The maximum expected data value, used when data are	Data are not
	expected to increase (or decrease) to a threshold and then	expected to reset.
	reset, for example raw precipitation values that reset to zero	
	when a container fills.	
ExpectedTrend	Indicates trend of data, used when values can reset:	Data are variable
	• Decreasing – values should decrease and then reset	and don't reset at
	• Increasing – values should increase and then reset	fixed thresholds.
AnalysisStart	The date/time to start analyzing data.	Full period is
		analyzed.
AnalysisEnd	The date/time to end analyzing data.	Full period is
		analyzed.
Flag	A string to flag problem values, or Auto for default flags:	Do not flag
	• R – indicates reset transition out of range > ResetMax	problem values.
	• r - indicates reset transition out of range < ResetMin	
	• V - indicates value out of range > ResetMax	
	• v – indicates value out of range < ResetMin	
Alias	Alias to assign to created time series. A literal string can be	None (but is
	specified or use %-specifiers to set the alias dynamically	highly
	(e.g., %L) to use the location part of the identifier.	recommended).

This page is intentionally blank.
Command Reference: DeselectTimeSeries()

Deselect time series Version 10.13.00, 2012-10-25

The DeselectTimeSeries() command deselects output time series, as if done interactively, to indicate which time series SHOULD NOT be operated on by following commands. The command minimizes the need for the free() command when used in conjunction with other commands that use a time series list based on selected time series (TSList=SelectedTS). See also the SelectTimeSeries() command.

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

😹 Edit DeselectTimeSeries() Command		×			
This command deselects time series and is often used with the SelectTimeSeries() command.					
When matching a time series identifier (TSID) patt	ern:				
The dot-delimited time series identifier parts are	e Location.DataSource.DataType.Interval.Scenario				
The pattern used to select/deselect time series	will be matched against aliases and identifiers.				
Use * to match all time series.					
Use A* to match all time series with alias or loca	ation starting with A.				
Use *.*.XXXXX.*.* to match all time series with	a data type XXXX.				
When specifying time series positions:					
The first time series created is position 1.	a far susmala as 1.2				
Separate numbers by a comma. Specify a rang					
TS list: A	IIMatchingTSID Optional - indicates the time series to process (default=AIIT5).				
TSID (for TSList=AllMatchingTSID): 4	0*	-			
EnsembleID (for TSList=EnsembleID):		-			
Time series position(s) (for TSList=TSPosition): \Box	For example, 1,2,7-8 (positions are 1+).				
Select all first?:	rue Optional - eliminates need for separate select (default=False).				
a	eselectTimeSeries(TSList=AllMatchingTSID.TSID="40*".SelectAllFirst=True)				
	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,, _,, _	. 1			
Command:		. 1			
	DeselectTimeSe	ries			

DeselectTimeSeries() Command Editor

The command syntax is as follows:

```
DeselectTimeSeries(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
TSList	 Indicates the list of time series to be processed, one of: AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will 	Allts
	 be modified. AllTS – all time series before the command. EncombletD, all time series in 	
	 Ensemble() - an time series in the ensemble will be modified. LastMatchingTSID - the last 	
	 time series that matches the TSID (single TSID or TSID with wildcards) will be modified. TSPosition – time series 	
	specified by position in the results list (see TSPosition parameter below).	
TSID	The time series identifier or alias for the time series to be modified, using the * wildcard character to match multiple time series.	TSID or EnsembleID must be specified if identifiers are being matched.
EnsembleID	The ensemble to be modified, if processing an ensemble.	TSID or EnsembleID must be specified if identifiers are being matched.
TSPosition	A list of time series positions in output (1+), separated by commas.	Required if TSList=TSPosition.
SelectAllFirst	Indicates whether all time series should be selected before deselecting the specified time series: True or False.	False

A sample command file is as follows:

```
NewPatternTimeSeries(Alias="401234",NewTSID="401234..Precip.Day",
Description="Example data",SetStart="2000-01-01",SetEnd="2000-12-31",
Units="IN",PatternValues="0,1,3,0,0,0")
DeselectTimeSeries(TSList=AllMatchingTSID,TSID="40*",SelectAllFirst=True)
```

Command Reference: Disaggregate()

Create a new time series with shorter interval

Version 10.00.01, 2011-05-12

The Disaggregate() command creates a new time series by disaggregating a time series with a longer data interval into a time series with a shorter data interval. The resulting time series will have the same metadata and identifier as the original time series, with a different data interval. See also the general ChangeInterval() command.

Converting longer-interval data may cause a perceived shift in the time. For example, 1Day data shifted to 24Hour data will result in the daily values being set at hour zero of the following day. This shift is necessary to generically represent different time precision. Plots will also reflect the shift because hours are not considered when computing plot positions for daily data. It is important to understand how disaggregated data is treated with respect to time when using with other applications. If necessary, use the ShiftTimeByInterval() command to manipulate the resulting output time series.

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

🗞 Edit Disaggregate() Command 🛛 🛛 🔀						
Disaggregation conv	erts a longer-inter	val time series to	o a shorter int	erva	al.	
Specify the new inte	ral as Nbase, whe	re base is Minute	e, Hour, Day (omn	hitting N assumes a value of 1).	
The new interval mu	st be evenly divisit	ole into the old in	terval.			
The Ormsbee metho	d is only enabled t	o disaggregate 1	.Day to 6Hour	tim	e series.	
The SameValue meth	nod can disaggrega	ate Year->Month), Month->Da	γ, D	ay->NHour, Hour->NMinute.	
Original time series:	DayTS				*	
Alias to assign:	HourTS Insert: Select Specifier 💌 Required - use %L for location, etc.					
Method:	Ormsbee V Required - disaggregation method.					
New interval:	6Hour Required - e.g., Day, 6Hour.					
New data type:					Optional - default is same as original.	
New data units:	Optional - default is same as original.					
	Disaggrega	te(TSID="D	ayTS",Al:	ias	="HourTS",Method=Orms	
Command	bee,NewInte	erval=6Hou	r)			
Command.						
Cancel OK						
					Disagg	

Disaggregate() Command Editor

The command syntax is as follows:

Disaggregate(Parameter=Value,...)

The following older command syntax is updated to the above syntax when a command file is read:

TS Alias = Disaggregate(Parameter=Value,...)

Parameter	Description	Default
TSID	The time series identifier or alias for the time series to be	None – must be
	disaggregated.	specified.
Alias	The alias to assign to the time series, as a literal string or	None – must be
	using the special formatting characters listed by the	specified.
	command editor. The alias is a short identifier used by	
	other commands to locate time series for processing, as an	
	alternative to the time series identifier (TSID).	
Method	The method used to perform the disaggregation, one of	None – must be
	the following:	specified.
	Orsmbee – this method was presented in "Rainfall	
	Disaggregation Model for Continuous Hydrologic	
	Modeling," Ormsbee, Lindell E., Journal of Hydraulic	
	Engineering, ASCE, April, 1989. Currently the method	
	has only been enabled for disaggregating 1Day (not	
	24Hour) data to 6Hour data.	
	C	
	same value – uns simple method causes the resulting	
	time series to have the same value as the original. For	
	deily time series will result in each deily value being the	
	daily time series will result in each daily value being the	
	same as for the corresponding value in the original	
	disaggregations are supported:	
	disaggregations are supported.	
	• Year to Month	
	• Month to Day	
	• Day to NHour (including 24Hour)	
	• Hour to NMinute (including 60Minute)	
NewInterval	The data interval for the disaggregated time series	None – must be
	(NHour, NDay, etc.).	specified.
NewDataType	The data type for the disaggregated time series, if	Same data type
	different from the original.	as the original
		time series.
NewUnits	The units for the disaggregated time series, if different	Same units as
	from the original.	the original time
		series.

Command Parameters

An example command file to process data from the State of Colorado's HydroBase is as follows:

08223000 - RIO GRANDE RIVER AT ALAMOSA ReadTimeSeries(TSID="08223000.DWR.Streamflow.Day~HydroBase",Alias="DayTS") Disaggregate(TSID="DayTS",Alias="HourTS",Method=Ormsbee,NewInterval=6Hour) Examples of graphs for the original and disaggregated data are shown below, for the two disaggregation methods:



Daily Input Time Series and 6-Hour Disaggregated Time Series using SameValue Method



Daily Input Time Series and 6-Hour Disaggregated Time Series using Ormsbee Method

This page is intentionally blank.

Command Reference: Divide()

Divide the data values in one time series by data values in another time series

Version 08.16.04, 2008-09-24

Divide

The Divide() command divides one time series by another. This is useful for comparing the relative size of time series values (see also RelativeDiff()). If the divisor is zero or missing, the result is set to missing. Use the Scale() command to divide by a numerical value.

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

🛃 Edit Divide() Command	×			
Divide one time series by another	(the first time series is modified)			
Missing data in either time series o	or a zero denominator sets the result to missing.			
Time series to modify:	2184.NOAA.TempMean.Month			
Time series to divide by (divisor):	5706.NOAA.TempMean.Month			
Command:	Divide(TSID="2184.NOAA.TempMean.Month",DivisorTSID ="5706.NOAA.TempMean.Month")			
	Cancel OK			

Divide() Command Editor

The command syntax is as follows:

```
Divide(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
TSID	The time series identifier or alias for the time series to be	None – must be
	modified.	specified.
DivisorTSID	The time series identifier or alias for the time series that is the	None – must be
	divisor.	specified.

A sample command file to process data from the State of Colorado's HydroBase database is as follows:

2184 - DEL NORTE 2 E
2184.NOAA.TempMean.Month~HydroBase
5706 - MONTE VISTA 2 W
5706.NOAA.TempMean.Month~HydroBase
Divide(TSID="2184.NOAA.TempMean.Month",
 DivisorTSID="5706.NOAA.TempMean.Month")

The resulting graph is as follows:



Results from Divide() Command

Command Reference: Exit()

Stop processing commands

Version 08.16.04, 2008-09-25

Exit

The Exit() command can be inserted anywhere in a command file and causes the processing of commands to stop at that line. This is useful for temporarily processing a subset of a long list of commands. Multi-line comments (/* */) can also be used to temporarily disable one or more commands. It may also useful to add an Exit() command at the end of the file so that it is easy to insert commands above this command when the end line is selected (rather than having to deselect all commands when editing).

In the future the command may be enhanced to have parameters that more explicitly control processing shut-down.

Edit Exit() command	X

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

Edit Exit() command	5			
This command stops command processing, which is useful when troubleshooting.				
See also the /* */ comments, which can be used to comment a block of commands.				
Exit()				
Command:				
Cancel OK				

Exit() Command Editor

The command syntax is as follows:

```
Exit(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
	There are currently no command parameters.	

A sample command file is as follows:

Exit()

This page is intentionally blank.

Command Reference: ExpandTemplateFile()

Process a template file to create a fully-expanded file

Version 10.21.00, 2013-06-21

The ExpandTemplateFile() command processes a template file (such as a command file, time series product file, or HTML but can be any text file) to create a fully-expanded file and/or processor property. Templates facilitate utilizing conditional logic, loops, and other dynamic processing functionality that is not provided directly by TSTool. For example, a template can be used to repeat commands for multiple location identifiers. One advantage of using the template approach is that problems in the expanded file are clearly indicated, whereas a problem in logic that is represented as a loop might be difficult to diagnose.

The FreeMarker software (http://freemarker.org) is used to implement templates (support for other templating engines such as Apache Velocity could be added if needed). Refer to the online Freemarker documentation for information about the markup language used to create templates. Because TSTool checks commands for errors and does not itself understand FreeMarker syntax, template files must be edited with a text editor outside the normal TSTool editing. Attempts to edit a template command file in TSTool may result in error indicators and some command editors may not allow changes to be saved, such as when template notation is used for a filename and the command expects a parent folder name to exist. TSTool may be enhanced in the future to provide template editing features. Examples below illustrate how to use common FreeMarker features.

The FreeMarker built-in normalizeNewlines user directive is automatically used to ensure that expanded files use newline characters appropriate for the operating system. Otherwise the results may have all lines merged together (not an issue for HTML used by web browsers but a big issue with TSTool). The normalizeNewlines directive leads to temporary extra first and last lines in the template during processing, which need to be accounted for when interpreting FreeMarker warning messages. For example, a FreeMarker warning about line 21 would actually be line 20 in the original template file. FreeMarker messages may be difficult to interpret. In general errors are because variable names are not spelled correctly or there is an error specifying FreeMarker syntax.

The following information is automatically passed from TSTool to the ExpandTemplateFile() command:

- Properties set with the SetProperty() command are passed to the template processor. Consequently, the property names can be referenced with \${Property} in the template without using a FreeMarker assign command.
- One-column tables are passed as FreeMarker lists, using the table identifier (TableID) as the list property name. Null values in the table are passed as an empty string so that list have the correct number of items for iteration. Use the CopyTable() command to create a one-column table that can be used as a list for template expansion. The UseTables command parameter can be used to turn off this transfer, for example in cases where an ExpandTemplateFile() command is being repeated many times, does not use the tables, and is slowed down by converting the tables to FreeMarker lists.

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

a template file into a fu	Illy-expanded file and/or processor property.	
implemented using the	e FreeMarker package (freemarker.org).	
ige can be applied to c	ommand files to implement conditional logic, loops, etc.	
l receives properties s	et with SetProperty() and one-column tables are passed as a list using the tab	le identifier as the list nam
file names be relative t	to the working directory, which is:	
urceBuild\TSTool\test\	regression\commands\general\ExpandTemplateFile	
Data/List_Property_ten	mplate.TSTool	Browse
		Browse
istProperty	Optional - output string property (default=no output property).	
+	Optional - use 1-column tables as input lists (default=True).	
*	Optional - list expanded file in results (default=True).	
ExpandTemplate erty="ListProp	eFile(InputFile="Data/List_Property_template.TS" berty")	Fool",OutputProp
Add Working Direc	ctory (Input) Add Working Directory (Output) Cancel OK	
	implemented using the ge can be applied to co l receives properties s file names be relative to urceBuild\TSTool\test\ Data/List_Property_ter istProperty ExpandTemplate erty="ListProp Add Working Direct	implemented using the FreeMarker package (freemarker.org). ge can be applied to command files to implement conditional logic, loops, etc. I receives properties set with SetProperty() and one-column tables are passed as a list using the tab file names be relative to the working directory, which is: urceBuild\TSTool\test\regression\commands\general\ExpandTemplateFile Data/List_Property_template.TSTool istProperty Optional - output string property (default=no output property). v Optional - use 1-column tables as input lists (default=True). v Optional - list expanded file in results (default=True). ExpandTemplateFile (InputFile="Data/List_Property_template.TS" erty="ListProperty") Add Working Directory (Input) Add Working Directory (Input) Add Working Directory (Output) Cancel

ExpandTemplateFile() Command Editor

The command syntax is as follows:

ExpandTemplateFile(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
InputFile	The name of the template file to process. It is	None – must be
	recommended that the filename include "template"	specified.
	and that a comment in the file include @readOnly,	
	which will cause TSTool to warn users when saving	
	the expanded result.	
OutputFile	The name of the expanded output file.	None – must be
		specified.
OutputProperty	The name of a property to receive the results of the	No property value
	template expansion. This is appropriate when	will be set.
	templates are used to expand single-line text, for	
	example.	
UseTables	Indicate whether 1-column tables should be passed to	True
	the template expander. Doing so is a performance hit	
	and should be avoided if tables are not used in the	
	template.	
ListInResults	Indicate whether the results of the expansion should	True
	be listed in the TSTool results area. This may be	
	undesirable for "worker" files that users will	
	normally not view.	

Example Using Simple Variable Assignment

The following example illustrates a simple template command file and expanded result.

```
# Simple test to expand a text file using FreeMarker
#@readOnly
<#assign message="Hello World">
${message}
```

```
# Simple test to expand a text file using FreeMarker
#@readOnly
Hello World
```

Example of Passing Time Series Processor Properties to Templates

TSTool uses the \${Property} notation to dynamically replace the string with the corresponding property value (as a string). FreeMarker uses the \${Variable} notation to dynamically replace the string with the corresponding variable (as a string). Because the same notation is used by both software components, care must be taken to ensure that values are properly interpreted.

TSTool automatically passes all TSTool properties to the ExpandTemplateFile() command. Consequently, one of the main ways to avoid conflicts is to ensure that template command files do not use any of the properties defined in TSTool. One way to check property names is to insert a WritePropertyToFile() command at the appropriate line in a command file and review the list of properties that are shown when editing the command.

To utilize TSTool processor properties in a template, do not use the FreeMarker assign command and instead reference the property directly. The following TSTool command file illustrates how to define a property that is used by the ExpandTemplateFile() command:

```
# Simple test to expand a text file using FreeMarker
SetProperty(PropertyName="HelloWorldProp",PropertyType=String,
        PropertyValue="Hello World")
ExpandTemplateFile(InputFile="Data\ProcessorStringProperty.txt",
        OutputFile="Results/Test_ExpandTemplateFile_HelloWorld_out.txt")
```

The corresponding template command file is as follows:

```
# Simple test to expand a text file using FreeMarker
${HelloWorldProp}
```

The variables also can be used in assignment, similar to the following:

```
# Simple test to expand a text file using FreeMarker
<#assign message="${HelloWordProp}">
${message}
```

Example of Protecting TSTool Properties in Template with a Literal FreeMarker String

It is possible to use TSTool property notation in a template and cause FreeMarker to ignore the property. This will ensure that the property notation is present in the output, for TSTool to interpret at run-time.

The following example illustrates an input file that uses the FreeMarker $\{r"..."\}$ raw literal string notation:

```
# Simple test to expand a text file using FreeMarker
# and also escape text so that it passes through to the expanded file
# @readOnly
<#assign message="Hello World">
${r"SomeCommand($SomeProperty)"}
```

The corresponding output file is as follows:

```
# Simple test to expand a text file using FreeMarker
# and also escape text so that it passes through to the expanded file
# @readOnly
Hello World
SomeCommand($SomeProperty)
```

Example of Using a Comment in the Template, which is Omitted from Expanded Output

It often is desirable to have comments in the template file to explain the template, but not have the comments propagated to the expanded output. The following example illustrates an input file that uses the FreeMarker = - notation for comments:

```
# Simple test to expand a text file using FreeMarker
# There should be no comment in the expanded output below this line
<#-- This is a comment in the template -->
<#assign message="Hello World">
${message}
```

The corresponding output file is as follows:

```
# Simple test to expand a text file using FreeMarker
# There should be no comment in the expanded output below this line
Hello World
```

Example Using Variable Assignment and Loop Using List

The following example illustrates a template command file repeat a command for a list of location identifiers. A block of multiple commands can be repeated, as appropriate. Long lines are indented for illustration but would exist on a single line without indentation in the template file. Note that the loc_index FreeMarker syntax allows the loop counter to be used.

```
# Simple template to illustrate how to repeat commands with a list of
# location identifiers
# Create a time series for each location
# The following ensures that the created template is read-only, so users
# modify the template instead:
#@readOnly
<#assign setStart = "2000-01-01">
<#assign setStart = "2000-01-01">
<#assign setEnd = "2000-03-15">
<#assign units = "CFS">
<#assign units = "CFS">
<#assign locList = ["loc1", "loc2", "loc3", "loc4"]>
<#list locList as loc>
```

```
NewPatternTimeSeries(Alias="${loc}",NewTSID="${loc}..Streamflow.Day",
SetStart="${setStart}",SetEnd="${setEnd}",Units="${units}",
PatternValues="${loc_index + 1},0")
</#list>
```

The expanded command file is as follows:

```
# Simple template to illustrate how to repeat commands with a list of
# location identifiers
# Create a time series for each location
# The following ensures that the created template is read-only, so users
# modify the template instead:
#@readOnly
NewPatternTimeSeries(Alias="loc1",NewTSID="loc1..Streamflow.Day",
        SetStart="2000-01-01",SetEnd="2000-03-15",Units="CFS",PatternValues="1,0")
NewPatternTimeSeries(Alias="loc2",NewTSID="loc2..Streamflow.Day",
        SetStart="2000-01-01",SetEnd="2000-03-15",Units="CFS",PatternValues="2,0")
NewPatternTimeSeries(Alias="loc3",NewTSID="loc3..Streamflow.Day",
        SetStart="2000-01-01",SetEnd="2000-03-15",Units="CFS",PatternValues="2,0")
NewPatternTimeSeries(Alias="loc3",NewTSID="loc3..Streamflow.Day",
        SetStart="2000-01-01",SetEnd="2000-03-15",Units="CFS",PatternValues="3,0")
NewPatternTimeSeries(Alias="loc4",NewTSID="loc4..Streamflow.Day",
        SetStart="2000-01-01",SetEnd="2000-03-15",Units="CFS",PatternValues="3,0")
```

Example Using a One-Column Table for a List for Looping

The following example illustrates a template command file that reads the location list from a table. Note that the list must be a one-column table. If the original table has more than one column, read the original file and then use the CopyTable() command to create a new one-column table. A comma-separated-value (CSV) file is used for the list:

# Simple lis	st to	use	during	template	expansion
"Location"					
loc1					
loc2					
loc3					
loc4					

The template file is similar to the previous example; however, the list of locations is now provided via the table (no <#assign> element for the list) rather than having to hard-code in the template, which separates data from the processing logic:

```
# Simple template to illustrate how to repeat commands with a list of
# location identifiers
# Create a time series for each location
# The following ensures that the created template is read-only,
# so users modify the template instead:
# The list is provided by the processor as a one-column table with ID
# matching the list name
#@readOnly
<#assign setStart = "2000-01-01">
<#assign setEnd = "2000-03-15">
<#assign units = "CFS">
<#list locList as loc>
NewPatternTimeSeries(Alias="${loc}", NewTSID="${loc}..Streamflow.Day",
    SetStart="${setStart}",SetEnd="${setEnd}",Units="${units}",
    PatternValues="${loc_index + 1},0")
</#list>
```

The following command file reads the list of locations from the table and then expands the template file. Note that the TableID must match the list name in the #list...> element in the template.

Example Using a Multiple-Column Table to Loop Through Two Lists

The previous example illustrated how a one-column table can be used to loop over a list. However, often it is necessary to loop over one list and access the corresponding items from another list. The following example illustrates how a template command file can perform this task. Note that each list must be a one-column table in TSTool. If the original table has more than one column, use the CopyTable() command to create as many one-column tables as are necessary. In this example, a comma-separated-value (CSV) file is used for the table:

```
# Simple list to use during template expansion
"Location","Value"
loc1,1.0
loc2,2.0
loc3,3.0
loc4,4.0
```

The template file to expand is similar to the one-column example; however, the second list is also used to provide information when expanding the commands (see **bold text below**):

```
# Simple template to illustrate how to repeat commands with a list of
# location identifiers
# Create a time series for each location
# The following ensures that the created template is read-only,
# so users modify the template instead:
# The location list is provided by the processor as a one-column table
# with ID matching the list name
# The value list is provided by the processor as a corresponding one-column
# table with ID "valueList"
#@readOnly
<#assign setStart = "2000-01-01">
<#assign setEnd = "2000-03-15">
<#assign units = "CFS">
# The loc_index is referenced to zero
<#list locList as loc>
<#assign value = valueList[loc_index]>
NewPatternTimeSeries(Alias="${loc}",NewTSID="${loc}..Streamflow.Day",
    SetStart="${setStart}",SetEnd="${setEnd}",Units="${units}",
    PatternValues="${value}")
NewPatternTimeSeries(Alias="${loc}",NewTSID="${loc}..Streamflow.Day",
    SetStart="${setStart}",SetEnd="${setEnd}",Units="${units}",
    PatternValues="${valueList[loc index]}")
</#list>
```

The following command file reads the list of locations from the table and then expands the template file. Note that the TableID must match the list name in the <#list...> element and corresponding arrays in the template.

```
# Test expanding a FreeMarker template for a list of time series, using a
# two-column table as the list
# Read a one-column table that will be passed to the template as a list
ReadTableFromDelimitedFile(TableID="locList2",
    InputFile="Data\2column-table.csv")
CopyTable(TableID="locList2",NewTableID="locList",
    IncludeColumns="Location")
CopyTable(TableID="locList2",NewTableID="valueList",
    IncludeColumns="Value")
ExpandTemplateFile(InputFile="Data\ProcessorTable.txt",
    OutputFile="Results/Test ExpandTemplateFile ProcessorTable out.txt")
```

Example of Expanding a Template to a Processor Property

The following example illustrates how to expand a list into a SQL "in" clause, which is used to query specific matching records. A one-column table with identifier locList must have been created to supply the list of identifiers. The property set with the OutputProperty command parameter can then be used in the SQL statement for the ReadTableFromDataStore() command.

IN (<#list locList as loc><#if (loc_index > 0)>,</#if>'\${loc}'</#list>)

Example of Using ExpandTemplateFile() in a Loop to Expand Multiple Files

A template file cannot be expanded to multiple files using the approach illustrated above. However, by placing an ExpandTemplateFile() command inside a template that uses a loop (a list), it is possible to expand a template to multiple files. One example of this is the automated generation of time series product files used with TSTool graphs, where a graph is created for each location being processed. Graphs that are formatted with time series product files allow a certain amount of dynamic information to be considered. However, because the products are organized into product (page), sub-product (graph on page), and data (time series in graph), dynamic data may not be configurable at the desired level. For example, time series legend text can be configured to automatically use the time series identifier; however, this information is not appropriate for the page title because the software cannot automatically decide which time series to use for the main title.

A solution is to use a template time series product (TSP) file that has a place-holder variable for the title and then expand the TSP file as the command file is processed. For troubleshooting and data management purposes, it is recommended that the TSP files are saved in a folder separate from final output. The same TSP filename could be reused; however, the files are small and saving distinct files allows them to be used individually if necessary.

The following TSP file illustrates a simple graph:

```
# Template product file for graphs
[Product]
ProductType = "Graph"
TotalWidth = "600"
```

```
TotalHeight = "400"
MainTitleString = "${loc} Streamflow"
[SubProduct 1]
GraphType = "Line"
[Data 1.1]
TSID = "${loc}..Streamflow.Day"
TSAlias = "${loc}"
```

The following template command file illustrates how a property is set to control expansion of a template time series product file (note that templates are stored in a *Data* folder and final output in a *Results* folder for testing purposes but data management will vary by application):

```
# Simple template to illustrate how to repeat commands
# with a list of location identifiers, and produce individual graphs.
# The list is provided by the processor as a one-column table
# with ID matching the list name
# The @readOnly comment ensures that the created template is read-only,
# so users modify the template instead.
#@readOnly
<#assign setStart = "2000-01-01">
<#assign setEnd = "2000-03-15">
<#assign units = "CFS">
<#list locList as loc>
# Set the loc variable for the processor so that it can pass
# to the ExpandTemplateFile command below
SetProperty(PropertyName="loc", PropertyType="String", PropertyValue="${loc}")
# Create the time series
NewPatternTimeSeries(Alias="${loc}",NewTSID="${loc}..Streamflow.Day",
    SetStart="${setStart}",SetEnd="${setEnd}",Units="${units}",
   PatternValues="${loc_index + 1},0")
# Expand the time series product file (graph) for the time series
ExpandTemplateFile(InputFile="...\Data\ProcessorTable_TSP_template.tsp",
   OutputFile="${loc}.tsp")
# Process the graph
ProcessTSProduct(TSProductFile="${loc}.tsp",OutputFile="${loc}.tsp)
</#list>
```

The following expanded command file creates time series and graphs. Although blocks of commands are repeated, the location identifier is different in each block of commands. Any errors in processing can be pinpointed when the expanded command file is loaded and generally are due to logic errors in the original template (errors repeated throughout the expanded command file) or data availability issues in specific time series (errors in one part of the expanded command file).

Simple template to illustrate how to repeat commands # with a list of location identifiers, and produce individual graphs. # The list is provided by the processor as a one-column table # with ID matching the list name # The @readOnly comment ensures that the created template is read-only, # so users modify the template instead. #@readOnly # Set the loc variable for the processor so that it can pass

```
# to the ExpandTemplateFile command below
SetProperty(PropertyName="loc", PropertyType="String", PropertyValue="loc1")
# Create the time series
NewPatternTimeSeries(Alias="loc1",NewTSID="loc1..Streamflow.Day",
    SetStart="2000-01-01",SetEnd="2000-03-
15", Units="CFS", PatternValues="1,0")
# Expand the time series product file (graph) for the time series
ExpandTemplateFile(InputFile="...\Data\ProcessorTable_TSP_template.tsp",
    OutputFile="loc1.tsp")
# Process the graph
ProcessTSProduct(TSProductFile="loc1.tsp",OutputFile="loc1.tsp)
# Set the loc variable for the processor so that it can pass
# to the ExpandTemplateFile command below
SetProperty(PropertyName="loc",PropertyType="String",PropertyValue="loc2")
# Create the time series
NewPatternTimeSeries(Alias="loc2",NewTSID="loc2..Streamflow.Day",
    SetStart="2000-01-01",SetEnd="2000-03-
15", Units="CFS", PatternValues="2,0")
# Expand the time series product file (graph) for the time series
ExpandTemplateFile(InputFile="...Data\ProcessorTable_TSP_template.tsp",
    OutputFile="loc2.tsp")
# Process the graph
ProcessTSProduct(TSProductFile="loc2.tsp",OutputFile="loc2.tsp)
# Set the loc variable for the processor so that it can pass
# to the ExpandTemplateFile command below
SetProperty(PropertyName="loc", PropertyType="String", PropertyValue="loc3")
# Create the time series
NewPatternTimeSeries(Alias="loc3",NewTSID="loc3..Streamflow.Day",
    SetStart="2000-01-01",SetEnd="2000-03-
15", Units="CFS", PatternValues="3,0")
# Expand the time series product file (graph) for the time series
ExpandTemplateFile(InputFile="..\Data\ProcessorTable_TSP_template.tsp",
    OutputFile="loc3.tsp")
# Process the graph
ProcessTSProduct(TSProductFile="loc3.tsp",OutputFile="loc3.tsp)
# Set the loc variable for the processor so that it can pass
# to the ExpandTemplateFile command below
SetProperty(PropertyName="loc",PropertyType="String",PropertyValue="loc4")
# Create the time series
NewPatternTimeSeries(Alias="loc4",NewTSID="loc4..Streamflow.Day",
    SetStart="2000-01-01",SetEnd="2000-03-
15", Units="CFS", PatternValues="4,0")
# Expand the time series product file (graph) for the time series
ExpandTemplateFile(InputFile="..\Data\ProcessorTable_TSP_template.tsp",
    OutputFile="loc4.tsp")
# Process the graph
ProcessTSProduct(TSProductFile="loc4.tsp",OutputFile="loc4.tsp)
```

Also note that when the expanded command file is first opened in TSTool the ProcessTSProduct() commands will have a failure indicated. This is because the TSP file being used by the command has not yet been created (it is created as the commands are run). After the commands are run one time, the files will exist and subsequent loads of the command file will not show the warnings. This illustrates a potential issue, which is that templates can result in large numbers of files and the files should be cleared at appropriate times to ensure that old files are not used by mistake.

The FreeMarker language provides many features beyond those illustrated in these examples, including conditional ("if") statements. However, the more complex the templates become, the more difficult they are to implement, troubleshoot, and maintain. Enhancements to TSTool may help with a solution that otherwise might require undesirable complexity.

This page is intentionally blank.

This page is intentionally blank.

Command Reference: FillConstant()

Fill missing time series data using a constant value

Version 09.07.02, 2010-08-20

The FillConstant() command fills the missing data in a time series with the specified value. This fill technique is useful for filling missing data with zeros, perhaps as the last step in a sequence of filling commands.

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

👌 Edit FillConstant() Comma	nd	
Fill time series data missing values with	a constant value	
The time series to process are indicate	d using the TS list	
If TS list is "AllMatchingTSID", pick a sir	ngle time series, or	r enter a wildcard time series identifier pattern.
TS list:	AllMatchingTSID	Optional - indicates the time series to process (default=AlITS).
TSID (for TSList=AllMatchingTSID):	08236500.DWR.5	Streamflow.Month
EnsembleID (for TSList=EnsembleID):		×
Constant value:	500	Required - constant value to fill missing data.
Fill start date/time:	1970-02	Optional - fill start (default=fill all).
Fill end date/time:	1970-10	Optional - fill end (default=fill all).
Fill flag:	с	Optional - string to flag filled values.
Command:	FillConsta .DWR.Streau ="1970-02"	nt(TSList=AllMatchingTSID,TSID="08236500 mflow.Month",ConstantValue=500,FillStart ,FillEnd="1970-10",FillFlag="C")
	Ca	

FillConstant() Command Editor

The command syntax is as follows:

```
FillConstant(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one	Allts
	of:	
	• AllMatchingTSID – all time series that	
	match the TSID (single TSID or TSID with	
	wildcards) will be modified.	
	• AllTS – all time series before the command.	
	• EnsembleID – all time series in the ensemble	
	will be modified.	
	• FirstMatchingTSID – the first time series	
	that matches the TSID (single TSID or TSID	
	with wildcards) will be modified.	
	• LastMatchingTSID – the last time series	
	that matches the TSID (single TSID or TSID	
	with wildcards) will be modified.	
	• SelectedTS – the time series are those	
	selected with the SelectTimeSeries()	
מדפיד	Command. The time series identifier or clies for the time series	Dequired for
1010	to be modified using the * wildcard character to	
	match multiple time series	
EnsembleID	The ensemble to be modified if processing an	Required for
	ensemble.	TSList=EnsembleID.
ConstantValue	Constant value to use when filling missing data.	None – must be specified.
FillStart	Date/time indicating the start of filling, using a	Fill the entire time series.
	precision appropriate for the time series, or	
	OutputStart.	
FillEnd	Date/time indicating the end of filling, using a	Fill the entire time series.
	precision appropriate for the time series, or	
	OutputEnd.	
FillFlag	If specified, data flags will be enabled for the time	No flag is assigned.
	series and each filled value will be tagged with the	
	specified string. The flag can then be used later to	
	label graphs, etc. The flag will be appended to	
1	existing flags if necessary.	

A sample command file to fill a time series from the State of Colorado's HydroBase is as follows:

```
# 08236500 - ALAMOSA RIVER BELOW TERRACE RESERVOIR
08236500.DWR.Streamflow.Month~HydroBase
FillConstant(TSList=AllMatchingTSID,TSID="08236500.DWR.Streamflow.Month",
ConstantValue=500,FillStart="1970-02",FillEnd="1970-10",FillFlag="C")
```

Command Reference: FillDayTSFrom2MonthTSAnd1DayTS()

Fill a daily time series from monthly volumes and daily pattern

The FillDayTSFrom2MonthTSAnd1DayTS() command fills a daily time series using the following relationship:

 $D1_i = D2_i^*(M1_i/M2_i)$

where:

i = dayD1 is the daily data at location 1M1 is the monthly data at location 1 (for the month corresponding to the day)D2 is the daily data at location 2M2 is the monthly data at location 2 (for the month corresponding to the day)

This fill method assumes the monthly time series are filled and reasonably correlated and that the daily pattern D2 can be applied at D1. For example, use this command to fill daily streamflow where filled monthly data are available at nearby locations and filled daily data is available at the independent (D2) station.

The following dialog is used to edit the command and illustrates the syntax of the command. For all the time series identifiers, the last matching identifier before the command will be matched for processing. Currently there is no way to fill multiple time series with one command.

Edit FillDayTSFrom2MonthTS	and1DayTS() Command	۲
Fill a daily time series using the relation	nship: D1[i] = D2[i]*M1[i]/M2[i].	
The monthly values M1/M2 give an ave	rage estimate of the volume ratio and D2 provides an estimate of the daily pattern in the month	h.
Note that TSTool cannot verify wh	ether time series are daily or monthly until the command is run.	
Daily time series to fill (D1):	08235350.USGS.Streamflow.Day]
Associated monthly time series (M1):	08235350.USGS.Streamflow.Month]
Monthly time series for total (M2):	08236000.DVVR.Streamflow.Month]
Daily time series for distribution (D2):	08236000.DWR.Streamflow.Day]
Fill start date/time:	Optional - default=fill al	Ι.
Fill end date/time:	Optional - default=fill al	۱.
Command:	FillDayTSFrom2MonthTSAndlDayTS(TSID_D1="08235350.USGS.Streamflow .Day",TSID_M1="08235350.USGS.Streamflow.Month",TSID_M2="08236000 .DWR.Streamflow.Month")	
	Cancel OK	<u>л</u> е

FillDayTSFrom2MonthTSAnd1DayTS() Command Editor

The command syntax is as follows:

FillDayTSFrom2MonthTSAnd1DayTS(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSID_D1	The time series identifier or alias for the daily time	None – must be
	series to be filled.	specified.
TSID_M1	The time series identifier or alias for the monthly	None – must be
	time series, corresponding to TSID_D1, to supply	specified.
	the monthly values to be distributed to daily.	
TSID_M2	The time series identifier or alias for the independent	None – must be
	monthly time series.	specified.
TSID_D2	The time series identifier or alias for the independent	None – must be
	daily time series, corresponding to TSID_M2.	specified.
FillStart	Date/time indicating the start of filling, using a	Fill the entire time
	precision appropriate for the time series, or	series.
	OutputStart.	
FillEnd	Date/time indicating the end of filling, using a	Fill the entire time
	precision appropriate for the time series, or	series.
	OutputEnd.	

An example command file to process data from the State of Colorado's HydroBase is shown below with the resulting graph of daily time series.

The following is D1:
(1995-1998) ALAMOSA RIVER ABOVE JASPER, CO USGS Streamflow Daily
08235350.USGS.Streamflow.Day~HydroBase
The following is M1:
(1995-1998) ALAMOSA RIVER ABOVE JASPER, CO USGS Streamflow Monthly
08235350.USGS.Streamflow.Month~HydroBase
The following is D2:
(1914-1998) ALAMOSA RIVER ABOVE TERRACE RESERVOIR, CO. DWR Streamflow Daily
08236000.DWR.Streamflow.Day~HydroBase
The following is M2:
(1914-1998) ALAMOSA RIVER ABOVE TERRACE RESERVOIR, CO. DWR Streamflow Monthly
08236000.DWR.Streamflow.Month~HydroBase
FillRegression(TSID="08235350.USGS.Streamflow.Month",
IndependentTSID="08236000.DWR.Streamflow.Month",
NumberOfEquations=OneEquation,Transformation=Linear)
FillDayTSFrom2MonthTSAnd1DayTS(TSID_D1="08235350.USGS.Streamflow.Day",
TSID_M1="08235350.USGS.Streamflow.Month",
TSID M2="08236000.DWR.Streamflow.Month",TSID D2="08236000.DWR.Streamflow.Day")

The following graph shows the two daily time series used in the command (zoomed in). Note that the shape of the filled time series is similar to the other time series.



Example of Filled Data

Command Reference: FillFromTS()

Fill missing time series data using data from another time series (or ensemble)

Version 10.03.00, 2011-12-19

The FillFromTS() command fills missing data in a time series (or ensemble) by transferring nonmissing values from another time series (or ensemble). This is useful when two time series typically have very similar values. The filled time series is not automatically extended. A period can be specified to limit the period that is checked for missing data. See also the SetFromTS() command, which will transfer all values. If multiple time series or an ensemble is being processed, the number of independent time series must be one or the same number as the time series being filled. Data transfer occurs by date/time, not sequentially. This may be a problem if trying to fill from a time series that has been shifted and leap years have caused an offset – an enhancement may be made in the future to address this issue. The following dialog is used to edit the command and illustrates the command syntax.

Sedit FillFromTS() Command	
Copy data values from the independent time series to replace mis	ssing values in the dependent time series. Only data in the fill period will be checked.
If one independent time series is specified, it will be used for all d	ependent time series.
If multiple independent time series are specified (e.g., for ensemb	bles), the same number of dependent time series must be specified.
Use a SetOutputPeriod() command if the dependent time series p	eriod will be extended.
Specify dates with precision appropriate for the data, use blank f	or all available data, OutputStart, or OutputEnd.
Dependent TS List:	AllMatchingTSID V Optional - indicates the time series to process (default=AllTS).
TSID (for TSList=*MatchingTSID):	08241000.DWR.Streamflow.Month
EnsembleID (for T5List=EnsembleID):	
Independent TS List:	AllMatchingTSID Optional - indicates the time series to process (default=AllTS).
Independent TSID (for Independent TSList=*MatchingTSID):	08240500.DWR.Streamflow.Month
Independent EnsembleID (for Independent T5List=EnsembleID);	
Fill start date/time:	Optional - start of period to fill (default=fill all).
Fill end date/time:	Optional - end of period to fill (default=fill all).
Fill flag:	Optional - string to indicate filled values (default=no flag).
Fill flag description:	Optional - description for fill flag.
Recalculate limits:	Optional - recalculate original data limits after fill (default=False).
Command:	FillFromTS(TSList=AllMatchingTSID,TSID="08241000.DWR. Streamflow.Month",IndependentTSList=AllMatchingTSID,I ndependentTSID="08240500.DWR.Streamflow.Month")
	Cancel OK

FillFromTS() Command Editor

The command syntax is as follows:

```
FillFromTS(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of:	AllTS
	• AllMatchingTSID – all time series that match	

Parameter	Description	Default
	the TSID (single TSID or TSID with wildcards)	
	will be modified.	
	• AllTS – all time series before the command.	
	• EnsembleID – all time series in the ensemble will	
	be modified.	
	• FirstMatchingTSID – the first time series that	
	matches the TSID (single TSID or TSID with	
	wildcards) will be modified.	
	• LastMatchingTSID – the last time series that	
	matches the TSID (single TSID or TSID with	
	wildcards) will be modified.	
	• SelectedTS – the time series are those selected	
	with the SelectTimeSeries() command.	
TSID	The time series identifier or alias for the time series to	Required when a
	be modified, using the * wildcard character to match	TSList=*TSID
	multiple time series.	
EnsembleID	The ensemble to be modified, if processing an	Required when
	ensemble.	TSList=EnsembleID.
Independent	Indicates how to determine the list of independent time	AllTS
TSList	series (see the explanation of TSList).	
Independent	The time series identifier or alias for the independent	Required when a
TSID	time series (see the explanation of TSID).	Independent TSList=
Indopondont	The encomple identifier for the independent time cories	ATSID Dequired when
FngembleTD	(see the explanation of Engemble JD)	Independent TSList -
EIISCIIDICID	(see the explanation of Ensembleid).	EnsembleID
FillStart	The date/time to start filling.	Fill the entire period.
FillEnd	The date/time to end filling.	Fill the entire period.
FillFlag	If specified, data flags will be enabled for the time	No flag is assigned.
	series and each filled value will be tagged with the	
	specified string. The flag can then be used later to label	
	graphs, etc. Prefix with + to append the flag.	
FillFlagDesc	Description for the fill flag, used in reports.	Automatically generated.
Recalc	Available only for monthly time series. Indicate	False (only the values in
Limits	whether the original data limits for the time series	the initial time series will
	should be recalculated after the filling the time series.	be used for historical
	Setting to True is appropriate if the independent time	data).
	series provides additional data values.	

A sample command file to fill data from the State of Colorado's HydroBase is as follows:

```
# 08241000 - TRINCHERA CREEK ABOVE MOUNTAIN HOME RESERVOIR
08241000.DWR.Streamflow.Month~HydroBase
# 08240500 - TRINCHERA CREEK ABOVE TURNER'S RANCH
08240500.DWR.Streamflow.Month~HydroBase
FillFromTS(TSList=AllMatchingTSID,TSID="08241000.DWR.Streamflow.Month",
IndependentTSList=AllMatchingTSID,
IndependentTSID="08240500.DWR.Streamflow.Month")
```

Command Reference: FillHistMonthAverage()

Fill missing time series data using historical monthly average data

Version 09.07.00, 2010-06-14

The FillHistMonthAverage() command fills missing data in monthly time series with the average monthly values. The average values are computed using the available data period (or specified averaging period – see the SetAveragePeriod() command) immediately after the time series is read and are then applied when this command is encountered.

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

COLOTE FILIFIISTMONTINA VERAGE) Command	
Fill monthly time series with historical m	nonthly averages.	
Historical averages are computed imme	ediately after reading th	e data and therefore do not include filled values.
Only monthly time series can be proces	ssed.	
TS list: 🗚	llMatchingTSID 🛛 🐱	Optional - indicates the time series to process (default=AIITS).
TSID (for TSList=AllMatchingTSID): 11	179.NOAA.Precip.Month	✓
Fill start:		Optional - default is full period.
Fill end:		Optional - default is full period.
Fill flag: H		Optional - string to indicate filled values (use "auto" for MonAvg).
Fill flag description:		Optional - description for fill flag.
F Command: 9	illHistMonthAve .NOAA.Precip.Mo	erage(TSList=AllMatchingTSID,TSID="117 onth",FillFlag="H")
	Cancel	ОК

FillHistMonthAverage() Command Editor

The command syntax is as follows:

FillHistMonthAverage(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSList	Indicate how to determine the list of time series to process, one of:	None – must be specified.
	 AllMatchingTSID - process time series that have identifiers matching the TSID parameter. AllTS - process all the time series. FirstMatchingTSID - process the first time series that has an identifier matching the TSID parameter. 	

Parameter	Description	Default
	 LastMatchingTSID - process the last time series that has an identifier matching the TSID parameter. SelectedTS - process the time series that are selected (see SelectTimeSeries()). 	
TSID	Used if TSList=AllMatchingTSID to indicate the time series identifier or alias for the time series to be filled. Specify * to match all time series or use a wildcard for one or more identifier parts.	Required if TSList=AllMatchin gTSID.
FillStart	Date/time indicating the start of filling, using a precision appropriate for the time series, or OutputStart.	Fill the entire time series.
FillEnd	Date/time indicating the end of filling, using a precision appropriate for the time series, or OutputEnd.	Fill the entire time series.
FillFlag	If specified, data flags will be enabled for the time series and each filled value will be tagged with the specified string. The flag can then be used later to label graphs, etc. The flag will be appended to existing flags if necessary. Use Auto to use a flag with the month abbreviation + Avg.	No flag is assigned.
FillFlag Desc	Description for the fill flag, used in reports.	Automatically generated.

The following command files fill a time series from the State of Colorado's HydroBase:

```
# 0125 - ALAMOSA
0125.NOAA.Precip.Month~HydroBase
FillHistMonthAverage(TSList=AllMatchingTSID,TSID="0125.NOAA.Precip.Month",
FillFlag="H")
```

```
0125.NOAA.Precip.Month~HydroBase
FillHistMonthAverage(TSList=AllMatchingTSID,TSID="019*",FillFlag="H")
```

Time series data limits for the averages are printed to the log file, similar to the following examples (note that the period for averaging is always shown and may be different than the output period).

Status: historic averages for the series fortow Time series: 0125.NOA.Precip.Month (IN) Monthly limits for period 1948-08 to 1949-12 are: # % # Not % Not Month Min MinDate Max MaxDate Sum Miss. Miss. Jan 0.2 1949-01 0.2 1949-01 0.2 0 Jan 0.1 1949-02 0.1 1949-02 0.1 0 0.00 1 100.00 0. Feb 0.1 1949-03 0.1 1949-03 0.1 0 0.00 1 100.00 0. Max -999.0 -999.0 100.00 0.00 -999. May -999.0 -999.0 100.00 0.00 -999. Jun 0.7 1949-06 0.7 1 0.5 0 0.00 1 100.00 0. Jul 1.5 1949-07 1.5 0 0.00 1 100.00 0. Aug 0.7 1949-08 0.8 1948-08 1.5 0 0.00 2 100.00 0. Sep 0.1 1948-09 1.1 1949-09 1.2 0 0.00 2 100.00 0. Nov 0.0 1949-11 0.8 1948-10 0.7 0 0.00 2 100.00 0. Dec 0.0 1949-12	Status	Higtori	a 20072900	a for time	goriog f						
Monthly limits for period 1948-08 to 1949-12 are: Monthly limits for period 1948-08 to 1949-12 are: Month Min MinDate Max MaxDate Sum Miss. Miss. Miss. Mess. Jan 0.2 1949-01 0.2 1949-01 0.2 0 0.000 1 100.00 0. Feb 0.1 1949-02 0.1 1949-02 0.1 0 0.00 1 100.00 0. Mar 0.1 1949-03 0.1 1949-03 0.1 0 0.00 1 100.00 0. Apr -999.0 -999.0 -999.0 1 100.00 0 0.00 -999. Jun 0.7 1949-06 0.7 1949-07 1.5 0 0.00 1 100.00 0. Jul 1.5 1949-07 1.5 1949-07 1.5 0 0.00 1 00.00 0. Aug 0.7 1949-08 0.8 1948-08 1.5 0 0.00 2 100.00 0. Sep 0.1 1948-09 1.1 1949-09 1.2 0 0.00 2 100.00 0. Nov 0.0 1949-11 0.8 1948-11 0.8 0 0.00 2 100.00 0.	Time de	riog: 0	L AVELAGE	Drogin Mon	+b (IN)	0110w					
# % # Not % Not Month Min MinDate Max MaxDate Sum Miss. Miss. Miss. Miss. Mean Jan 0.2 1949-01 0.2 1949-02 0.1 0.00 1 100.00 0. Feb 0.1 1949-02 0.1 1949-02 0.1 0 0.00 1 100.00 0. Mar 0.1 1949-03 0.1 0 0.00 1 100.00 0. Apr -999.0 -999.0 -999.0 1 100.00 0 0.00 -999. Jun 0.7 1949-06 0.7 1949-06 0.7 0 0.00 1 100.00 0. Jul 1.5 1949-07 1.5 0 0.00 1 100.00 1. Aug 0.7 1949-08 0.8 1948-08 1.5 0 0.00 2 100.00 0. Sep 0.1 1948-09 1.1 1948-09 1.2 0 0.00 2 10	Manthla	lies. U	125.NOAA.	J 1040 00	LII (IN)	0					
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	MONUNLY	limits	tor perio	d 1948-08	1949-1	z are.		<u>^</u>		0 37 1	
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$						_	#	8	# Not	% Not	
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	Month	Min	MinDate	Max	MaxDate	Sum	Miss.	Miss.	Miss.	Miss.	Mean
Odif 0.2 1.949 0.1 0.12 1.949 0.12 0.12 0.000 1 100.000 0.000 Mar 0.1 $1949-02$ 0.1 $1949-02$ 0.1 0 0.000 1 100.000 0.000 Apr -999.0 -999.0 -999.0 -999.0 1 100.00 0 0.000 -999.0 Jun 0.7 $1949-06$ 0.7 $1949-06$ 0.7 0 0.000 1 100.000 0.000 Jul 1.5 $1949-07$ 1.5 0 0.000 1 100.000 0.000 Aug 0.7 $1949-08$ 0.8 $1948-08$ 1.5 0 0.000 2 100.000 0.000 Sep 0.1 $1948-09$ 1.1 $1949-09$ 1.2 0 0.000 2 100.000 0.000 Nov 0.0 $1949-11$ 0.8 $1948-11$ 0.8 0 0.000 2 100.000 0.000 Dec 0.0 $1949-12$ 0.2 $1948-12$ 0.2 0.000 2 100.000 0.000		 0 2	1949_01	0 2	1949_01			0 00	1	100 00	0 2
Mar 0.1 1949-02 0.1 1949-02 0.1 100.00 1 100.00 0. Apr -999.0 -999.0 -999.0 1 100.00 0 0.00 -999.0 May -999.0 -999.0 -999.0 1 100.00 0 0.00 -999.0 Jun 0.7 1949-06 0.7 1949-06 0.7 0 0.00 1 100.00 0 Jul 1.5 1949-07 1.5 0 0.00 1 100.00 0. Jul 0.7 1949-08 0.8 1948-08 1.5 0 0.00 2 100.00 0. Sep 0.1 1948-09 1.2 0 0.00 2 100.00 0. Nov 0.0 1949-10 0.5 1948-10 0.7 0 0.00 2 100.00 0. Dec 0.1 1949-12 0.2 1948-12 0.2 0 0.00 2 100.00 0.	Tab	0.2	1040 02	0.2	1040 02	0.2	0	0.00	1	100.00	0.2
Mar 0.1 1949-03 0.1 1949-03 0.1 0 0.00 1 100.00 0 Apr -999.0 -999.0 -999.0 1 100.00 0 0.00 -999.0 May -999.0 -999.0 1 100.00 0 0.00 -999.0 Jun 0.7 1949-06 0.7 0 0.00 1 100.00 0 0.00 -999.0 Jul 1.5 1949-07 1.5 1949-07 0.00 1 100.00 0.0 Jul 1.5 1949-07 1.5 0 0.00 1 100.00 0. Aug 0.7 1949-08 0.8 1948-08 1.5 0 0.00 2 100.00 0. Sep 0.1 1948-09 1.1 1949-09 1.2 0 0.00 2 100.00 0. Oct 0.1 1949-11 0.8 1948-11 0.8 0 <th< td=""><td>гер</td><td>0.1</td><td>1949-02</td><td>0.1</td><td>1949-02</td><td>0.1</td><td>0</td><td>0.00</td><td>1</td><td>100.00</td><td>0.1</td></th<>	гер	0.1	1949-02	0.1	1949-02	0.1	0	0.00	1	100.00	0.1
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	Mar	0.1	1949-03	0.1	1949-03	0.1	0	0.00	1	100.00	0.1
May -999.0 -999.0 -999.0 1 100.00 0 0.00 -999.0 Jun 0.7 1949-06 0.7 1949-06 0.7 0 0.00 1 100.00 0. Jul 1.5 1949-07 1.5 1949-07 1.5 0 0.00 1 100.00 0. Aug 0.7 1949-08 0.8 1948-08 1.5 0 0.00 2 100.00 0. Sep 0.1 1949-09 1.1 1949-09 1.2 0 0.00 2 100.00 0. Oct 0.1 1949-10 0.5 1948-10 0.7 0 0.00 2 100.00 0. Nov 0.0 1949-11 0.8 1948-11 0.8 0 0.00 2 100.00 0. Dec 0.0 1949-12 0.2 1948-12 0.2 0 0.00 2 100.00 0.	Apr	-999.0		-999.0		-999.0	1	100.00	0	0.00	-999.0
Jun 0.7 1949-06 0.7 1949-06 0.7 0 0.00 1 100.00 0. Jul 1.5 1949-07 1.5 1949-07 1.5 0 0.00 1 100.00 1. Aug 0.7 1949-08 0.8 1948-08 1.5 0 0.00 2 100.00 0. Sep 0.1 1948-09 1.1 1949-09 1.2 0 0.00 2 100.00 0. Oct 0.1 1949-10 0.5 1948-10 0.7 0 0.00 2 100.00 0. Nov 0.0 1949-11 0.8 1948-11 0.8 0 0.00 2 100.00 0. Dec 0.0 1949-12 0.2 1948-12 0.2 0 0.00 2 100.00 0.	May	-999.0		-999.0		-999.0	1	100.00	0	0.00	-999.0
Jul 1.5 1949-07 1.5 1.5 0 0.00 1 100.00 1. Aug 0.7 1949-08 0.8 1948-08 1.5 0 0.00 2 100.00 0. Sep 0.1 1948-09 1.1 1949-09 1.2 0 0.00 2 100.00 0. Oct 0.1 1949-10 0.5 1948-10 0.7 0 0.00 2 100.00 0. Nov 0.0 1949-11 0.8 1948-11 0.8 0 0.00 2 100.00 0. Dec 0.0 1949-12 0.2 1948-12 0.2 0 0.00 2 100.00 0.	Jun	0.7	1949-06	0.7	1949-06	0.7	0	0.00	1	100.00	0.7
Aug 0.7 1949-08 0.8 1948-08 1.5 0 0.00 2 100.00 0. Sep 0.1 1948-09 1.1 1949-09 1.2 0 0.00 2 100.00 0. Oct 0.1 1949-10 0.5 1948-10 0.7 0 0.00 2 100.00 0. Nov 0.0 1949-11 0.8 1948-11 0.8 0 0.00 2 100.00 0. Dec 0.0 1949-12 0.2 1948-12 0.2 0 0.00 2 100.00 0.	Jul	1.5	1949-07	1.5	1949-07	1.5	0	0.00	1	100.00	1.5
Sep 0.1 1948-09 1.1 1949-09 1.2 0 0.00 2 100.00 0. Oct 0.1 1949-10 0.5 1948-10 0.7 0 0.00 2 100.00 0 Nov 0.0 1949-11 0.8 1948-11 0.8 0 0.00 2 100.00 0 Dec 0.0 1949-12 0.2 1948-12 0.2 0 0.00 2 100.00 0	Aug	0.7	1949-08	0.8	1948-08	1.5	0	0.00	2	100.00	0.8
Oct 0.1 1949-10 0.5 1948-10 0.7 0 0.00 2 100.00 0. Nov 0.0 1949-11 0.8 1948-11 0.8 0 0.00 2 100.00 0 Dec 0.0 1949-12 0.2 1948-12 0.2 0 0.00 2 100.00 0	Sep	0.1	1948-09	1.1	1949-09	1.2	0	0.00	2	100.00	0.6
Nov 0.0 1949-11 0.8 1948-11 0.8 0 0.00 2 100.00 0. Dec 0.0 1949-12 0.2 1948-12 0.2 0 0.00 2 100.00 0	Oct	0.1	1949-10	0.5	1948-10	0.7	0	0.00	2	100.00	0.3
Dec 0.0 1949-12 0.2 1948-12 0.2 0 0.00 2 100.00 0.	Nov	0.0	1949-11	0.8	1948-11	0.8	0	0.00	2	100.00	0.4
	Dec	0.0	1949-12	0.2	1948-12	0.2	0	0.00	2	100.00	0.1
Period 0.0 1949-11 1.5 1949-07 6.9 2 11.76 15 88.24 0.	Period	0.0	1949-11	1.5	1949-07	6.9	2	11.76	15	88.24	0.5

Command Reference: FillHistYearAverage()

Fill missing time series data using historical yearly average data

Version 10.00.01, 2011-05-12

The FillHistYearAverage() command fills missing data in yearly time series with the average annual value. The average values are computed using the available data period (or specified averaging period – see the SetAveragePeriod() command) immediately after the time series is read and are then applied when this command is encountered.

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

🚯 Edit FillHistYearAverage() Command				
Fill yearly time series with historical yearly averages.				
Historical averages are computed immediately after reading the data and therefore do not consider filled values.				
Only yearly time series can be processed.				
The time series to process are indicated using the TS list.				
If TS list is "AllMatchingTSID", pick a single time series, or enter a wildcard time series identifier pattern.				
TS list:	AllMatchingTSID 😽	How to get the time series to fill.		
Identifier (TSID) to match:	LARIMER.NASS.CropArea	a-Vegetables, Harvested.Year 🛛 👻		
Fill start:		Optional - default is full period.		
Fill end:		Optional - default is full period.		
Fill flag:		Optional - string to flag filled values.		
	FillHistYearAve	rage(TSList=AllMatchingTSID,TSID="		
Command:	LARIMER.NASS.CropArea-Vegetables, Harvested.Year")			
command.				
Cancel OK				
		FillHistY		

FillHistYearAverage() Command Editor

The command syntax is as follows:

FillHistYearAverage(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSList	Indicate how to determine the list of time series to process, one of:	None – must be specified.
	 AllMatchingTSID – process time series that have identifiers matching the TSID parameter. AllTS – process all the time series. 	

Parameter	Description	Default
	 SelectedTS – process the time 	
	series that are selected (see	
	<pre>selectTimeSeries()).</pre>	
TSID	Used if TSList=AllMatchingTSID	Required if
	to indicate the time series identifier or	TSList=AllMatchingTSID.
	alias for the time series to be filled.	
	Specify * to match all time series or use	
	a wildcard for one or more identifier	
	parts.	
FillStart	Date/time indicating the start of filling,	Fill the entire time series.
	using a precision appropriate for the time	
	series, or OutputStart.	
FillEnd	Date/time indicating the end of filling,	Fill the entire time series.
	using a precision appropriate for the time	
	series, or OutputEnd.	
FillFlag	If specified as a single character, data	No flag is assigned.
	flags will be enabled for the time series	
	and each filled value will be tagged with	
	the specified character. The flag can	
	then be used later to label graphs, etc.	
	The flag will be appended to existing	
	flags if necessary.	

A sample command file to fill data from the State of Colorado's HydroBase is as follows:

```
LARIMER.NASS.CropArea-Vegetables, Harvested.Year~HydroBase
FillHistYearAverage(TSList=AllMatchingTSID,
TSID="LARIMER.NASS.CropArea-Vegetables, Harvested.Year")
```

Time series data limits for the averages are printed to the log file, similar to the following example (note that the period for averaging is always shown and may be different than the output period).

Min:		95	.00	000	ACRE	on	1954
Max:		2684	.00	000	ACRE	on	1959
Sum:		11090	.00	000	ACRE		
Mean:		1008	.18	318	ACRE		
Number	Missing:	4	2 ((79.	.25%)		
Number Not Missing: 11 (20.75%)							
Total period: 1945 to 1997							
Non-missing data period: 1945 to 1997							

Command Reference: FillInterpolate()

Fill missing time series data by interpolating between known values

Version 09.07.02, 2010-08-20

The FillInterpolate() command fills missing data in a time series by interpolating between known values within the same time series. The command currently will not extrapolate past end points.

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

🚯 Edit FillInterpolate() Command 🛛 🔀					
Fill time series missing values by interpolating between non-missing values.					
Filling by extrapolating past known end points is not currently implemented.					
If the maximum intervals to fill is 0, then interpolation will be used regardless of the data gap.					
TS list:	AllMatchingTSID V Optional - indicates the time series to process (default=AllTS).				
TSID (for TSList=AllMatchingTSID):	06707500.DWR.Streamflow.Month				
EnsembleID (for TSList=EnsembleID):	~				
Fill start date/time:		Optional - start of period to fill(default=fill all).			
Fill end date/time:		Optional - end of period to fill (default=fill all).			
Maximum intervals to fill:	3	Optional (default=0 to fill all).			
Transformation for interpolation:	None 🔽	Optional - currently only default value (None) is recognized.			
Fill flag:		Optional - string to flag filled values (default=no flag).			
Command:	FillInterpolate(TSList=AllMatchingTSID,TSID="06707 500.DWR.Streamflow.Month",MaxIntervals=3,Transform ation=None)				
Cancel OK					

FillInterpolate() Command Editor

The command syntax is as follows:

FillInterpolate(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of:	Allts
	 AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will be modified. AllTS – all time series before the 	

Parameter	Description	Default
	 command. EnsembleID – all time series in the ensemble will be modified. FirstMatchingTSID – the first time series that matches the TSID (single TSID or TSID with wildcards) will be modified. LastMatchingTSID – the last time series that matches the TSID (single TSID or TSID with wildcards) will be modified. SelectedTS – the time series are those selected with the SelectTimeSeries() command 	
TSID	The time series identifier or alias for the time series to be modified, using the * wildcard character to match multiple time series.	Required when TSList=*TSID
EnsembleID	The ensemble to be modified, if processing an ensemble.	Required when TSList=EnsembleID.
FillStart	The starting date/time for the fill.	Available period.
FillEnd	The ending date/time for the fill.	Available period.
MaxIntervals	The maximum number of consecutive intervals to fill (0 indicates no limits on the number of consecutive intervals that can be filled).	0
Transformation	Indicate the data transformation to occur for interpolation. Currently, None is the only option and is the default. Earlier versions used Linear.	None (no transformation).
FillFlag	A string to flag data values that are filled.	None – do not flag filled data.

A sample command file using data from the State of Colorado's HydroBase is as follows:

```
# 06707500 - SOUTH PLATTE RIVER AT SOUTH PLATTE
06707500.DWR.Streamflow.Month~HydroBase
FillInterpolate(TSList=AllMatchingTSID,TSID="06707500.DWR.Streamflow.Month",
MaxIntervals=3,Transformation=None)
```
Command Reference: FillMixedStation()

Fill missing data in dependent time series using the best fit from 1+ independent time series, using OLS regression or MOVE2, data transforms, one/monthly equations

This command is under development. It is envisioned that the FillRegression() command enhancements will be completed first. Then the FillMixedStation() analysis command will utilize much of the same logic, using the output statistics table to eliminate candidate relationships and use the remaining relationships to calculate estimated values to check against standard error of prediction, etc.

The FillMixedStation() command fills missing data in a time series where one or more independent time series is used to sequentially fill missing data. This approach has been developed to automate analysis of regression filling (see **Mixed Station Analysis Tool** below) and to facilitate batch filling of many related time series. This implementation is based on the Mixed Station Model implemented for Colorado's Decision Support Systems (Ayres Associates, 2000), which was based on the similarly named approach implemented by the USGS (Alley and Burns, 1981).

The time series involved in the analysis are typically related, such as being from nearby locations in a region. The main uses of the command are:

- 1. To automatically fill every time series in a data set, using other time series in the data set. For example, for hydrologic modeling natural flow time series may have been estimated by processing measured streamflow, diversion, and reservoir time series. The natural flow time series can be filled for use in modeling.
- 2. To generate a report on relationships, so that the user can configure individual FillRegression() and FillMOVE2() commands in TSTool. This may be appropriate when using FillMixedStation() on a list of time series is inappropriate.

Important: TSTool does not automatically exclude time series that have been filled in previous steps. Consequently, care must be taken when specifying the list of independent time series to NOT use time series that were filled in a previous step.

For each dependent time series being filled, the Mixed Station Analysis (MSA) selects the independent time series and parameters that result in the best filling results, considering combinations of the following:

- The list of independent time series being considered can be constrained to a subset of available time series.
- Filling methods include ordinary least squares (OLS) regression (see the FillRegression() command for details) and MOVE2 (see the FillMOVE2() command for details).
- One equation or monthly equations can be used. However, both options cannot be evaluated together due to the complexity of ranking and reporting results.
- The data can be transformed using log_{10} , or no transformation can be applied.
- A minimum number of overlapping data points (sample size N1) can be specified to indicate a valid relationship.
- A minimum correlation coefficient *r* can be specified to indicate a valid relationship.

- A minimum confidence level for the slope of the regression line can be specified (see T-Test discussion below).
- The best fit indicator can be the correlation coefficient (*R*), or the standard error of prediction (SEP, described below).

Because extensive analysis may be necessary to evaluate all the combinations of parameters, the FillMixedStation() command will be slower than other commands that specifically indicate how to perform the filling. The number of combinations can also be limited by reducing the number of parameter options and using stricter limitations on the number of overlapping points and correlation coefficients that are required for a good regression result.

The full MSA process is as follows:

- 1. For each dependent time series, perform a regression analysis using a unique combination of parameters (e.g., use an independent time series, OLS regression with one equation, no data transform). This results in 1+ regression results for each dependent time series.
- 2. Qualifying results (those that meet the requirements of minimum number of overlapping points and correlation coefficient) are retained in a list for the dependent time series, for processing in the next step.
- 3. The qualifying results are used to estimate each missing value. Typically, the SEP is used to select the relationship to use (the one that has lowest SEP).
- 4. Missing data in the dependent time series are filled using the regression results for he selected relationship. If missing values remain, the next highest ranking regression result is used until all missing values are filled (or no additional qualifying regression results are available). Monthly filling occurs on each of the 12 months. This approach may use different stations because of the goodness of fit of the relationship and because different stations may or may not have data that overlap the period to be filled.

Implementation in Colorado's Decision Support Systems

The Mixed Station Model implemented for the State of Colorado typically used the following input:

- Log transform
- Monthly relationships
- Rank on SEP
- Ordinary lease squares regression
- Minimum concurrent values = 5
- Confidence level = 95%
- Fill all time series in data set

The following dialog is used to edit the FillMixedStation() command and illustrates the syntax of the command. Note that this interface will be updated to be similar to that of the FillRegression() command.

🜌 Edit FillMixedStation() Command						
This command determines the best fit to fill the d	This command determines the best fit to fill the dependent time series with data from the independent time series, and performs the filling.					
The dependent and independent time series can The working directory for files is: C:\Develop\TST	be selected using the TS list parameters. 'ool_SourceBuild\TSTool\test\regression\cu	ormands) general) FillMixedStation				
Dependent TS list:		Ontional - indicates the time series to process (default=AllTS).				
Dependent TSID (for TSList=AllMatchingTSID);						
Independent TS list:	V	Optional - time series used to fill (default=AlITS).				
Independent TSID (for TSList=AllMatchingTSID):						
Best Fit:	SEP 🔽	Required - best fit indicator, for ranking output.				
Analysis method(s):	MOVE2 OLSRegression	Optional - method(s) to use in analysis (default=OLSRegression).				
Number of equations:	MonthlyEquations 🖌	Optional - number of equations to use in the analysis (default=OneEquation).				
Transformation(s):	Log None	Optional - transformation(s) to use in analysis (default=None).				
Intercept:		Optional - 0.0 is allowed with Transformation=None (default=no fixed intercept).				
Confidence level:	95	Optional - confidence level (%) for line slope (default=do not consider level).				
Analysis period:		to				
Fill period:		to				
Minimum data count:	5	Optional - minimum number of overlapping points required for analysis (default=10).				
Minimum R:	.7	Optional - minimum correlation coefficient R required for a best fit (default = 0.5).				
Fill flag:	Auto	Optional - string (or "Auto") to indicate filled values (default=no flag).				
Output file:	Results/Test_FillMixedStation_gunnv.xbg	g_MonthlyEquations_out.txt Browse				
FillMixedStation(Be	stFitIndicator=SEP,Analys	isMethod="OLSRegression",NumberOfEquations=MonthlyEquations,Tra				
nsformation="Log", ConfidenceLevel=95, MinimumDataCount=5, MinimumR=.7, FillFlag="Auto", OutputFile="Results/Tes						
t_FillMixedStation_gunnv.xbg_MonthlyEquations_out.txt")						
		EillMiyadStation				

FillMixedStation() Command Editor

The command syntax is as follows:

FillMixedStation(Parameter=value,...)

Command Parameters

Parameter	Description	Default
DependentTSList	Indicates the list of independent time series to be processed one of:	None – must be specified.
	 AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will be processed. AllTS – all time series before the command will be processed. 	
	 EnsembleID – all time series in the ensemble will be processed. FirstMatchingTSID – the first time series that matches the TSID (single TSID or TSID with wildcards) will be processed. 	

Parameter	Description	Default
DopondontTSID	 LastMatchingTSID - the last time series that matches the TSID (single TSID or TSID with wildcards) will be processed. SelectedTS - the time series selected with the SelectTimeSeries() command will be processed. 	Poquired if
ренаенствтр	dependent time series to be processed, using the * wildcard character to match multiple time series.	DependentTSList= *TSID.
IndependentTSList	 Indicates the list of independent time series to be considered for each dependent time series, one of: AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will be processed. AllTS – all time series before the command will be processed. EnsembleID – all time series in the ensemble will be processed. FirstMatchingTSID – the first time series that match the TSID (single TSID or TSID with wildcards) will be processed. EnsembleID – all time series in the ensemble will be processed. FirstMatchingTSID – the first time series that matches the TSID (single TSID or TSID with wildcards) will be processed. LastMatchingTSID – the last time series that matches the TSID (single TSID or TSID with wildcards) will be processed. SelectedTS – the time series selected with the SelectTimeSeries() command will be processed. 	None – must be specified.
IndependentTSID	The time series identifier or alias for the independent time series to be compared, using the * wildcard character to match multiple time series.	Required if IndependentTSList= *TSID.

Parameter	Description	Default
BestFitIndicator	Specifies the indicator to use when determining	SEP
	the best fit, one of:	
	 R – correlation coefficient 	
	 SEP – Standard Error of Prediction, defined 	
	as the square root of the sum of differences	
	between the known dependent value, and	
	the value determined from the equation of	
	best fit at the same point.	
	 SEPTransformedpoin – same as SEP; 	
	however the data values have first been	
	transformed as per the Transformation	
	parameter.	
	 SEPTotal, when used with one equation, 	
	it is the same as SEP. When used with	
	monthly equations, it is the SEP considering	
	all months.	
	 SEPTransformedTotal, when used 	
	with one equation, it is the same as	
	SEPTransformed. When used with	
	monthly equations, it is the	
	SEPTransformed considering all	
	months.	
Analysismethod	specify the method(s) to analyze the data, in order to determine the best fit including	OLSRegression
	Of CD opposed of an and/or MOVE2. If multiple	
	methods are specified separate with commas	
	and surround with double quotes	
NumberOfEquations	The number of equations to use for the analysis:	None – must be specified
	OneEquation or MonthlyEquations.	indie mase se speemed.
	Only one may be chosen. If necessary, use	
	more than one command to use different	
	parameter combinations for different groups of	
	time series.	
Transformation	Indicates how to transform the data before	None (no transformation)
	analyzing. Specify as None (no transformation)	
	or Log (for Log_{10}). If the Log option is used,	
	zero and negative values in data are set to	
	.001. Missing data are ignored. If multiple	
	values are selected, separate with a comma and	
	surround with double quotes.	
Intercept	Specify as 0 to force the intercept of the best-fit	Do not force the intercept
	line through the origin. This is made available	through zero.
	only for OLS regression analysis on	
	untransformed data, to be consistent with the	
	FillRegression() command.	
ConfidenceLevel	Required confidence level for the T-Test on the	No limit on confidence
	regression slope. Relationships not passing the	level.
	test are not allowed for filling.	

Parameter	Description	Default
AnalysisStart	The date/time to start the analysis, to focus on a	If blank, analyze the full
	period appropriate for analysis. For example,	period.
	specify the unregulated period for streamflow.	
AnalysisEnd	The date/time to end the analysis.	If blank, analyze the full
		period.
FillStart	The date/time to start filling, if other than the	If blank, fill the full
	full time series period.	period.
FillEnd	The date/time to end filling, if other than the full	If blank, fill the full
	time series period.	period.
MinimumDataCount	The minimum number of overlapping data	10
	points that are required for a valid analysis (N1	
	in FillRegression() and FillMOVE2()	
	documentation). If the minimum count is not	
	met, then the independent time series is ignored	
	for the specific combination of parameters. For	
	example, if monthly equations are used, the	
	independent time series may be ignored for the	
	specific month; however, it may still be	
	analyzed for other months.	
MinimumR	The minimum correlation coefficient required	0.5
	for a best fit. If the minimum is not met, then	
	the results are not considered in the best fit	
	ranking or filling.	
OutputFile	Output file for the results, either as a file name	If not specified, partial
	to be written to the working directory, or a full	results of the analysis may
	path.	be available in the log file.

The following example command file fills natural flow time series from a StaeMod file using one equation (not monthly):

```
# Test filling the gunnison monthly baseflow time series with
# Mixed Station Analysis (all combinations for one equation)
StartLog(LogFile="fill-baseflow.log")
ReadStateMod(InputFile="gunnv.xbg")
FillMixedStation(BestFitIndicator=SEP,AnalysisMethod="MOVE2,OLSRegression",
NumberOfEquations=OneEquation,
Transformation="Log,None",OutputFile="Results.txt")
# Check for missing data - all should be filled
CheckTimeSeries(CheckCriteria="Missing",MaxWarnings=10)
# Check for negative flows - should not be any
CheckTimeSeries(CheckCriteria="<",Value1=0,MaxWarnings=10)</pre>
```

Command Reference: fillMOVE1()

Fill Missing Time Series Data Using MOVE1 Procedure

Version 06.08.02, 2004-08-02, Color, Acrobat Distiller

The fillMOVE1() command has not been enabled. This documentation serves as a reference for the MOVE1 procedure. Refer to the fillMOVE2() command.

The fillMOVE1() command is more sophisticated than the fillRegression() command.

Maintenance of variance extension (MOVE) procedures are methods of fitting straight lines to data. The slope and intercept of the MOVE equations are computed differently than in ordinary least squares (OLS) regression (see the fillRegression() command for a discussion of OLS regression). As shown below, an area of a triangle is minimized in the MOVE procedures rather than a vertical distance as in OLS regression. The MOVE procedures do not provide the minimum-variance estimate of a single value but an ensemble of points estimated by the MOVE procedures will have the same variability as the true values.

MOVE procedures are useful in extending record at gaging stations where the extended record will be subsequently used in another analysis such as frequency analysis. MOVE procedures will provide about the same estimates as OLS regression near the mean of the data but will provide smaller and larger estimates at the extremes of the data set. The slope of the MOVE relation is steeper than OLS regression. The MOVE procedures are based on only one independent variable and the assumption is that there is a linear relation between the dependent and independent variables. If the untransformed data are not linearly related, then it is common to transform the data using a logarithmic transformation.

The MOVE.1 procedure uses just the data from the N_1 years of concurrent data. The MOVE.2 procedure (see the fillMOVE2() command) uses the Two-Station Comparison procedure described in **Appendix 7 of Bulletin 17B, Guidelines for Determining Flood Flow Frequency, USGS**, to compute improved estimates of the mean and variance for the dependent time series and uses all the data at the dependent time series to estimate the mean and variance of the dependent time series. The MOVE.2 procedure has been shown to be marginally better than MOVE.1.



Maintenance of Variance Extension (MOVE)

The MOVE.1 equation is used to estimate values for the dependent time series from the independent time series:

$$Y_{i} = \overline{Y}_{1} + \frac{S_{y1}}{S_{x1}} \left[X_{i} - \overline{X}_{1} \right]$$

or

$$Y_i = a + bX_i$$

where

 N_{1} = concurrent or overlapping period of record \overline{X}_{1} = mean for independent variable for N_{1} years \overline{Y}_{1} = mean for dependent variable for N_{1} years S_{y1} = standard deviation for N_{1} years S_{x1} = standard deviation for N_{1} years

$$b = \frac{S_{y1}}{S_{x1}}$$
$$a = \overline{Y}_1 - b\overline{X}_1$$

Note that the slope of the line does not include the correlation coefficient. This is the only difference between OLS regression and MOVE.1.

Command Reference: FillMOVE2() Fill missing data in time series using the Maintenance of Variance Extension (MOVE.2) procedure

The FillMOVE2() command fills missing data in a time series using the MOVE.2 procedure (see the FillMOVE1() command for background information). The MOVE.2 procedure uses the Two-Station Comparison procedure described in **Appendix 7 of Bulletin 17B**, **Guidelines for Determining Flood Flow Frequency, USGS**, to compute improved estimates of the mean and variance at the dependent or short-term station and uses all the data at the dependent time series to estimate the mean and variance of the dependent time series. The MOVE.2 procedure has been shown to be marginally better than MOVE.1. The following MOVE.2 equation is used to estimate values for the dependent time series from the independent time series:

$$Y_{i} = \overline{Y} + \frac{S_{y}}{S_{x}} \left[X_{i} - \overline{X} \right]$$

where

 Y_i = discharge for dependent time series

 X_i = discharge for independent time series

 \overline{X} = mean for independent time series for $N_1 + N_2$ years (N₂ is the additional years in the long-term time series)

 S_x = standard deviation for independent time series for $N_1 + N_2$ years

 $\overline{Y} = \overline{Y_1} + \frac{N_2}{N_1 + N_2} \left[b \left(\overline{X_2} - \overline{X_1} \right) \right]$ (Equation 7-5a for Two-Station Comparison in Appendix 7 of Bulletin 17B)

$$\mathbf{S}_{y}^{2} = \frac{1}{(N_{1} + N_{2} - 1)} \left[(N_{1} - 1)S_{y1}^{2} + (N_{2} - 1) b^{2}S_{x2}^{2} + \frac{N_{2}(N_{1} - 4)(N_{1} - 1)}{(N_{1} - 3)(N_{1} - 2)} (1 - r^{2})S_{y1}^{2} + \frac{N_{1}N_{2}}{N_{1} + N_{2}} b^{2}(\overline{X}_{2} - \overline{X}_{1})^{2} \right]$$

(Equation 7-10 for Two-Station Comparison in **Appendix 7 of Bulletin 17B**) where

 $b = r \frac{S_{y1}}{S_{x1}}$, r = correlation coefficient (Note that b is the slope of the ordinary least squares regression

line.)

 N_{1} = concurrent or overlapping period of record

 N_{2} = additional years available at long-term site

 \overline{X}_1 = mean of independent time series for N_1 years

 \overline{X}_2 = mean of independent time series for N_2 years

 S_{y1} = standard deviation of dependent time series for N_1 years

 S_{x1} = standard deviation of independent time series for N_1 years

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

🗼 Edit FillMOVE2() command 🛛 🛛 🔀						
See the TSTool documentation	See the TSTool documentation for a description of the MOVE2 procedure.					
The analysis period(s) will be u	used to determine the relationship:	s used for filling.				
Use a setOutputPeriod() comm	and if the dependent time series p	eriod will be extended.				
Specify dates with precision a	ppropriate for the data, use * for a	all available data, OutputStart, or OutputEnd.				
Time series to fill (dependent):	06758500.DVVR.Streamflow.Mon	th				
Independent time series:	06754000.DVVR.Streamflow.Mon	th				
Number of equations:	MonthlyEquations	Number of equations to use (blank=one equation).				
Transformation:		How to transform data before analysis (blank=None).				
Dependent analysis period:	1952-10	to 2004-09				
Independent analysis period:	1901-01	to 1950-12				
Fill Period:	1930-01	to 1940-12				
Fill flag:	m	1-character flag to indicate fill.				
Command:	<pre>FillMOVE2(TSID="06758500.DWR.Streamflow.Month",IndependentTSID="0 6754000.DWR.Streamflow.Month",NumberOfEquations=MonthlyEquations, DependentAnalysisStart="1952-10",DependentAnalysisEnd="2004-09",I nd: ndependentAnalysisStart="1901-01",IndependentAnalysisEnd="1950-12",FillStart="1930-01",FillEnd="1940-12",FillFlag="m")</pre>					
Cancel OK						

FillMOVE2() Command Editor

The command syntax is as follows:

FillMOVE2(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSID	The time series identifier or alias for the	None – must be specified.
	time series to be filled (dependent time	
	series).	
IndependentTSID	The time series identifier or alias for the	None – must be specified.
	independent time series, to supply data.	
NumberOf	OneEquation or	OneEquation
Equations	MonthlyEquations, indicating how	
	many relationships are to be determined.	
Transformation	Log or None, indicating the type of data	None
	transformation. If the Log option is	
	used, zero and negative values are set to	
	.001 (-999 values are treated as	
	missing data and are ignored), and the	
	data values are transformed using log10.	
Dependent	The period for N_1 (overlapping data) that	Analyze the full period.
Analysis	is used to analyze the dependent time	
Start/End	series. For example, this may be the	
	unregulated period for streamflow data.	
	Typically, this is longer than the	
	independent analysis period.	
Independent	The period for N_2 (non-overlapping data)	Analyze the full period.
Analysis	that is used to analyze the independent	
Start/End	time series. For example, this may be the	
	unregulated period for streamflow data.	
FillStart	The date/time to start filling.	Fill the full period.
FillEnd	The date/time to end filling.	Fill the full period.
FillFlag	A single character to be used to flag	Do not flag filled data.
	filled points on graphs and other output.	

A sample command file illustrating how to fill time series from the State of Colorado's HydroBase is as follows (MOVE2 and ordinary least squares regression are used to allow comparing the results):

```
StartLog(LogFile="Results/commands.TSTool.log", Suffix="Date")
SetOutputPeriod(OutputStart="1901-01",OutputEnd="2004-12")
# 06758500 - SOUTH PLATTE RIVER NEAR WELDONA
06758500.DWR.Streamflow.Month~HydroBase
# 06754000 - SOUTH PLATTE RIVER NEAR KERSEY
06754000.DWR.Streamflow.Month~HydroBase
FillMOVE2(TSID="06758500.DWR.Streamflow.Month",
  IndependentTSID="06754000.DWR.Streamflow.Month",
  NumberOfEquations=MonthlyEquations, DependentAnalysisStart="1952-10",
  DependentAnalysisEnd="2004-09", IndependentAnalysisStart="1901-01",
  IndependentAnalysisEnd="1950-12",FillStart="1930-01",
  FillEnd="1940-12",FillFlag="m")
# 06758500 - SOUTH PLATTE RIVER NEAR WELDONA
06758500.DWR.Streamflow.Month~HydroBase
# 06754000 - SOUTH PLATTE RIVER NEAR KERSEY
06754000.DWR.Streamflow.Month~HydroBase
FillRegression(TSID="06758500.DWR.Streamflow.Month",
IndependentTSID="06754000.DWR.Streamflow.Month")
```

Command Reference: FillPattern()

Fill missing time series data using historical average patterns

Version 08.16.04, 2008-09-19

The FillPattern() command fills missing data in a time series using historic averages based on a pattern file. For example, if May 1910 is missing and the pattern indicates that May 1910 is a WET month, then the average of all WET Mays is used to fill the time series. The pattern file indicates the WET/DRY/AVG patterns and the time series to be filled supplies data to compute averages, for use in filling. This feature is enabled for monthly data only. Averages are computed as described for the FillHistMonthAverage() command. There is currently no way to limit the fill operation to a period (the entire time series is filled). The pattern file is created with the AnalyzePattern() command and a saved file must be read with a ReadPatternFile() command. See below for an example of a fill pattern file. One or more patterns can be included in each pattern file, similar to StateMod time series files (see the StateMod Input Type appendix), and multiple pattern files can be used, if appropriate.

# Y # M	ears Shown = W issing monthly	Mater Years ' data fill	ed by th	ne Mixed	Station	Method,	USGS 19	89						
# T	ime series ide	ntifier	_ =	09034500).CRDSS_U	JSGS.QME	MONTH.1							
# D	escription		=	COLORADO) RIVER A	AT HOT SU	JLPHUR SI	PRINGS, C	20.					
# -е	-beb	eb	eb	eb	eb	eb	eb	eb	eb	eb-	eb-	eb	eb	e
1	0/1908 -	9/1996 AC	FT WYR											
1909	09034500	AVG	AVG	AVG	WET	WET	AVG	AVG	AVG	WET	WET	WET	WET	
1910	09034500	WET	WET	WET	WET	WET	WET	AVG	AVG	AVG	AVG	AVG	AVG	
1911	09034500	AVG	AVG	WET	AVG	AVG	AVG	AVG	WET	WET	WET	AVG	WET	
1912	09034500	WET	WET	WET	WET	WET	AVG	AVG	WET	WET	WET	WET	WET	
0	mmitted													

The following dialog is used to edit the FillPattern() command and illustrates the syntax of the command.

Edit FillPattern() Command					
Monthly time series can be filled usin	g historical average patterns (e.g., WET, DRY, AVG climate patterns).				
Patterns are defined with ReadPatternFile() command(s).					
TS list:	AllMatchingTSID Indicates the time series to process (default=AIITS).				
TSID (for TSList=AllMatchingTSID):	30*				
EnsembleID (for TSList=EnsembleID):					
Fill pattern ID:	9034500 Pattern ID used for filling.				
Command:	FillPattern(TSList=AllMatchingTSID,TSID="30*",Patte rnID="9034500")				
	Cancel OK				

FillPattern() Command Editor

The command syntax is as follows:

```
FillPattern(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
TSList	 Indicates the list of time series to be processed, one of: AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will be modified. AllTS – all time series before the command. EnsembleID – all time series in the ensemble will be modified. FirstMatchingTSID – the first time series that matches the TSID (single TSID or TSID with wildcards) will be modified. LastMatchingTSID – the last time series that matches the TSID (single TSID or TSID with wildcards) will be modified. SelectedTS – the time series are those selected with the SelectTimeSeries() command. 	AllTS
TSID	The time series identifier or alias for the time series to be modified, using the * wildcard character to match multiple time series.	Required for TSList=*TSID.
EnsembleID	The ensemble to be modified, if processing an ensemble.	Required for TSList=EnsembleID.
PatternID	The pattern identifier, matching a pattern read with ReadPatternFile() commands.	None – must be specified.

A sample command file to process data from the State of Colorado's StateMod model is as follows:

```
# Read StateMod time series to fill
ReadStateMod(InputFile="..\StateMod\sjm_prelim.ddh")
# Read the file containing the patterns
ReadPatternFile(PatternFile="fill.pat")
# Fill time series having identifiers that start with "30"
FillPattern(TSList=AllMatchingTSID,TSID="30*",PatternID="09034500")
# Write the results
WriteStateMod(TSList=AllTS,OutputFile="..\StateMod\sjm.ddh")
```

The above example fills all diversion time series with identifier starting with 30, using the pattern 09034500 (a stream gage for the region).

Command Reference: FillPrincipalComponent Analysis() Fill missing time series data using principal component analysis (PCA) Version 09.04.00, 2009-06-11

This command is under development.

This page is intentionally blank.

Command Reference: FillProrate() Fill missing time series data by prorating values in another time series

Version 08.16.04. 2008-09-30

The FillProrate() command fills missing data in time series by prorating values from another time series. This fill technique is useful, for example, where two time series are likely to have the same general trend and ratio of data values. The ratio can be computed two ways, as specified by the FactorMethod parameter:

- NearestPoint causes the ratio to be recomputed each time that a non-missing value is found in both time series. The ratio computed from the nearest points in each time series is used for filling until another value can be computed.
- AnalyzeAverage computes the ratio as the average ratio of the time series (numerator) and the independent time series (divisor). This was implemented to match an existing fill procedure but can lead to some bias in the results. A different overall average will be obtained depending on whether ratios are computed first and then averaged than if the sum of the numerators are added and divided by the sum of the denominators. In the former, the choice of which time series is in the denominator could impact results. More parameters may need to be added in the future to implement an analysis different from the current defaults.

The initial computation of the ratio may require specifying an initial value due to missing data on the endpoints of the time series (see the InitialValue parameter). Alternatively, the time series can be filled in one direction first and then filled in the other direction with a second command. The following dialog is used to edit the command and illustrates the syntax of the command:

🛃 Edit FillProrate() Command		X		
This command fills missing data in time series by prorating values from an independent time series.				
The proration factor is calculated in o	ne of the following way	/s (indicated by "FactorMethod"):		
1) Recompute the factor at each poir	nt where a non-missing	value is found in both time series (NearestPoint).		
The initial value in the filled time se	ries is used to compute	the initial factor, for missing data at the end of the fill period.		
2) Recompute the factor as the depe	ndent time series value	divided by the average of independent values (AnalysisAverage).		
The factor (ratio) is TS/IndependentTS	S and the filled value is	calculated as factor*IndependentTS.		
If the independent time series data va	lue is zero, the factor r	emains as previously calculated to avoid division by zero.		
The independent time series is not its	elf filled if matched for t	he TSList.		
The fill start and end, if specified, will	limit the period that is fi	lled.		
Use standard date/time formats appro	priate for the date/time	precision of the time series.		
TS List:	AllMatchingTSID	Indicates the time series to process (default=AlITS).		
TSID (for TSList=AllMatchingTSID):	06754000.DWR.Stream	nflow.Month		
EnsembleID (for TSList=EnsembleID):				
Independent time series:	06694700.USGS.Streamflow.Month			
Fill start date/time:		Optional - start date/time for filling (blank=fill all).		
Fill end date/time:		Optional - end date/time for filling (blank=fill all).		
Fill flag:		Optional - one-character flag to mark filled data.		
Fill direction:	Forward 💌	Optional - direction to traverse data when filling (default=Forward).		
Calculate factor how?:	_	Optional - how will factor be calculated? (default=NearestPoint).		
Analysis start date/time:		Optional - analysis start date/time, for FactorMethod=AnalyzeAverage).		
Analysis end date/time:		Optional - analysis end date/time, for FactorMethod=AnalyzeAverage).		
Initial value:	0 🔹	Optional - initial value in time series for missing end-points.		
Command:	FillProrate(TSList=AllMatchingTSID,TSID="06754000.DWR.Streamflow. Month",IndependentTSID="06694700.USGS.Streamflow.Month",FillDirec d: tion=Forward,InitialValue=0)			
Cancel OK				

FillProrate() Command Editor

The command syntax is as follows:

FillProrate(Parameter=Value,...)

Command Parameters

Parameter	Description	Default		
TSList	Indicates the list of time series to be	AllTS		
	processed, one of:			
	• AllMatchingTSID – all time			
	series that match the TSID (single			
	TSID or TSID with wildcards) will be			
	modified.			
	• AllTS – all time series before the			
	command.			
	• EnsembleID – all time series in the			
	ensemble will be modified.			
	• FirstMatchingTSID – the first			
	time series that matches the TSID			
	(single TSID or TSID with wildcards)			
	will be modified.			
	• LastMatchingTSID – the last			
	time series that matches the TSID			
	(single ISID of ISID with wildcards)			
	will be modified.			
	• Selected 15 - the time series are			
	Sologt TimoSoriog() command			
TSID	The time series identifier or alias for the	Required for TSLigt - *TSTD		
	time series to be modified. Use the *			
	wildcard character to match multiple time			
	series.			
EnsembleID	The ensemble to be modified, if	Required for		
	processing an ensemble.	TSList=EnsembleID.		
IndependentTSID	The time series identifier or alias for the	None – must be specified.		
	independent time series.			
FillStart	The starting date/time for the fill.	Available period.		
FillEnd	The ending date/time for the fill.	Available period.		
FillFlag	A one-character flag to tag data values	None – do not flag filled data.		
	that are filled.	Democrat		
FILIDIFECTION	Specify the direction of the fill as	Forward		
EastorMothod	Forward of Backward.	NearogtDoint		
Factormethod	in protection, one of:	Nearescroinc		
	Analyza Average calculate the			
	• AnalyzeAverage – calculate the factor of the average of the time series			
	divided by the independent time			
	series using the analysis period			
	 Nearest Point - calculate the 			
	factor at the nearest point where both			

Parameter	Description	Default
	time series have non-missing values.	
AnalysisStart	The starting date/time for the analysis, used when FactorMethod =AnalyzeAverage.	Analyze the full period.
AnalysisEnd	The ending date/time for the analysis, used when FactorMethod=AnalyzeAverage.	Analyze the full period.
InitialValue	 The initial value to use for the filled time series, for cases where a value may not be available on the ends of the fill period, one of: NearestBackward – search the time series backward for the nearest non-missing value. NearestForward – search the time series forward for the nearest non-missing value. Specify a number to use for the initial value. 	None – filling will not occur at the end.

A sample command file to fill data from the State of Colorado's HydroBase database is as follows:

```
# 06754000 - SOUTH PLATTE RIVER NEAR KERSEY
06754000.DWR.Streamflow.Month~HydroBase
# 06694700 - FOURMILE CREEK NEAR FAIRPLAY, CO.
06694700.USGS.Streamflow.Month~HydroBase
FillProrate(TSList=AllMatchingTSID,TSID="06754000.DWR.Streamflow.Month",
IndependentTSID="06694700.USGS.Streamflow.Month",FillDirection=Forward,
InitialValue=0)
06754000.DWR.Streamflow.Month~HydroBase
```

Command Reference: FillRegression() Fill missing time series data using ordinary least squares regression

The FillRegression() command fills missing data in a time series using ordinary least squares (OLS) regression and provides a variety of options for transforming the data and controlling the analysis. In OLS regression, the vertical distance from the data point to the regression line is minimized. OLS regression provides the minimum-variance estimate for a single value or observation. However, if an ensemble of points is estimated from OLS regression, the estimated values will have lesser variability than the true values.



See also the FillMOVE2() command, which utilizes additional variance from independent time series to determine the regression relationship, and the FillMixedStation() command, which automates the analysis of many time series to determine a "best estimate" filling approach. Regression can be applied only to regular interval time series. The dependent time series will be filled using the independent time series. The periods of record and output period for the time series should be verified to make sure that the time series periods overlap sufficiently. Regression relationships are developed using the analysis period for the time series and are applied to the fill period. Refer to the output statistics table, log file, and time series properties for analysis details. Several parameters are available to ensure that filling uses reasonable relationships. This command has functionality that may not be needed for simple analysis but which is useful for software testing and comparison with the FillMixedStation() command.

Important: TSTool does allow filled values to be flagged. However, other commands do not exclude these values from computations when determining relationships for subsequent fill steps. Therefore, it is important to perform regression data filling as early in data processing as possible so that data manipulation does not introduce derived values and bias.

The following OLS equation is used to estimate values for the dependent time series from the independent time series:

$$Y_{i} = \bar{Y}_{1} + \bar{r} \frac{S_{y1}}{S_{x1}} \left[X_{i} - \bar{X}_{1} \right]$$

or

 $Y_i = a + bX_i$

where

 N_1 = concurrent or overlapping period of record (the notation N_1 is used because the MOVE2 fill technique refers to N_2 , which is the number of additional points outside of N_1 in the independent time series)

$$\overline{X}_{1} = \text{mean for independent variable for } N_{I} \text{ years} = \frac{\sum X_{1_{i}}}{N_{1}}$$

$$\overline{Y}_{1} = \text{mean for dependent variable for } N_{I} \text{ years} = \frac{\sum Y_{1_{i}}}{N_{1}}$$

$$S_{y1} = \text{standard deviation for } N_{I} \text{ years} = \sqrt{\frac{1}{N_{1} - 1} \sum (Y_{1_{i}} - \overline{Y_{1}})^{2}}$$

$$S_{x1} = \text{standard deviation for } N_{I} \text{ years} = \sqrt{\frac{1}{N_{1} - 1} \sum (X_{1_{i}} - \overline{X_{1}})^{2}}$$

$$r = R = \text{correlation coefficient} = \frac{N_{1} \sum X_{1_{i}} Y_{1_{i}} - \sum X_{1_{i}} \sum Y_{1_{i}}}{\sqrt{[N_{1} \sum X_{1_{i}}^{2} - (\sum X_{1_{i}})^{2}]} \cdot [N_{1} \sum Y_{1_{i}}^{2} - (\sum Y_{1_{i}})^{2}]}$$

$$b = r \frac{S_{y1}}{S_{x1}}$$
$$a = \overline{Y}_1 - b\overline{X}_1$$

The correlation coefficient, *r*, is used to compute the slope, *b*, of the line.

A number of statistics are computed and are available for output to a table, as described below (see the TableID and related command parameters for how to specify the table output). Creating a statistics table and then writing the table to a file is useful for checking the analysis and software. For example, the CompareTables() command can be used to compare this statistics table with a verification data set that is calculated by another tool. In the following descriptions, the statistic for one equation has a name like Mean and monthly statistics correspondingly have a name like Mean_1, where 1 corresponds to January and 12 to December.

In some cases, statistics are relevant in units of the raw values, in some cases statistics are relevant in transformed (log10) units, and in some cases both are relevant. For example, if the log10 transform is used to compute the relationship, then a and b are in transformed units. However, error computations between the original data values and values that would be computed by the relationship are in the raw units (regardless of whether the data were transformed) – this allows errors to be compared between relationships using raw and transformed values (the FillMixedStation() command uses this information to compare relationships). Consequently, the third column of the following table indicates whether statistics are provided in raw (column name uses statistic only) or transformed units (additional _trans added to statistic for column name). Therefore, if the statistic is unitless, it will never have the _trans addition. If the analysis does not use a transformation, then _trans will be omitted from column headings.

Statistic	Involves	Statistics	
(Table	Dependent,	Output in Raw	
Column	Independent,	or Transformed	
Name)	or Both	units	Description
Nl	Both	N/A - unitless	The number (count) of non-missing data values
			overlapping in the dependent and independent time
			series.
MeanX1	Independent	raw, transformed	The mean of the independent N1 data values.
SX1	Independent	raw, transformed	The standard deviation of the independent N1
			values.
N2	Independent	N/A - unitless	The number (count) of non-missing independent
			values outside of N1.
MeanX2	Independent	raw, transformed	The mean of the independent N2 values.
SX2	Independent	raw, transformed	The standard deviation of the independent N2
			values.
MeanY1	Dependent	raw, transformed	The mean of the dependent N1 values.
SY1	Dependent	raw, transformed	The standard deviation of the dependent N1 values.
NY	Dependent	N/A - unitless	The total number of non-missing dependent values.
MeanY	Dependent	raw, transformed	The mean of the dependent NY values.
SY	Dependent	raw, transformed	The standard deviation of the dependent NY values.
SkewY	Dependent	raw, transformed	The skew, or non-symmetry, of the dependent NY
			values.
a	Both	transformed	The intercept for the relationship equation.
b	Both	transformed	The slope of the relationship equation.
R	Both	transformed	The correlation coefficient for N1 values.
R2	Both	transformed	R-squared, coefficient of determination for N1
			values.
MeanYlest	Dependent	raw, transformed	The mean for N1 values computed from the
			relationship (estimate the dependent values where
			values were previously known).
SYlest	Dependent	raw, transformed	The standard deviation for N1 values computed

Statistics From Regression Analysis

Statistic (Table Column Name)	Involves Dependent, Independent, or Both	Statistics Output in Raw or Transformed units	Description
			from the relationship (estimate the dependent at locations where values are known).
RMSE	Dependent	raw, transformed	The "room mean squared error" for N1 overlapping values, which is a measure of the overall error of using the regression equation to estimate values, is calculated as:
			$RMSE = \sqrt{\frac{\sum (Y_{1_{i}} - Y_{1_{i}})^{2}}{N_{1}}}$
			where Y_{l_i} is the original dependent value and Y_{l_i} ' is the value estimated with the regression relationship.
SEE	Dependent	raw, transformed	The standard error of estimate for N1 overlapping values, which is a measure of the overall error of using the regression equation to estimate values, calculated as:
			$SEE = \sqrt{\frac{\sum (Y_{1_i} - Y_{1_i})^2}{N_1 - 2}}$
			where Y_{1_i} is the original dependent value and Y_{1_i} ' is the value estimated with the regression relationship.
SEP	Both	raw	The standard error of prediction for each estimated value, calculated as:
			$S = = \sqrt{1 + \frac{1}{N_1} + \frac{(X_{1_i} - \overline{X}_1)^2}{\sum (X_{1_i} - \overline{X}_1)^2}} * S$
			where X_{1_i} is the original independent value and \overline{X}_{1_i} is the mean of the N1 independent values
			Note when using the mixed station analysis in the $\frac{1}{1}$
			may be used to determine the relationship. The SEP is not actually output in the statistics table but may be added as an optional output time series in the future.
SESlope	Both	N/A - unitless	The standard error (SE) of the slope (b) for N1 overlapping values, calculated as:

Statistic (Table Column Name)	Involves Dependent, Independent, or Both	Statistics Output in Raw or Transformed units	Description
			$S E = \frac{\sqrt{\frac{\sum (Y_{1_i} - Y'_{1_i})^2}{N_1 - 2}}}{\sqrt{\sum (X_{1_i} - \overline{X}_1)^2}}$ where X_{1_i} is the original independent value and \overline{X}_1 is the mean of the N1 independent values;
			Y_{l_i} is the original dependent value and Y_{l_i} is the value estimated with the regression relationship.
TestScore	Both	N/A - unitless	b/SESlope
Test Quantile	Both	N/A - unitless	The value at which the confidence interval is satisfied. Comes from the Student's T-test, which is a function of the confidence interval and degrees of freedom (DF), where DF is the degrees of freedom equal to $N1 - 2$ (corresponding to the intercept and the slope of the regression equation).
Test OK	Both	N/A - unitless	Will be No if TestScore >= TestQuantile, indicating that the b ≠ 0 data are related, and Yes if TestScore < TestQuantile, indicating that the data are not related. If the data are not related, then the relationship between the dependent and independent time series will not be used for filling.
Sample SizeOK	Both	N/A – unitless	Will be No if N1 < MinimumSampleSize and Yes if N1 >= MinimumSampleSize, indicating whether or not the number of overlapping points is greater than or equal to the number of overlapping points necessary.
R OK	Both	N/A – unitless	Will be No if R < MinimumR, indicating that the correlation is below the minimum threshold, and Yes if R >= MinimumR, indicating that the correlation is above the minimum threshold.
NYfilled	Dependent	N/A – unitless	The total number of missing points in the dependent time series that were filled through the regression.
MeanY	Dependent	Raw	The mean of the values that were used to fill
CVF1122	Donondart	Dorr	The standard deviation of the values that were seed.
SILILIEU	Dependent	Kaw	to fill missing points
SkewY filled	Dependent	Raw	The skew, or non-symmetry, of the values that were used to fill missing points

<u>Student's T-distribution</u> (http://en.wikipedia.org/wiki/Student's_t-distribution) is similar to a standard distribution, but has a higher probability of producing outliers. Using the <u>Apache Math</u> library

(http://commons.apache.org/proper/commons-math/javadocs/api-3.2/index.html), the appropriate distribution for the size of the dataset is generated, and the value at which the desired confidence level is satisfied is calculated. For example, if the desired confidence level is .8 and the size of the dataset is seven, then following this graph of the Student's T-distribution, values above approximately one would satisfy the confidence level.





The following dialog is used to edit the command and illustrates the syntax of the command:

• Edit FillRegression() con	nmand	×
This command is in the proces	s of being enhanced to includ	e the check criteria and table output.
Fill missing data using ordinary leas	t squares (OLS) regression.	
The analysis period is used to deter	mine relationships used for filling.	andant time covies. If necessary
Specify dates with precision approp	priate for the data, use blank for all	available data. OutputStart. or OutputEnd.
Data for Analysis		
Time series to fill (dependent):	06753400.USGS.Streamflow.Mont	th 💌
Independent time series:	06753500.USGS.Streamflow.Mont	th 💌
Number of equations:	MonthlyEquations 💙	Optional - number of equations (default=OneEquation).
Analysis month:	~	Optional - use with monthly equations (default=process all months).
Transformation:	Log 💙	Optional - how to transform data before analysis (blank=None).
Value to use when log and <= 0:		Optional - value to substitute when original is <= 0 and log transform (default=0.0010).
Intercept:		Optional - blank or 0.0 are allowed with no transformation.
Analysis start:		Optional - analysis start date/time (default=full period).
Analysis end:		Optional - analysis end date/time (default=full period).
Criteria for Valid Relationships (fil	ing will only occur if criteria are met	t)
Minimum sample size: 10	Optional - minimum number	of overlapping points for relationship (default=not checked).
Minimum R: .5	Optional - minimum correlati	ion coefficient R required for a best fit (default=not checked).
Confidence interval: 95	Optional - confidence interv	val (%) for line slope (default=do not check interval).
Control Filling		
Fill: 💙	Optional - fill missing	values in dependent time series (blank=True, False=analyze only).
Fill start:	Optional - fill start da	te/time (default=full period).
Fill end:	Optional - fill end date	e/time (default=full period).
Fill flag: R	Optional - string to in	dicate filled values.
Fill flag description: Filled with re	gression/log Optional - description	n for fill flag used in reports.
Specify Table for Analysis Statisti	cs Output-	
Table ID for output: Regression	tesults	 Optional - specify to output statistics to table.
Table TSID column: Location		Required if using table - column name for dependent TSID.
Format of TSID: %L	Insert: Select Specifier	Optional - use %L for location, etc. (default=alias or TSID).
FillRegression	n(TSID="06753400.USGS.	.Streamflow.Month", IndependentTSID="06753500.USGS.Str
eamflow.Month'	,NumberOfEquations=Mc	onthlyEquations, Transformation=Log, MinimumSampleSize=
Command: 10, MinimumR=.5	5,ConfidenceInterval=9	95,FillFlag="R",FillFlagDesc="Filled with
regression/log "%L")	<pre>{", laplelD="Regression</pre>	nkesuits", lapieTSiDColumn="Location", TableTSIDFormat=
	L	

FillRegression() Command Editor

FillRegression

The command syntax is as follows:

```
FillRegression(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
TSID	The time series identifier or alias for the time series	None – must be
	to be filled.	specified.
Independent	The time series identifier or alias for the independent	None – must be
TSID	time series.	specified.
NumberOf	The number of equations to use for the analysis:	OneEquation
Equations	OneEquation or MonthlyEquations.	
AnalysisMonth	Indicate the month to process when using monthly	Process all months.
	equations. Currently only a single month can be	
	specified.	
Transformation	Indicates how to transform the data before analyzing.	None (no
	Specify as None (previously Linear) or Log (for	transformation).
	Log_{10}). If the Log option is used, zero and negative	
	values are replaced with the value specified by the	
	LEZeroLogValue parameter value for analysis	
	(missing data values are ignored in the analysis).	
LEZeroLogValue	Value to use for data values less than or equal to zero	.0010
	when using a log transformation. The Log_{10} of this	
	value will be used in calculations.	
Intercept	Specify as 0 to force the intercept of the best-fit line	Parameter is optional and
	through the origin (not available for log	if specified the default is
	transformation).	to not force the intercept
		through zero.
AnalysisStart	The date/time to start the analysis – use to focus on	Analyze the full period.
	only a period appropriate from analysis. For	
	example specify the unregulated period for	
	streamflow.	
AnalysisEnd	The date/time to end the analysis – use to focus on	Analyze the full period.
	only a period appropriate from analysis.	
Minimum	The minimum number of overlapping values	2, due to requirements in
SampleSize	required to use a relationship for filling.	calculating the statistics
MINIMUMR	The minimum correlation coefficient required to use	No check is performed.
Confidence	A confidence interval in percent (e.e. 05) required	The T test is not
Intorval	A confidence interval in percent (e.g., 95) required	ne 1-test is not
IIICEIVAI	for the slope of the relationship. The 1-test is	confidence interval
	dependent time series are related	confidence interval.
	Indicate whether fill should occur (True) or just	True
LTTT	analyze to compute statistics (Eq. 1 dq). The letter is	IIue
	analyze to compute statistics (Faise). The latter is	
	prior to actually performing filling	
FillStart	The date/time to start filling, if other than the full	Fill the full period
FILISCALC	time series period	Fill the full period.
FillEnd	The date/time to end filling if other than the full	Fill the full period
	time series period	
FillFlag	A single character that will be used to flag filled	Filled values will not be
	data.	flagged.
FillFlaqDesc	Description for the fill flag, used in reports.	Automatically generated.

Parameter	Description	Default
TableID	A table identifier for a table to receive output of the	Statistics are not written
	regression analysis (statistics are described above).	to the table. Refer to the
		log file for information.
TableTSIDColumn	The name of the column in the table that contains	Required if TableID is
	time series identifier information. This is used to	specified.
	match the table with time series being analyzed so	
	that statistics can be written to the correct row.	
TableTSIDFormat	The specifier used to format the time series identifier	The alias will be used if
	in the TableTSIDColumn. The location part of the	available, or otherwise
	TSID, or the time series alias is typically used.	the full TSID will be
		used.
SEPTSID	The time series identifier of the SEP time series,	If not specified, no SEP
	calculated for ALL values in the analysis period.	time series will be
	This parameter is not enabled but is envisioned to	generated.
	help evaluate filling and test	
	FillMixedStation().	
SEPTSAlias	The alias to be assigned to the SEP time series. This	No alias is assigned to
	parameter is not yet enabled.	the SEP time series.
FlagToWarn	A parameter is envisioned to warn the user if any	
	values in the time series are flagged with a specific	
	flag value. This will allow checks to ensure that	
	FillRegression() is not used with data that	
	have been filled in a previous step.	

The command logic is as follows, with reference to command parameters that control the process:

- 1. The dependent (TSID) and independent time series (IndependentTSID) are retrieved using the time series identifiers or aliases.
- 2. Data arrays of overlapping non-missing values are extracted from time series to be used as the samples for analysis, as specified by command parameters (analysis period specified by AnalysisStart and AnalysisEnd; transformation specified by Transformation, LEZeroLogValue, and Intercept; number of equations specified by NumberOfEquations and AnalysisMonth).
- 3. The independent and dependent statistics and relationships are calculated, computing as many of the statistics as possible (some are skipped if the sample size results in division by zero). Computing the statistics allows them to be saved in the output table for review, and is controlled by the TableID, TableTSIDColumn, and TableTSIDFormat parameters.
- 4. The statistics are analyzed to determine if the relationships are acceptable for filling by checking the minimum sample size (MinimumSampleSize), minimum correlation coefficient (MinimumR), and that the relationship meets the confidence interval (ConfidenceInterval). If monthly equations are used, then it is possible that some months can be filled but not others.
- 5. If Fill=True (the default), then the relationships that are acceptable from step 4 are used to fill the dependent time series for the period specified by the FillStart and FillEnd parameters, with FillFlag and FillFlagDesc optionally being used to indicate filled values.

This page is intentionally blank.

Command Reference: FillRepeat()

Fill missing time series data by repeating known data values

Version 09.09.00, 2010-09-23

The FillRepeat() command fills missing data in time series by repeating observations until another observation is found. This fill technique is useful, for example, where time series are likely to be stepwise or nearly constant, such as some reservoir and diversion time series.

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

\delta Edit FillRepeat() Command 🛛 🛛 🔀					
This command fills missing data in the time series by repeating non-missing values.					
The fill direction can be forward or bac	kward.				
Specify the maximum intervals as blan	or zero to fill any	gap.			
The fill start and end, if specified, will I	mit the period tha	t is fill	ed.		
Use standard date/time formats appro	priate for the date	e/time	precision of the time series.		
TS list:	AllMatchingTSID	×	Optional - indicates the time series to process (default=AllTS).		
TSID (for TSList=AllMatchingTSID):	08236500.DWR.5	Stream	nflow.Month		
EnsembleID (for TSList=EnsembleID):			×		
Fill start date/time:			Optional - fill start (default=fill all).		
Fill end date/time:			Optional - fill end (default=fill all).		
Fill direction:	Forward 🔽		Optional - fill direction (default=Forward).		
Max. intervals:			Optional - largest gap to fill (default=fill all).		
Fill flag:			Optional - string to flag filled values (default=no flag).		
Command:	FillRepeat(TSID="08236500.DWR.Streamflow.Month",Fi llDirection=Forward)				
Cancel OK					

FillRepeat() Command Editor

The command syntax is as follows:

```
FillRepeat(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
TSList	 Indicates the list of time series to be processed, one of: AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will be modified. AllTS – all time series before the command. EnsembleID – all time series in the ensemble will be modified. LastMatchingTSID – the last time series that matches the TSID (single TSID or TSID with wildcards) will be modified. SelectedTS – the time series are those selected with the SelectTimeSeries() 	AllTS
FillStart	The starting date/time for the fill.	Available period.
FillEnd	The ending date/time for the fill.	Available period.
FillDirection	Specify the direction of the fill as Forward or Backward.	Forward
MaxIntervals	The maximum number of intervals to fill in a data gap.	Fill all gaps.
Flag	String to flag filled values. Prefix with + to append the string to existing flag values.	Do not flag filled values.

A sample command file to fill a time series from the State of Colorado's HydroBase is as follows:

```
# 08236500 - ALAMOSA RIVER BELOW TERRACE RESERVOIR
08236500.DWR.Streamflow.Month~HydroBase
FillRepeat(TSList=AllMatchingTSID,TSID="08236500.DWR.Streamflow.Month",
FillDirection=Forward)
```

Command Reference: FillUsingDiversionComments()

Fill missing time series data using HydroBase diversion comments and structure CIU information

This command is only appropriate for use with diversion (e.g., DivTotal, DivClass data types) and reservoir release (e.g., RelTotal, RelClass data types) time series for the HydroBase input type.

The FillUsingDiversionComments() command fills missing data in time series by using diversion comment and structure "currently in use" (CIU) information in HydroBase. This information is used, for example, in cases where Water Commissioners have entered annual data values rather than daily or monthly records.

Diversion Comment Not Used Flag

HydroBase contains diversion comment data with a *not_used* field. If the *not_used* value matches one of the values shown in the following table for an irrigation year (November of the previous year to October of the irrigation year), the diversion (or reservoir release) data for the specified irrigation year can be interpreted as zero (see the **State of Colorado's Water Commissioner Manual** for more information):

not_used	Meaning (reason why diversion is zero)
А	Structure is not usable
В	No water is available
С	Water available, but not taken
D	Water taken in another structure

Diversion Comment not_used Flag Resulting in Additional Zero Values

Structure Currently in Use Flag

The HydroBase structure data contains a "currently in use" (CIU) field. Unlike diversion comments, this is a single value that is consistent with the current status of a structure (it is not a time series). The following CIU values are used.

Structure CIU Flag Values and Meaning

CIU	Meaning
А	Active structure with contemporary diversion records
В	Structure abandoned by the court
С	Conditional structure
D	Duplicate; ID no longer used
F	Structure used as FROM number; located in another water district
Н	Historical structure only-no longer exists or has records, but has historical data
Ι	Inactive structure which physically exists but no diversion records are kept
Ν	Non-existent structure with no contemporary or historical records
U	Active structure but diversion records are not maintained

If UseCIU=True is specified for this command, the following logic will be used to fill missing time series values:

- 1. If the HydroBase CIU value is H or I for the structure associated with the time series:
 - a. Fill using the diversion comments (see above for interpretation of comments).
 - b. The limits of the time series are recomputed based on diversion data and comments.
 - c. Missing data at the end of the period are filled with zeros, reflecting the fact that the structure is off-line. In this case, the limits are always recomputed, regardless of the value of the RecalcLimits command parameter. These values are not included in historical averages because they do not occur in the active life of the structure.
 - d. Missing data within the data period remain missing, and can be filled with other commands such as fillHistMonthAverage().
 - e. Missing data prior to the first diversion values or comments remain missing, and can be filled with other commands as appropriate, perhaps specific to each location.
- 2. If in HydroBase CIU=N:
 - a. Fill using the diversion comments (see above for interpretation of comments).
 - b. The limits of the time series are recomputed based on diversion data and comments.
 - c. Missing data at the beginning of the period are filled with zeros. In this case, the limits are always recomputed, regardless of the value of the Recalclimits command parameter.
 - d. The remaining missing data in the active data period or at the end of the period remain missing and can be filled with other commands.

The output period for filled time series is handled as follows:

- If a global output period has been specified (e.g., with the setOutputPeriod() command) then the time series will NOT be extended to include diversion comments and CIU codes beyond the output period.
- If NO output period has been specified, the time series WILL be extended to include the longer period from diversion comments. CIU information does not cause the time series to be extended.

After setting additional zero values using this command, the limits of the time series can be recomputed, if appropriate, for use with the fillHistMonthAverage() command (see the RecalcLimits=True parameter). If FillUsingCIU=true is specified, it overrides the RecalcLimits parameter as per the logic described above.

See also the ReadHydroBase() and TS Alias = ReadHydroBase() commands, which allow filling with diversion comments after reading data. Refer to the **HydroBase Input Type Appendix** for more information about diversion time series.

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

👌 Edit FillUsi	ngDiversionCo	mments() Command	×		
This command can be used to fill monthly, daily, and yearly diversions and reservoir releases for the HydroBase input type.					
The diversion comr	ments in HydroBas	e indicate years when no water was carried for an entire irrigation year.			
Consequently, mis	sing values in dive	rsion time series can be set to zero for the period November to October.			
If a yearly time ser	ries is filled, the ze	ro value in an irrigation year will be matched with the time series year.			
For the fill period,	use standard date	formats appropriate for the date precision of the time series.			
The recalculate limi	its flag, if set to Tr	rue, will cause the average to be recalculated, for use in other fill commands (see CIU note below).			
For example, use 1	Frue with a fillUsing	gDiversionComments() command immediately after reading diversions.			
If the "currently in	use" (CIU) flag is i -	used for filling, additional zeros will be added and limits are recalculated a specific way (see documentatio)n).		
Time series to fill:	*		~		
Fill start date:		Optional - start of period to fill (default=fill entire period).			
Fill end date:		Optional - end of period to fill (default=fill entire period).			
Fill flag:	Auto	Optional - string (or "Auto") to flag filled values (default=no fla	g).		
Fill using CIU:	True 🔽	Required - use currently in use (CIU) information to fill.			
Fill using CIU flag:		Optional - string (or "Auto") to flag filled values (default=no fla	g).		
Recalculate limits:	False 🔽	Optional - recalculate original data limits after fill? (default=Fals	;e).		
	FillUsingD	iversionComments(TSID="*",FillFlag="Auto",FillUsingCIU=True,Rec			
C	alcLimits=1	False)			
Command:					
		FillUsingDiversionCom	ments		

FillUsingDiversionComments() Command Editor

The command syntax is as follows:

FillUsingDiversionComments(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSID	The time series identifier or alias for the	None – must be specified.
	time series to be filled. Specify as * to	
	fill all time series.	
FillStart	The starting date/time for the fill.	Available period.
FillEnd	The ending date/time for the fill.	Available period.

Parameter	Description	Default
FillFlag	For each value that is filled using the diversion comment <i>not</i> used	No flag is assigned.
	information, tag the filled value as	
	follows:	
	• If FillFlag is specified as a single	
	specified character	
	 If FillFlag=Auto is specified, 	
	the diversion comment <i>not_used</i>	
	value (A, B, C, or D) from	
	HydroBase is used for the flag.	
	graphs etc. The flag will be appended to	
	existing flags if necessary.	
FillUsingCIU	Indicates whether the "currently in use"	False (CIU information is not
	(CIU) information is used to fill missing	used to fill missing data).
	data. This will result in additional zeros	
	depending on CIU value See the	
	description of the logic above. Note that	
	this will cause the time series data limits	
	to be automatically recomputed,	
	regardless of the value of the	
FillUsingCIUFlag	For each missing data value that is filled	No flag is assigned
	using the CIU information, tag the filled	i to mug is ussigned.
	value as follows:	
	• If FillUsingCIUFlag=Auto is	
	specified, the CIU value (H, I, or N)	
	Flse if FillusingCluElag is	
	specified, tag filled values with the	
	specified character.	
	The flag can then be used later to label	
	graphs, etc. The flag will be appended to	
RecalcLimits	Indicate whether the original data limits	False (additional zeros are not
	for the time series should be recalculated	considered in the original data
	after the zero values are set. Zero values	averages).
	are included in the monthly and annual	
	averages.	
	See the discussion above related to CIU –	
	time series that are impacted by CIU	
	always have their limits recalculated.	
A sample commands file to fill diversion time series from the State of Colorado's HydroBase is as follows:

```
# 0100506 - PUTNAM DITCH
0100506.DWR.DivTotal.Month~HydroBase
# 0100503 - RIVERSIDE CANAL
0100503.DWR.DivTotal.Month~HydroBase
# 0100501 - EMPIRE DITCH
0100501.DWR.DivTotal.Month~HydroBase
FillUsingDiversionComments(TSID="*",RecalcLimits=True)
```

The following example fills one time series and labels the values with the flag.

```
# Set the date to cause comments NOT to automatically extend the period.
# setOutputPeriod(1950-01,1989-06)
# 0100713 - PIONEER DITCH
0100713.DWR.DivTotal.Month~HydroBase
FillUsingDiversionComments(TSID="*",FillFlag="Auto",RecalcLimits=False)
```

The corresponding graph created with data flags as labels is shown below (note the D symbols on the right). It may be necessary to change the graph properties to display the data labels above the point in order to see labels at the bottom of the graph.



Example Graph Showing Fill Flag (D labels indicate additional zero values)

This page is intentionally blank.

Command Reference: FormatDateTimeProperty()

Format a date/time property as a new string property

The FormatDateTimeProperty() command creates a new global string property by formatting an existing date/time property. These properties are accessible to commands using \${Property} notation. A formatted date/time string is useful when specifying filenames more dynamically. Date/time properties will by default be formatted using the ISO 8061 format (e.g., YYYY-MM-DD hh:mm:ss). Support for properties varies by command and command documentation should be consulted. This command should not be confused with the SetTimeSeriesProperty() command, which sets a property on specific time series.

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

👌 Edit FormatDateT	Edit FormatDateTimeProperty() Command			
Format a date/time propert	y to create a new string property.			
The property can be refere	nced in parameters of some commands using \${Property} notation.			
For example, use the string	g property to create file names that include date/time information.			
Property name:	DateTimePropString	Required - new property (do not use spaces \$, { or } in name).		
Date/time property name:	DateTimeProp 💌	Required - existing date/time property to format.		
Format:	✓ Select Specifier ▼ => %Y-%m-%dT%H:%M:%S	Required - format string for formatter.		
	FormatDateTimeProperty(PropertyName="DateTimeP	ropString",DateTimePropertyName="Date		
Command:	Command: TimeProp", Format="%Y-%m-%dT%H:%N:%S")			
Cancel				
		FormatDateTimeProperty		

FormatDateTimeProperty() Command Editor

The command syntax is as follows:

```
FormatDateTimeProperty(Parameter=Value,...)
```

Parameter	Description	Default
PropertyName	The name of the string property to be created.	None – must be
		specified.
DateTimePropertyName	The name of the existing date/time property	None – must be
	to be formatted.	specified.
FormatterType	The date/time formatter type, which defines	С
	the format specifiers, one of:	
	• C – the C programming language	
	strftime() function, which has been	
	widely copied (described below).	
	• MS – Microsoft convention (currently not	
	supported but may be added in the	
	future).	

Parameter	Description	Default
Format	The format string for the formatter, which	None – must be
	defines how date/time data parts are formatted	specified.
	into the new string property. The string is	
	interpreted by the formatter as follows:	
	• Formatter=Strftime – The string	
	can contain literal characters and format	
	specifiers that start with the % character.	

The following table lists the supported formatting strings for FormatterType=C:

Format Specifier	Description	
å	Weekday abbreviation (e.g., Sun)	
۶A	Weekday (e.g., Sunday).	
%b	Month abbreviation (e.g., Jan).	
%В	Month (e.g., January).	
%d	Day (01-31).	
%H	Hour (00-23).	
%I	Hour (01-12).	
%j	Day of year (001-366).	
%m	Month (01-12).	
۶M	Minute (00-59).	
%p	AM, PM (noon=PM, midnight=AM).	
%S	Second (00-59).	
۶y	Year (00-99).	
۶Y ۲	Year (0000-9999).	
%Z	Time zone (e.g., MST).	

A sample command file is as follows:

Command Reference: FormatTableString()

Format a string column in a table, using other columns as input

Version 10.21.00, 2013-06-26

The FormatTableString() used zero or more table columns as input and formats an output table column. For example, it may be necessary to concatenate information from several columns to create an identifier. It can also be used to assign a literal string to a column. See also the ManipulateTableString() command. Formatting occurs as follows:

- The data types for input columns control the type of formatting that can be done. For example, columns containing floating-point numbers must use the format specifiers for floating-point numbers.
- Format specifiers are consistent with the C programming language.
- Missing values in input will result in blanks in output.

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

Edit Forma	tTableString() Command	22
Format one or n	nore input columns to create an output column value,	
Formatting uses	C-style format specifiers, including literal strings and format specifiers	4
%% - literal pe	ercent character	
%c - single cha	aracter	
%s, %-20.20s	- include entire string, fit to 20 characters left-justified	And the second se
%d, %4d, %0	04d, %-04d - include integer, pad with spaces for 4 digits, pad with zer	os for 4 digits, left-justify
%f, %8.2f, %	#8.2f, %-8.0f - include float, use width of 8 and 2 decimals, force de	cimal point, left-justify
Future enhance	ments may provide more cell range addressing - currently full columns	are processed.
Table ID:	Table 1	 Required - table to process.
Input columns	String 1, Integer, Double	Required - names of columns to process, separated by comma
Format:	← Select Specifier ← => %20.20s-%4d-%8.2f	Required - format string,
Dutput column:	Formatted	✓ Required - output column name.
Command:	FormatTableString(TableID="Table1",InputColumn="Formatted")	olumns="String1,Integer,Double",Format="%2
	Cancel	ĸ

FormatTableString() Command Editor

FormatTableString(Parameter=Value,...)

Parameter	Description	Default
TableID	The identifier for the table to process.	None – must be
		specified.
InputColumns	The names of one or more input columns. Values from	Required if format
	the columns will be formatted according to the Format	specifiers are given.
	parameter. Input columns can be omitted if the format	
	string is a literal value.	
Format	The format specifier string used to format the data	None – must be
	values. See the editor dialog for examples and refer to	specified.
	"sprintf" documentation on the internet for further	
	explanation. Specify as many format specifiers as input	
	columns. All other characters will be transferred to the	
	output string.	
OutputColumn	The name of the column to receive the output.	None – must be
		specified.

Command Reference: Free()

Free (remove) time series from memory

Version 09.10.01, 2010-11-18

The Free() command frees (removes) the selected time series from memory. The time series will therefore not be available for use after that line in the command file. This command is useful for discarding temporary time series needed for data manipulation (e.g., so that they are not written in output and are not available for interactive plots). Freed time series are also removed from any ensembles that reference the time series.

Rather than freeing time series, it may be more appropriate to use the SelectTimeSeries() command, which can be used in conjunction with some commands to select time series and then operate on the selected time series. This approach allows selective use of time series and minimized the need for Free() commands. Many commands also use a TSList parameter to indicate which time series should be operated on by a command.

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

JEdit Free() Command			X
This command frees (removes) time series, whi	ich is useful to remo	ve unneeded or temporary time series.	
The list of time series to be removed can be ind	dicated in several w	ays.	
Time series identifiers follow the pattern:			
Location.Source.DataType.Interval.Scenario			
Examples of wildcard use when TSList=AllMatch	hingTSID are shown	below:	
* - matches all time series			
ABC* - matches locations starting with ABC	which are with a	and the second second second	
ABC*,*, Type, Month - matches locations startin	ng with ABC, with a	ata type Type and Interval Month,	
Time series that are in an ensemble will be remo	oved from the ence	nde	
TS lict	AllMatchingTSID	Ontional - indicates the time series to process (default=AllTS)	
	1 milliocening 1 Sto	opara di maratas dia anto serios coprocess (daradare 11115).	
TSLD (for TSList=matching TSLD):	40"		
EncentrieID (for T5Lst=EnsembleID))	_		
Time series position(s) ((o) T5List=(SPosition))		Optional - use with TSList=TSPosition (e.g., 1,2,7-8).	
Free ensemble if empty?	~	Optional - for ensembles (default=True).	
	Free (TSList	=AllMatchingTSID,TSID="40*")	
Command:			
	L	And and a second second	
		Cancel OK	
			Fre

Free() Command Editor

Free(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be	AllTS
	processed, one of:	
	• AllMatchingTSID – all time	
	series that match the TSID (single	
	TSID or TSID with wildcards) will	
	be modified.	
	• AllTS – all time series before the command	
	• EnsembleTD - all time series in	
	the ensemble will be modified (see	
	the Ensemble ID parameter).	
	 LastMatchingTSID – the last 	
	time series that matches the TSID	
	(single TSID or TSID with	
	wildcards) will be modified.	
	• TSPosition – time series	
	specified by position in the results	
	list (see TSPosition parameter	
	below).	
TSID	The time series identifier or alias for	Required if TSList=*TSID
	the time series to be modified, using the	
	* wildcard character to match multiple	
	time series.	
EnsembleID	The ensemble to be modified, if	Required if
	processing an ensemble.	TSList=EnsembleID
TSPosition	A list of time series positions (1+) in	Required if
	output, separated by commas. Ranges	TSList=TSPosition
	can be specified as Start-End.	
FreeEnsembleIfEmpty	Indicate whether to free the ensemble	True
	from which time series were removed,	
	If the ensemble is empty (has no time	
	series remaining after the Free()	
	command).	

A sample command file is as follows:

Free(TSList=AllMatchingTSID,TSID="40*")

Command Reference: FreeTable()

Free a table Version 10.16.01, 2013-02-14

The FreeTable() command frees a table. The table will not be available for subsequent commands, although a new table with the same name can be created and used with subsequent commands. This command is useful, for example, when looping through blocks of commands where logic is repeated and the table contents are recreated.

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

Edit Free	Edit FreeTable() Command				
This comma	This command frees the specified table. The table will not be available to following commands unless it is recreated with another command.				
Table ID:	Table2	Required - original table.			
	FreeTable(TableID="Table2")				
Command:					
Cancel OK					
		FreeTable			

FreeTable() Command Editor

The command syntax is as follows:

```
FreeTable(Parameter=Value,...)
```

Parameter	Description	Default
TableID	The identifier for the original table.	None – must be
		specified.

This page is intentionally blank.

Command Reference: FTPGet() Retrieve file(s) from a remote system using file transfer protocol (FTP)

Version 10.01.00, 2011-11-15

The FTPGet() command retrieves one or more files from a remote system using file transfer protocol (FTP). The retrieval is not recursive to child folders.

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

\delta Edit FTPGet() command 🔀			
This command uses file transfer protocol (FTP) to retrieve files from a remote site and save on the local file system.			
The destination fold	der can be specified using \${Proper	ty} notation to utilize global properties.	
It is recommended	that the local folder name be relativ	ve to the working directory, which is:	
C:\Develop\TSTc	ol_SourceBuild\TSTool\test\regres: /	sion\UserManualExamples\TestCases\CommandReference\FTPGet	
Remote site:	ftp.somebody.com	Required - FTP site (can use \${Property}).	
Login:	anonymous	Optional - case-sensitive (default=anonymous).	
Password:	mypassword	Optional - case-sensitive (default=anonymous).	
Remote folder:	/outgoing	Optional (default=/ root on FTP server, can use \${Property}).	
File pattern:	data.txt	Optional (default=*, can use \${Property}).	
Destination folder:	Results	Browse	
Transfer mode:	ASCII 🔽	Optional (default=Binary).	
Retry count:		Optional (default=3).	
Retry wait:		Optional - seconds (default=3).	
Command:	Command: FTPGet(RemoteSite="ftp.somebody.com",Login="anonymous",Password=" mypassword",RemoteFolder="/outgoing",FilePattern="data.txt",Desti nationFolder="Results",TransferMode=ASCII)		
	Add Working Directory to Destination Folder Cancel OK		

FTPGet() Command Editor

```
FTPGet(Parameter=Value,...)
```

Parameter	Description	Default
RemoteSite	The address of the remote site, for	None – must be specified.
	example: <u>ftp.acme.com</u>	
	Global properties can be used with the	
	\${Property} syntax.	
Login	The FTP login to use.	anonymous
Password	The FTP password to use.	anonymous
RemoteFolder	The folder on the remote site, for	Root folder (/).
	example: /outgoing/data	
	Global properties can be used with the	
	\${Property} syntax.	
FilePattern	The pattern to use to determine which	Retrieve all files in the
	files should be transferred. Simple	RemoteFolder.
	patterns are used, where * is a wildcard.	
	Global properties can be used with the	
	\${Property} syntax.	
DestinationFolder	The folder to receive the files, can be	None – must be specified.
	relative to the working directory. Global	
	properties can be used with the	
	\${Property} syntax.	
TransferMode	The transfer mode:	Binary
	• ASCII – for text files	
	• Binary – for binary files	
RetryCount	The number of times to retry the login if	3
	it fails (e.g., due to busy site).	
RetryWait	The amount of time to wait between	3
	retries, seconds.	

Command Reference: InsertTimeSeriesIntoEnsemble ()

Insert 1+ time series into an existing ensemble

Version 10.13.00, 2012-10-25

The InsertTimeSeriesIntoEnsemble() command inserts 1+ time series into an ensemble. The time series must have the same interval and data units as the time series in the ensemble. For example, use the command to insert scenario time series into an ensemble.

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

😹 Edit InsertTimeSeriesIntoEnse	emble() Command
Insert 1+ time series into an ensemble	
The time series must have the same da	ta interval and units as existing time series in the ensemble.
The original time series will remain avail	able and can be accessed directly in the ensemble.
TS list:	Optional - indicates the time series to process (default=AllTS).
TSID (for TSList=AllMatchingTSID);	v
EnsembleID (for TSList=EnsembleID);	v
Receiving ensemble ID:	TestEnsemble Required - identifier for ensemble to receive time series.
	InsertTimeSeriesIntoEnsemble(EnsembleID2="TestEnsem
Commente	ble")
Command:	
	Cancel OK

InsertTimeSeriesIntoEnsemble () Command Editor

```
InsertTimeSeriesIntoEnsemble (Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be added to the ensemble, one of:	Allts
	• AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards).	
	• AllTS – all time series before the command.	
	• EnsembleID – all time series in the ensemble.	
	• FirstMatchingTSID – the first time	
	series that matches the TSID (single TSID or TSID with wildcards).	
	• LastMatchingTSID – the last time series that matches the TSID (single TSID or TSID with wildcards).	
	• SelectedTS – the time series are those selected with the SelectTimeSeries() command.	
TSID	The time series identifier or alias for the time	Required when
	series to be processed, using the * wildcard	TSList=*TSID
-	character to match multiple time series.	
EnsembleID	The ensemble from which to retrieve time series,	Required when
	if inserting time series from an ensemble.	TSList=EnsembleID.
EnsembleID2	The identifier for the ensemble that is receiving the time series.	None – must be specified.

A sample command file to create an ensemble from user-defined time series is as follows:

```
# Test inserting time series into an ensemble from year interval time series
NewPatternTimeSeries(Alias="ts1",NewTSID="ts1..Flow.Year",
    SetStart="1960",SetEnd="2000",Units="ACFT",
    PatternValues="1,2,5,8,,20")
NewPatternTimeSeries(Alias="ts2",NewTSID="ts2..Flow.Year",
    SetStart="1950",SetEnd="2005",Units="ACFT",
    PatternValues="2,4,10,16,,40")
NewEnsemble(TSList=AllTS,
    NewEnsembleID="TestEnsemble",NewEnsembleName="Test Ensemble")
InsertTimeSeriesIntoEnsemble(Ensemble2="TestEnsemble")
```

Command Reference: LagK()

Lag and attenuate (route) a time series

The LagK() command can be used to lag and attenuate an input time series, resulting in a new time series. The time series identifier for the new time series is the same as the original time series with "routed" appended to the scenario. The command is commonly used to route an instantaneous flow time series through a stretch of river (reach). Lag and K routing is a common routing method that combines the concepts of:

- 1. Lagging the inflow to simulate travel time in a reach and,
- 2. Attenuating the wave to simulate the storage-outflow relationship for the reach (see Figure 1).



Figure 1: Lag and K Routing

At its fundamental level, the method solves the continuity equation using an approach similar to Muskingum routing (assuming that the Muskingum parameter representing wave storage is negligible). The governing equation for this routing method is given as:

$$Q_{in} - Q_{out} = \frac{\Delta S}{\Delta t}$$

where:

 Q_{in} = instantaneous inflow [rate] lagged appropriately, Q_{out} = instantaneous outflow [rate] lagged appropriately, ΔS = change in storage in the reach [volume], Δt = time difference. The relationship assumes an outflow-storage relationship of the form:

$$S = k \cdot Q_{out}$$

where:

k = attenuation for the outflow [time].

To ensure accurate results, k should be larger or equal to $\Delta t/2$. For discrete time steps these relationships translate into:

$$O_2 = \frac{I_1 + I_2 + \frac{2S_1}{\Delta t} - O_1}{\frac{2k}{\Delta t} + 1}, \qquad k \ge \frac{\Delta t}{2}$$

where: I_1 and I_2 are the lagged inflows into the reach at the previous and current time step, respectively,

 O_1 and O_2 are the outflows out of the reach at the previous and current time step, respectively, S_1 is the storage within the reach at the previous time step, defined as $S_1 = k \cdot O_1$, and Δt is the time difference between the two time steps.

In the case that either I_1 , I_2 or O_1 are missing, these values will be set in the following order:

- 1. Use data from an observed time series (see ObsTSID parameter below).
- 2. Use the nearest value in the input time series (see FillNearest parameter below).
- 3. Use the nearest value in the observed time series (see FillNearest parameter and the ObsTSID parameter below).
- 4. Use a defined default flow value (see DefaultFlow parameter below).

By default, the identifier of the resulting time series is the same as the original input time series, with the data subtype set to "routed" (e.g., Streamflow becomes Streamflow-routed)

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

Lag and attenuate a time series	, creating a new tir	ne series,	
The time series to be routed car	nnot contain missing	g values.	
The observed time series is use	d for filling, then Fil	Nearest, and finally DefaultFlow.	
See the documentation for a co	mplete description	of the algorithm.	
Time series to lag (TSID):	ts1		8
Observed time series for filling:	1		(Y
Alias to assign:	ts1Routed	Insert: Select Specifier	Required - use %L for location, etc.
Fill nearest?:	True 😽		Optional - fill missing with nearest data from TSID? (default=False).
Default flow:			Optional - use if no other filling works (default=0).
Lag:	3		Required - lag in time series base interval time units.
к:	2		Required - attenuation in time series base interval time units.
Inflow states:			Optional - separate values by commas (default=0 for all).
Outflow states:			Optional - separate values by commas (default=0 for all).
Command:	LagK(Alias=	"ts1Routed",TSID="ts1	",FillNearest=True,Lag=3,K=2)
		Cancel	ж

LagK() Command Editor

Values for Lag and K can usually be established by comparing routed flows to downstream observations. Alternatively, the Lag can be estimated using the reach length and wave speed in the reach. Without any other information, K can be set to Lag/2.

The command syntax is as follows:

LagK(Parameter=Value,...)

The following older command syntax is updated to the above syntax when a command file is read:

TS Alias = LagK(Parameter=Value,...)

Parameter	Description	Default
TSID	Identifier or alias for the time series to be routed. It is assumed that	None –
	this series describes an instantaneous flow. Due to the lagging, the	must be
	first data values required for the computation of O_2 are not available	specified.
	within this time series and are therefore set to values set in the	_
	InflowStates parameter. See also the ObsTSID time series, and	
	the FillNearest and DefaultFlow parameters.	
ObsTSID	Identifier or alias for an observed time series. If specified, the	None

Parameter	Description	Default
	missing values in the TSID time series will be taken from the	
	observed time series if non-missing. ObsTSID can be used in	
	conjunction with FillNearest to substitute a missing value in the	
	TSID time series with the nearest non-missing value in ObsTSID.	
Alias	The alias to assign to the time series, as a literal string or using the	None –
	special formatting characters listed by the command editor. The alias	must be
	is a short identifier used by other commands to locate time series for	specified.
	processing, as an alternative to the time series identifier (TSID).	
FillNearest	If set to True, then when a missing data value is found anywhere in	False
	the lagged period, a replacement value will be determined by	
	searching forward and back in time in the input time series to find the	
	nearest non-missing value. The maximum search window depends	
	on the interval of the TSID time series:	
	• <= Seconds: 1000 intervals	
	• Minute, Hour: 1 day	
	• Day: 1 Week	
	• > Day: 1 interval only	
	The assumption is that a flow value close in time will be	
	representative of the missing value and will not result in significant	
	errors.	
	This option has lower precedence than specifying the ObsTSID data.	
	It can also find non-missing data in the ObsTSID if ObsTSID is	
	defined (lower precedence). Both options have a higher precedence	
	than DefaultFlow.	
DefaultFlow	A flow value in the units of the input time series that is substituted for	0
	missing values in the input time series. This has the lowest	
	precedence of all missing data substitutions. It will be applied at any	
	time in the lagged period.	
Lag	Lag time for the modeled reach in the units of the TSID time series	Required
	base interval. For example, if the input time series is 10 minutes, the	
	units of Lag are assumed to be minutes. The Lag value is not	
	required to be evenly divisible by the time step interval; values in the	
	time series between time steps will be linearly interpolated.	
K	Attenuation factor to be applied to the wave. The units of K are time,	Required
	and like the Lag value, it is assumed to have the same units as the	
	input time series.	
InflowStates	Comma-delimited list of default inflow values prior to the start of the	0 for each
	time series. The order of the values is earliest to latest. The array	value
	must specify (Lag/multiplier) + 1 values; i.e., a 10 minute interval	
	with a LAG of 30 must be provided with $30/10 + 1 = 4$ inflow	
	carryover values. Note: Specifying values that are not consistent	
	with the Lag and K parameters will result in oscillation!	
OUTIIOWStates	Comma-delimited list of default outflow values prior to the start of	0 tor each
	the time series. See InflowStates for details.	value

A sample command file is as follows (commands to read time series are omitted):

LagK(Alias="LKPN6routed", TSID=LKPN6.USGS.QIN.1HOUR,Lag=3,K=2,FillNearest=true)

Command Reference: LookupTimeSeriesFromTable()

Crate new time series by using an input time series and a lookup table

Version 10.05.00, 2012-02-12

The LookupTimeSeriesFromTable() command uses an input time series and lookup table to create the output time series. Examples of using this command include:

- Converting reservoir elevation to storage, surface area, seepage, or other values
- Converting river stage to discharge
- Converting a time series to category values

In many cases the lookup table will apply throughout the analysis period. However, it is possible that the table will change over time (e.g., as a stream channel changes or a reservoir fills with silt). In these cases, the command allows for an effective date to be specified – the table then is applicable on and after the specified date/time, until another effective date is encountered. The values in the table should be sorted in ascending order prior to lookup. This command currently does not handle rating table shifts; however, this capability may be added in the future. The following dialog is used to edit the command and illustrates the syntax of the command:

🛇 Edit LookupTimeSeriesFromTable() Command 🛛 🛛 🔀		
Create a new time series by using	an input time series and lookup table.	
Specify new time series identifier (TSID) information for the copy to avoid errors with the copy being mistaken for the original.		
Currently the lookup table m	ust only contain data for only one input time s	eries and the effective date is not checked.
Input time series:	Original	Y
New time series ID:	OutputVolumn.Day	Required - specify to avoid confusion with TSID from original time series.
		Edit Clear
Alias to assign:	%L-%T Insert: Select Specifier -	- 🔽 Optional - use %L for location, etc.
Lookup table ID:	LookupTable	Required - lookup table.
Table TSID column:		Optional - column name for input time series TSID.
Format of TSID:	Insert: Select Specifier	 Optional - use %L for location, etc. (default=alias or TSID).
Column for input value:	Value1	Required - column for input time series values.
Column for output value:	Value2	Required - column for output time series values.
Output data units:	FT3	Optional - for example: ACFT, CFS, IN (default=no units).
Column for effective date:		Optional - column for lookup data effective date.
Lookup method:	Interpolate 💌	Optional - how to lookup values (blank=Interpolate).
Out of range lookup method:	Extrapolate 💌	Optional - how to lookup values outside table values (default=SetMissing).
Out of range notification:	~	Optional - how to notify about out of range values (default=Ignore).
Transformation:	~	Optional - how to transform data if interpolating (blank=None).
Value to use when log and <= 0:		Optional - value to substitute when original is ≤ 0 and log transform (default=0.0010).
Analysis start:		Optional - analysis start date/time (default=full time series period).
Analysis end:		Optional - analysis end date/time (default=full time series period).
Command:	LookupTimeSeriesFromTable(TSID="Original",NewTSID="OutputVolumn.Day",Alias="&L-&T",Ta bleID="LookupTable",TableValue1Column="Value1",TableValue2Column="Value2",Units="FT3",L ookupMethod=Interpolate,OutOfRangeLookupMethod=Extrapolate)	
Cancel OK		

LookupTimeSeriesFromTable() Command Editor

LookupTimeSeriesFromTable(Parameter=Value,...)

Parameter	Description	Default
TSID	The time series identifier or alias for the time series used as	None – must be
	input.	specified.
NewTSID	The time series identifier for the time series being created.	None – must be
	Use the <i>Edit</i> button to edit the time series identifier parts.	specified.
Alias	The alias to assign to the time series, as a literal string or	No alias.
	using the special formatting characters listed by the command	
	editor. The alias is a short identifier used by other commands	
	to locate time series for processing, as an alternative to the	
	time series identifier (TSID).	
TableID	The lookup table identifier.	None – must be
malal a		specified.
	I able column name that is used to match the time series	If not specified,
ISIDCOLUMII	supported but will be enabled in the future	that the optime
	supported but will be enabled in the future.	lookun table
		applies
Table	The specification to format the time series identifier to match	Time series
TSIDFormat	the TableTSIDColumn column This parameter	alias if
	currently is not supported but will be enabled in the	available. or
	future.	otherwise the
		time series
		identifier.
Table	Table column name for data values that correspond to the	None – must be
ValuelColumn	input time series (TSID).	specified.
Table	Table column name for data values that correspond to the	None – must be
Value2Column	output (new) time series identifier (NewTSID).	specified.
Units	The data units to assign to the new time series.	No data units
		will be
		assigned.
EIIective	Table column name for the effective date. This parameter	The lookup data
Datecolumn	future	apply to the
LookupMethod	Indicate how to select the value to use for output:	Interpolate
поскариссиоа	• Intercelate interpolate between points if input	incerporace
	values do not exactly align with table values: if	
	Transformation-Log then interpolation will use the	
	transformed values	
	• PreviousValue – pick the previous (lower) value in	
	the table (exact matches use the lookup table value)	
	• NextValue – pick the next (higher) value in the table	
	(exact matches use the lookup table value)	
OutOfRange	Indicate the value to use when estimating values that are	SetMissing
LookupMethod	outside the range of the rating table:	

Parameter	Description	Default
	• Extrapolate – use the two known values at the end of	
	the table to extrapolate; if Transformation=Log,	
	then extrapolation will use the transformed values	
	 SetMissing – set output to missing 	
	• UseEndValue – use the data value on the end	
OutOfRange	Indicate the notification to generate when a value is outside	Ignore
Notification	the range of the lookup table:	
	• Ignore – do not generate warning or failure message	
	• Warn – generate a warning message	
	• Fail – generate a failure message	
Transformation	Indicates how to transform the data before interpolation, used	None (no
	when LookupMethod=Interpolate and	transformation).
	OutOfRangeMethod=Extrapolate). Specify as None	
	to compare raw values or Log (for log_{10}) to transform values	
	before interpolation and extrapolation. If the Log option is	
	used, zero and negative values are replaced with the value	
	specified by the LEZeroLogValue parameter value for	
	analysis (missing data values are ignored in the analysis).	
LEZero	Value to use for data values less than or equal to zero when	.0010
LogValue	using a log transformation.	
AnalysisStart	The date/time to start the analysis.	Analyze the full
		period.
AnalysisEnd	The date/time to end the analysis.	Analyze the full
		period.

This page is intentionally blank.

Command Reference: ManipulateTableString()

Manipulate string a string column in a table

Version 10.21.00, 2013-06-14

The ManipulateTableString() command manipulates a string column in a table. For example, it may be necessary to manipulate strings in a table in order to match time series identifier parts, so that lookups can occur.

The input is specified by:

- a table column name (InputColumn1)
- either a second input column name (InputColumn2) or a constant string value (InputValue2)
- optionally, some operators require an additional input value (InputValue3)

The result is placed in the output column (OutputColumn). Missing/blank input will be considered as empty strings when formatting the output. The output column can be the same as an existing table column.

The following dialog is used to edit the command and illustrates the syntax of the command (in this case illustrating how the contents of column String2 are prepended to the contents of a column named String1 and placed in the output column String3).

💧 Edit Mani	pulateTableString() Comma	nd 🔀
Perform simple manipulation on columns of string data in a table, using one of the following approaches:		
- process input	t from two columns to populate the	output column
- process input	t from a column and a constant to p	opulate the output column
Future enhance	ments may provide more cell range	addressing - currently full columns are processed.
Table ID:	Table1	Required - table to process.
Input column 1:	String1	Required - first input column name.
String operator:	Prepend 🗸	Required - string manipulation to perform on input.
Input column 2:	String2	Required if no input value 2 - second input column name.
Input value 2:		Required if no input column 2 - constant string.
Output column:	String3	Required - output column name.
Command:	ManipulateTableString(TableID="Table1",InputColumn1="S tring1",Operator="Prepend",InputColumn2="String2",Outp utColumn="String3")	
Cancel OK		

ManipulateTableString() Command Editor

ManipulateTableString(Parameter=Value,...)

Parameter	Description	Default
TableID	The identifier for the table to process.	None – must be
		specified.
InputColumn1	The name of a column containing strings, as the first	None – must be
	input.	specified.
Operator	The operation to perform on the input strings:	None – must be
	• Append – append the second input to the first input	specified.
	• Prepend – prepend the second input before the	
	first input	
	 Replace – replace in the second input in the first input with the third input 	
	 ToDate – convert the first input to a DateTime 	
	object with date precision	
	• ToDateTime – convert the first input to a	
	DateTime object	
	• ToDouble – convert the first input to a double	
	precision object	
	• ToInteger – convert the first input to an integer	
	object	
InputColumn2	The name of a column containing strings, as the second	Required if a second
	input.	input value is needed
		and InputValue2 is
		not specified.
InputValue2	A string constant, as the second input.	Required if a second
		input value is needed
		and InputColumn2
		is not specified.
InputValue3	A string constant, as the third input.	Required if a third
		input value is needed.
OutputColumn	The name of a column to receive the output.	None – must be
		specified.

Command Reference: Multiply()

Multiply the data values in a time series by data values in another time series

Version 08.16.04, 2008-09-24

Multiply

The Multiply() command multiplies one time series by another. Missing data in either time series causes the result to be missing. See also the Scale() command, which multiplies time series by a numerical value.

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

🛃 Edit Multiply() Command	
Multiply one time series by	another (the first time series is modified)
Missing data in either time :	series sets the result to missing.
Time series to modify:	2184.NOAA.TempMean.Month
Time series to multiply by:	5706.NOAA.TempMean.Month
Command:	Multiply(TSID="2184.NOAA.TempMean.Month",Multiplie rTSID="5706.NOAA.TempMean.Month")
	Cancel OK

Multiply() Command Editor

Multiply(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSID	The time series identifier or alias for the time series to be	None – must be
	modified.	specified.
MultiplierTSID	The time series identifier or alias for the time series that is the	None – must be
	multiplier.	specified.

A sample command file is as follows (this example does not necessarily make sense but illustrates how the Multiply() command can be used for numerical calculations in an analysis):

```
# 2184 - DEL NORTE 2 E
2184.NOAA.TempMean.Month~HydroBase
# 5706 - MONTE VISTA 2 W
5706.NOAA.TempMean.Month~HydroBase
Multiply(TSID="2184.NOAA.TempMean.Month",
   MultiplierTSID="5706.NOAA.TempMean.Month")
```

Command Reference: NewDayTSFromMonthAndDayTS()

Create a new daily time series from monthly total and daily pattern

The NewDayTSFromMonthAndDayTS() command creates a new daily time series by distributing a monthly time series "volume" according to the pattern of the independent daily time series. This command currently only handles processing monthly ACFT and daily CFS time series. This command is useful where a monthly flow time series is known at a location, and a daily pattern is known at a related gage. The new time series is assigned the given identifier and alias. The following calculations are performed:

$$Day TS2_{i} = MonthTS2 \frac{ACFT}{NDAYS} * \left(\frac{1DAY}{86400s}\right) \left(\frac{43560FT^{2}}{1AC}\right) * \left(\frac{Day TS1_{i}}{\sum_{i=1}^{i=Nday \sin Month} \sum_{i=1}^{i=Nday TS1_{i}} Day TS1_{i}}\right)$$

where, for days in a month:

 $DayTS2_i$ = the daily value being estimated in daily time series 2 MonthTS2 = the monthly value being used for volumes for time series 2, shown in units of ACFT/NDAYS (equivalent to ACFT/Month) NDAYS = the number of days in the month $DayTS1_i$ = the daily value for indicator daily time series 1 $\Sigma DayTS1_i$ = the sum of the daily values for indicator time series for the a month

In summary, the monthly volume in ACFT/NDAYS is first converted to an average monthly CFS rate by multiplying by 43560/86400 (or 1/1.9835), and finally the average CFS value is prorated by the ratio of the indicator daily time series daily value divided by the total daily flows for the month, to give a daily CFS value for each day of the month. In this case, the last term is simply a ratio (converting daily average CFS to daily ACFT and calculating the ratio would result in the same value).

Days with missing data are excluded from the summation and the estimated values. The output period is the global output period from SetOutputPeriod(), or if not set the period from the daily time series is used.

For example, consider May a may total for MonthTS2 = 1001.7 ACFT and daily values (CFS) as follows:

Day 1 = 14
14
13
13
14
14
15
15
15
16
17
17
16
18
18
17
18
18
18
18
17
17
17
17
16
16
17
18
18
17
Day 31 = 17

The total is 505 CFS. The estimated value for day 1 of the second daily time series would then be:

1001.7 * (1/1.9835) * (14/505) = 14 CFS

In this case, the indicator time series was the same as the time series being estimated and therefore the estimated value should be the same as the indicator.

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

Create a daily time series by distributing a monthly total using a daily pattern. Currently this command is only implemented to convert from acre-feet to CFS. The period is taken from the global output period if set, or the daily time series.		
Currently this command is only implemented to convert from acre-feet to CFS. The period is taken from the global output period if set, or the daily time series.		
The period is taken from the global output period if set, or the daily time series.		
Manthly time series for tataly 09226000 DWD Streensflow Manth		
Monality time series for total: 08236000.DWR.Streamnow.Monan	×	
Daily time series for distribution: 08236500.DWR.Streamflow.Day	*	
New time series ID: 08236000.DWR.Streamflow.Day Specify to avoid confusion with TSID fr	rom original TS.	
Edit Clear		
Alias to assign: DayTS Insert: Select Specifier 💌 Required - use %L for location, etc.		
Command: 36500.DWR.Streamflow.Day")	NewDayTSFromMonthAndDayTS(Alias="DayTS",NewTSID="08236000.DWR.Stre amflow.Day",MonthTSID="08236000.DWR.Streamflow.Month",DayTSID="082 36500.DWR.Streamflow.Day")	
Cancel OK		

NewDayTSFromMonthAndDayTS() Command Editor

The command syntax is as follows:

```
NewDayTSFromMonthAndDayTS(Parameter=Value,...)
```

The following older command syntax is updated to the above syntax when a command file is read:

TS Alias = NewDayTSFromMonthAndDayTS(Parameter=Value,...)

Parameter	Description	Default
MonthTSID	The time series identifier or alias for a	None – must be specified.
	monthly time series supplying monthly	
	ACFT values.	
DayTSID	The time series identifier or alias for a	None – must be specified.
	daily time series supplying daily flow	
	values (only the pattern is used).	
NewTSID	The time series identifier of the new time	None – must be specified.
	series. The interval must be Day.	
Alias	The alias to assign to the time series, as a	None – must be specified.
	literal string or using the special	
	formatting characters listed by the	
	command editor. The alias is a short	
	identifier used by other commands to	
	locate time series for processing, as an	
	alternative to the time series identifier	
	(TSID).	

A sample command file to process data from the State of Colorado's HydroBase is as follows:



A graph of data resulting from this command may look similar to the following. Note that the each time series has a similar pattern, but at different levels.



Result of NewDayTSFromMonthAndDayTS() Command

Command Reference: NewEndOfMonthTSFromDayTS()

Use a daily time series to create an end of month time series

The NewEndOfMonthTSFromDayTS() command is typically used to convert a daily reservoir storage time series to an end of month reservoir storage time series. The command can also be applied to other data types (e.g., measured well levels).

Changing from a daily to an end of month monthly time series is accomplished by starting on the month ending day and searching in both directions (backward then forward by expanding until the bracket is reached) for a daily measurement. The number of days to search in each direction (the bracket) should not be so large as to produce unrealistic results. It is possible that no value will be found for a particular month, with the given restraints. In this case, other fill commands (e.g., FillInterpolate()) can be applied to estimate the remaining missing data.

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

\delta Edit NewEndOfMonthTSFromDayTS() Command			
Create a new end of month time series from a daily time series.			
Use the alias to reference the new time series. Only the data interval is changed in output (units, etc. remain).			
The number of days to search in either direction must be non-zero.			
Specifying a number of days > 31 may result in a constant pattern in the output.			
Daily time series:	2003536.DWR.R	ResMeasStorage.Day	
Alias to assign:	Continental	Insert: Select Specifier	Required - use %L for location, etc.
Number of days to search:	15]	Required - must be greater than zero.
Command:	NewEndOfMonthTSFromDayTS(Alias="Continental",DayTSID ="2003536.DWR.ResMeasStorage.Day",Bracket=15)		
Cancel OK			

NewEndOfMonthTSFromDayTS() Command Editor

NewEndOfMonthTSFromDayTS(Parameter=Value,...)

The following older command syntax is updated to the above syntax when a command file is read:

TS Alias = NewEndOfMonthTSFromDayTS (Parameter=Value,...)

Parameter	Description	Default
DayTSID	The time series identifier or alias of the	None – must be specified.
	daily time series to be searched for data.	
Alias	The alias to assign to the time series, as a	None – must be specified.
	literal string or using the special	
	formatting characters listed by the	
	command editor. The alias is a short	
	identifier used by other commands to	
	locate time series for processing, as an	
	alternative to the time series identifier	
	(TSID).	
Bracket	The number of days to search from the	None – must be specified.
	end of the month, in order to find a daily	_
	value to transfer to the end of the month.	

Command Parameters

A sample command file for estimating reservoir contents, using data from the State of Colorado's HydroBase database is:

A sample command file for estimating well levels is:

```
# 384549104445101 - SC01506611ABC
384549104445101.USGS.WellLevel.Day~HydroBase
NewEndOfMonthTSFromDayTS(Alias="WellMonth",
    DayTSID="384549104445101.USGS.WellLevel.Day",Bracket=30)
    FillInterpolate(TSList=AllMatchingTSID,TSID="WellMonth",
    MaxIntervals=0,Transformation=None)
```

To evaluate the results of this command, it is useful to graph both the input and results, changing the graph properties to add symbols to see the individual measurements, as shown in the following figure.



Results of NewEndOfMonthTSFromDayTS() Command

This page is intentionally blank.

Command Reference: NewEnsemble ()

Create a new ensemble and optionally include 1+ time series

Version 10.11.00, 2012-07-18

The NewEnsemble() command creates a new ensemble and optionally inserts 1+ existing time series. For example, use the command to create an ensemble that includes multiple scenarios. Although it is typical that an ensemble contains time series at the same location, it is also possible to use ensembles to group time series at different locations (e.g., to group all time series for stations in a county).

It is envisioned that time series added to the ensemble can optionally be copied and the period changed, in order to isolate the data from the original time series. However, currently the time series from the main processor list are simply associated with the ensemble. Consequently, if other commands change the time series (for example free the time series), the ensemble will reflect the changes. Overcoming this issue will require design changes that need to be evaluated.

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

👌 Edit NewEnsemble() Command		
Create a new ensemble and optionally add time series to it. The time series must have the same data interval (to iterate through data) and units. The original time series will remain available and can be accessed directly in the ensemble.		
Possible ruture enhancements (current Convitbe time series to isolate the s	(iy aisablea): Insemble from additional changes t	n the time series
Specifying the input period for copie	ed time series.	o di c di la serios.
TS list:	~	Optional - indicates time series to add to new ensemble (default=none).
TSID (for TSList=AllMatchingTSID):		
EnsembleID (for TSList=EnsembleID):		×
New ensemble ID:	TestEnsemble	Required - identifier for new ensemble.
New ensemble name:	Test Ensemble	Optional - name for new ensemble.
Input start:		Optional - default is time series period.
Input end:		Optional - default is time series period.
Copy time series?:	~	Optional - whether to copy time series (default=False).
Command:	NewEnsemble (NewEnsem)	bleID="TestEnsemble",NewEnsembleName="Test Ensemble")
Cancel OK		

NewEnsemble () Command Editor

The command syntax is as follows:

NewEnsemble (Parameter=Value,...)

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of:	AllTS

Parameter	Description	Default
	 AllMatchingTSID - all time series that match the TSID (single TSID or TSID with wildcards). AllTS - all time series before the command. EnsembleID - all time series in the ensemble. FirstMatchingTSID - the first time series that matches the TSID (single TSID or TSID with wildcards). LastMatchingTSID - the last time series that matches the TSID (single TSID or TSID with wildcards). SelectedTS - the time series are those selected with the SelectTimeSeries() command. 	
TSID	The time series identifier or alias for the time series to be processed, using the * wildcard character to match multiple time series.	Required when TSList=*TSID
EnsembleID	The ensemble to be processed, if processing an ensemble.	Required when TSList=EnsembleID.
NewEnsembleID	The new ensemble identifier.	None – must be specified.
NewEnsembleName	The name for the new ensemble.	Blank.
InputStart	The date/time to start transferring data from the time series. Envisioned as future enhancement.	Use all data.
InputEnd	The date/time to end transferring data from the time series. Envisioned as future enhancement.	Use all data.
CopyTimeSeries	Copy the time series to the ensemble rather than	Associate time series in
	using time series in the main time series list. This protects the data in the ensemble from general	the main time series list with the new ensemble.
	processing commands. Envisioned as future enhancement.	

A sample command file to create an ensemble from user-defined time series is as follows:

```
# Test creating an ensemble from year interval time series
NewPatternTimeSeries(Alias="ts1",NewTSID="ts1..Flow.Year",
SetStart="1960",SetEnd="2000",Units="ACFT",
PatternValues="1,2,5,8,,20")
NewPatternTimeSeries(Alias="ts2",NewTSID="ts2..Flow.Year",
SetStart="1950",SetEnd="2005",Units="ACFT",
PatternValues="2,4,10,16,,40")
NewEnsemble(TSList=AllTS,
NewEnsembleID="TestEnsemble",NewEnsembleName="Test Ensemble")
```
Command Reference: NewPatternTimeSeries()

Create a new time series containing a pattern of repeating values

The NewPatternTimeSeries() command creates a new time series containing a repeating pattern of numbers. This command is useful for generating data to test other commands.

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

O Edit NewPatternTimeS	eries() Command			
Create a new time series, which a	Create a new time series, which can be referenced using the alias or TSID, using a repeating pattern of values.			
Specify period start and end date	e/times using a precision consistent	with the data interval.		
If the start and end for the perio	d are not set, then a SetOutputPer	riod() command must be specifie	d before the command.	
If the time series has an interval	of irregular, provide the interval to	define data (more options for i	regular data may be added later).	
Alias to assign:	ts2	Insert: Select Specifier	~	Required - use %L for location, etc.
New time series ID:	ts2Streamflow.Day			Required - specify unique TSID information to define time series.
				Edit Clear
Interval for irregular time series:	~			Required for irregular time series - used to initialize data.
Description/name:	Test data			Optional - description for time series.
Start:	1950-01-01]		Optional - starting date/time for data (default=global start).
End:	1951-03-12			Optional - ending date/time for data (default=global end).
Data units:	CFS			Optional - for example: ACFT, CFS, IN.
Missing value:	NaN			Optional - missing data value (default=-999, recommended=NaN).
Pattern values:	5,10,,,75			Required - separate by spaces or commas, blank for missing.
Pattern flags:				Optional - annotates data (default=no flags).
Command:	NewPatternTimeSeries(Alias="ts2",NewTSID="ts2Streamflow.Day",Description="Test data",SetStart="1950-01-01",SetEnd="1951-03-12",Units="CFS",MissingValue=NaN,PatternValues="5,10,,, 75")			
Cancel OK				

NewPatternTimeSeries() Command Editor

The command syntax is as follows:

NewPatternTimeSeries(Parameter=Value,...)

The following older command syntax is updated to the above syntax when a command file is read:

TS Alias = NewPatternTimeSeries(Parameter=Value,...)

Parameter	Description	Default
Alias	The alias to assign to the time series, as a literal string	None – must be
	or using the special formatting characters listed by the	specified.
	command editor. The alias is a short identifier used	
	by other commands to locate time series for	
	processing, as an alternative to the time series	
	identifier (TSID).	
NewTSID	The time series identifier to be assigned to the new	None – must be
	time series, which is useful to avoid confusion with	specified.
	the original time series.	
IrregularInterval	Interval to use to populate irregular time series (e.g.,	None – must be
	1Hour, Month), necessary because data need to be	specified for
	assigned somehow.	irregular time series.
Description	Description for the time series.	None.
SetStart	Start date/time to set data.	None – must be
		specified.
SetEnd	End date/time to set data.	None – must be
		specified.
Units	Units for the data values.	None.
MissingValue	Value to use to indicate missing data values999 is	-999
	the default for historical reasons; however, NaN (not a	
	number) is being phased in and should be specified if	
	possible. Time series can be missing and be flagged.	
PatternValues	Data values, separated by commas. Missing values	None – must be
	can be omitted (e.g., indicate with adjacent commas).	specified.
PatternFlags	Short strings to assign to the values (used to annotate	No flags are
	graphs and other output) separated by commas.	assigned.
	Missing flags can be omitted (e.g., indicate with	
	adjacent commas).	

Command Parameters

Examples

The following example command file illustrates how to create a pattern time series for testing:

```
NewPatternTimeSeries(Alias="ts1",NewTSID="ts1..Streamflow.Day",
    Description="Test data",SetStart="1950-01-01",
    SetEnd="1951-03-12",Units="CFS",PatternValues="5,10,12,13,75")
WriteDateValue(OutputFile=",Example_NewPatternTimeSeries_out.dv")
```

Command Reference: NewStatisticTimeSeries() Create a time series containing a repeating year of statistics determined from a time series

The NewStatisticTimeSeries() command uses data from a time series to calculate a statistic for each interval in the year, and assigns the statistic value to each corresponding interval for the full period. For example, for a statistic of Mean calculated from a daily time series, all January 1 values are averaged and the resulting January 1 values for the entire time series are set to the mean value. Similarly, if monthly data are analyzed, all January values in the result will be set to the mean of the January values in the original time series. This command is useful for superimposing the long-term historical statistic on the original time series or real-time conditions. Leap year statistics are computed from Feb 29 values and are visible only in leap years of the output time series. Missing data in the original time series will by default still result in the statistic being computed, but the AllowMissingCount and MinimumSampleSize parameters control the impacts of missing values.

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

👌 Edit NewStatisticTimeSeries() Command 🛛 🛛 🔀		
Create a time series as a repeating statistic determined from the input time series.		
The statistic is computed from \circ	a sample consisting of values from the same date/ti	ne in each year of the time series.
For example, the Mean statisti	applied to a daily time series will result in the output	It time series having the mean of all January 1 data on each January 1 in the result.
A new time series identifier car	be assigned and is highly recommended if there is	any chance that the new time series will be mistaken for the original.
Time series to analyze (TSID):	ts1	×
Alias to assign:	ts1_mean Insert: Select Specifie	r 💌 Required - use %L for location, etc.
New time series ID:	ts1Streamflow.Month.Mean	Optional - specify unique TSID information to define time series.
		Edit Clear
Statistic:	Mean 💙	Required - statistic to calculate.
Allow missing count:		Optional - number of missing values allowed in sample (default=no limit).
Minimum sample size:		Optional - minimum required sample size (default=determined by statistic).
Analysis start:		Optional - analysis start date/time (default=full time series period).
Analysis end:		Optional - analysis end date/time (default=full time series period).
Output start:		Optional - output start date/time (default=full time series period).
Output end:		Optional - output end date/time (default=full time series period).
Command:	NewStatisticTimeSeries(TSID="t: th.Mean",Statistic=Mean)	51", Alias="ts1_mean", NewTSID="ts1Streamflow.Mon
Cancel OK		

NewStatisticTimeSeries() Command Editor

The command syntax is as follows:

NewStatisticTimeSeries(Parameter=value,...)

The following older command syntax is updated to the above syntax when a command file is read:

```
TS Alias = NewStatisticTimeSeries(Parameter=value,...)
```

Parameter	Description	Default
TSID	The time series identifier (or alias) of the time series	None – must be
	to analyze.	specified.
Alias	The alias to assign to the time series, as a literal string	None – must be
	or using the special formatting characters listed by the	specified.
	command editor. The alias is a short identifier used	
	by other commands to locate time series for	
	processing, as an alternative to the time series	
	identifier (ISID).	XX .1
NewTSID	The time series identifier to be assigned to the new	Use the same
	time series, which is needed to avoid confusion with	identifier as the
	the original time series.	original time series
		with the statistic
		appended to the
		scenario.
Statistic	See the Available Statistics table below.	None – must be
		specified.
Allow	The number of missing values allowed in the source	Allow any number
Missing	interval(s) in order to produce a result. This capability	of missing values.
Count	should be used with care because it may result in data	
	that are not representative of actual conditions.	
MinimumSampleSize	The minimum number of values required in the	Minimum sample
	sample to compute the statistic. If the minimum	size is defined by
	sample size is not available, the result will be set to	the statistic.
	missing.	
AnalysisStart	The date/time for the analysis start, using a precision	Analyze the full
	that matches the original time series. This controls the	period.
	sample size.	
AnalysisEnd	The date/time for the analysis start, using a precision	Analyze the full
	that matches the original time series. This controls	period.
	the sample size.	
OutputStart	The date/time for the output start, using a precision	Output the full
	that matches the original time series. The repeating	period.
	statistic will fill this period.	
OutputEnd	The date/time for the analysis start, using a precision	Output the full
	that matches the original time series. The repeating	period.
	statistic will fill this period.	

Statistic	Description	Limitations	
GeometricMean	Geometric mean of all values in the sample.	All values must be $\geq = 0$.	
Max	Maximum of all values in the sample.	None.	
Mean	Arithmetic mean of all values in the sample.	None.	
Median	Median of all values in the sample.	None.	
Min	Minimum of all values in the sample.	None.	

Available Statistics

Examples

The following example command file illustrates how to generate test data and a corresponding statistics time series:

Test of computing a statistic time series for monthly data,
Assign 2 months of data so that the mean is different from any month
NewPatternTimeSeries(Alias="tsl",NewTSID="tslStreamflow.Month",
Description="Test data",SetStart="1950-01",SetEnd="1951-12",Units="CFS",
PatternValues=".5,1.5,,3.5,4.5,5.5,6.5,7.5,8.5,9.5,10.5,11.5,1.5,2.5,3.5,
4.5,5.5,6.5,7.5,8.5,9.5,10.5,11.5,12.5")
Double the above
NewPatternTimeSeries(Alias="ts2",NewTSID="ts2Streamflow.Month",
Description="Test data",SetStart="1951-01",SetEnd="1952-12",Units="CFS",
PatternValues="1.5,3.5,,7.5,9.5,11.5,13.5,15.5,17.5,19.5,
21.5,23.5,2.5,4.5,6.5,8.5,10.5,12.5,14.5,16.5,18.5,20.5,22.5,24.5")
NewStatisticTimeSeries(TSID="ts1",Alias="ts1_mean",
NewTSID="tslStreamflow.Month.Mean",Statistic=Mean)
NewStatisticTimeSeries(TSID="ts2",Alias="ts2_mean",
NewTSID="ts2Streamflow.Month.Mean",Statistic=Mean)
$WriteDateValue(OutputFile="Results\Test_NewStatisticTimeSeries_Month_Mean_out.dv")$

The following figure illustrates the results. Note that by default the statistic is computed even if missing values exist in the sample. This can be controlled by the AllowMissingCount and MinimumSampleSize parameters.

😹 TSTool -	Time Series - Table			
DATE	ts1, Streamflow, CFS	ts2, Streamflow, CFS	ts1_mean, Streamflow, CFS	ts2_mean, Streamflow, CFS
1950-01	0.50		1.00	
1950-02	1.50		2.00	
1950-03			3.50	
1950-04	3.50		4.00	
1950-05	4.50		5.00	
1950-06	5.50		6.00	
1950-07	6.50		7.00	
1950-08	7.50		8.00	
1950-09	8.50		9.00	
1950-10	9.50		10.00	
1950-11	10.50		11.00	
1950-12	11.50		12.00	
1951-01	1.50	1.50	1.00	2.00
1951-02	2.50	3.50	2.00	4.00
1951-03	3.50		3.50	6.50
1951-04	4.50	7.50	4.00	8.00
1951-05	5.50	9.50	5.00	10.00
1951-06	6.50	11.50	6.00	12.00
1951-07	7.50	13.50	7.00	14.00
1951-08	8.50	15.50	8.00	16.00
1951-09	9.50	17.50	9.00	18.00
1951-10	10.50	19.50	10.00	20.00
1951-11	11.50	21.50	11.00	22.00
1951-12	12.50	23.50	12.00	24.00
1952-01		2.50		2.00
1952-02		4.50		4.00
1952-03		6.50		6.50
1952-04		8.50		8.00
1952-05		10.50		10.00
1952-06		12.50		12.00
1952-07		14.50		14.00
1952-08		16.50		16.00
1952-09		18.50		18.00
1952-10		20.50		20.00
1952-11		22.50		22.00
1952-12		24.50		24.00
	Grap	oh Summary	Save Clos	e

Command Reference: NewStatisticTimeSeriesFromEnsemble()

Create a time series containing a statistic determined from a time series ensemble Version 10.18.00, 2013-02-21

The NewStatisticTimeSeriesFromEnsemble() command uses data from time series in an ensemble to calculate a statistic for each interval in the ensemble, and assigns the statistic value to the corresponding interval in the result. For example, for a statistic of Mean applied to a daily time series, all January 1, 1970 values will be used for the sample and the mean value will be assigned to January 1, 1970 in the output time series. Leap year values will be included if they are included in the period of the ensemble.

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

👌 Edit NewStatistic TimeSer	iesFromEnsemble() Command	
Create a time series as a statistic determined from an ensemble of time series, giving the result an alias.		
A statistic is a value computed from	a sample consisting of values at an interval from each time series in the ensemble.	
It is recommended that a new time s	eries identifier (TSID) be specified for the result to avoid confusion with the original time series.	
Ensemble to analyze (EnsembleID):	TestEnsemble	
New time series ID:	TestStreamflow.6Hour Specify to avoid confusion with TSID from original TS.	
	Edit Clear	
Alias to assign:	Mean Insert: Select Specifier 💌 Required - use %L for location, etc.	
Statistic:	Mean 💌 Required - statistic to calculate.	
Allow missing count:	Optional - number of missing values allowed in sample (default=no limit).	
Minimum sample size:	Optional - minimum required sample size (default=determined by statistic).	
Analysis start:	Optional - analysis start date/time (default=full time series period).	
Analysis end:	Optional - analysis end date/time (default=full time series period).	
Output start:	Optional - output start date/time (default=full time series period).	
Output end:	Optional - output end date/time (default=full time series period).	
Command:	NewStatisticTimeSeriesFromEnsemble(Alias="Mean",EnsembleID="TestEnsemble",NewTS ID="TestStreamflow.6Hour",Statistic=Mean)	
Cancel OK		

NewStatisticTimeSeriesFromEnsemble() Command Editor

The command syntax is as follows:

NewStatisticTimeSeriesFromEnsemble(Parameter=Value,...)

The following older command syntax is updated to the above syntax when a command file is read:

TS Alias = NewStatisticTimeSeriesFromEnsemble(Parameter=Value,...)

Parameter	Description	Default
EnsembleID	The identifier for the ensemble to analyze.	None – must be
		specified.
NewTSID	The time series identifier to be assigned to the new time	None – use the same
	series, which is useful to avoid confusion with the original	identifier as the
	time series. This parameter may be required in the	original time series.
	future.	
Alias	The alias to assign to the time series, as a literal string or	None – must be
	using the special formatting characters listed by the	specified.
	command editor. The alias is a short identifier used by	
	other commands to locate time series for processing, as an	
	alternative to the time series identifier (TSID).	
Statistic	The statistic to compute. See the Available Statistics	None – must be
	table below.	specified.
Allow	The number of missing values allowed in the sample of	Missing values are
Missing	values in order to produce a result. This capability should	ignored in the sample
Count	be used with care because it may result in data that are not	used to compute the
	representative of actual conditions.	statistic.
MinimumSample	The minimum number of values in the sample that are	Use the sample with
Size	required to compute the statistic.	no restrictions,
		although some
		statistics may have
		requirements.
AnalysisStart	The date/time for the analysis start, using a precision that	Analyze the full
	matches the original time series.	period.
AnalysisEnd	The date/time for the analysis start, using a precision that	Analyze the full
	matches the original time series.	period.
OutputStart	The date/time for the output start, using a precision that	Output the full
	matches the original time series. An output period longer	period.
	than the analysis period will result in missing values in	
	output.	
OutputEnd	The date/time for the output start, using a precision that	Output the full
	matches the original time series. An output period longer	period.
	than the analysis period will result in missing values in	
	output.	

Command Parameters

Available Statistics

Statistic	Description	Limitations
Exceedance	The data value corresponding to a 10%	Small sample size will skew –
Probability10	chance of value being exceeded.	see statistic details.
Exceedance	The data value corresponding to a 30%	Small sample size will skew –
Probability30	chance of value being exceeded.	see statistic details.
Exceedance	The data value corresponding to a 50%	Small sample size will skew –
Probability50	chance of value being exceeded.	see statistic details.
Exceedance	The data value corresponding to a 70%	Small sample size will skew –
Probability70	chance of value being exceeded.	see statistic details.
Exceedance	The data value corresponding to a 90%	Small sample size will skew –

Statistic	Description	Limitations
Probability90	chance of value being exceeded.	see statistic details.
GeometricMean	Geometric mean of all values in the sample.	All values must be ≥ 0 .
Max	Maximum of all values in the sample.	None.
Mean	Arithmetic mean of all values in the sample.	None.
Median	Median of all values in the sample.	None.
Min	Minimum of all values in the sample.	None.
Missing	The count of values that are missing.	This statistic will be computed
Count		regardless of
		AllowMissingCount and
		MinimumSampleSize.
Missing	The percent of values that are missing.	See above.
Percent		
Nonmissing	The count of values that are not missing.	See above.
Count		
Nonmissing	The percent of values that are not missing.	See above.
Percent		
Total	Total of values in the sample.	None.

Statistic Details

Statistic	Description
Exceedance	The statistic for each time step in the analysis period is computed as follows:
Probability*	1. The data values are extracted for each trace with missing values being ignored.
	The sample size is <i>n</i> .
	2. The data values are sorted into ascending order.
	3. Exceedance probabilities are computed for the number of sample values
	according to Weibull plotting positions as follows (for $i=1,,n$):
	a. If $n = 1$, the exceedance probability $P_i = 1.0$. This is an extreme case
	due to small sample size.
	b. Otherwise, $P_i = (n - (i - 1))/(n + 1)$. Therefore, when $i = 1$, $P_i = n/(n+1)$
	and when $i=n$, $P_i=1/(n+1)$. The probabilities will be listed from high
	to low value (the opposite order of the sorted data values).
	4. The data value corresponding to the requested probability is calculated by
	iterating over the probabilities until the calculated probability for a value is
	less than the requested probability:
	a. If the first probability satisfies the condition, the computed value is set
	to the minimum value in the sample (no extrapolating past the end).
	b. Otherwise, the value is interpolated from the previous and current
	sample values.
	In no calculated probability is less than the requested probability, the computed value is set to the maximum value in the sample (no avtrapolating past the
	value is set to the maximum value in the sample (no extrapolating past the
	chu).
	To create an exceedance probability plot use several commands with different
	exceedance probability levels (listed low to high). Graphing the time series in a
	har graph with BarOverlap=True will draw the bars on top of each other to
	give the desired appearance. The edges of the colors will represent the specific
	exceedance probabilities and the colored areas will represent ranges of exceedance
	probabilities.

Examples

The following example command file illustrates how to compute the mean statistic for one monthly data:

```
# Test computing a statistic time series for Month data where Statistic=Mean
StartLog(LogFile="Results/Test_NewStatisticTimeSeriesFromEnsemble_Month_Mean.TSTool.log")
# Define 2 years of data that when averaged equal even numbers
# The 2nd time series is shifted by 1 from the first.
# Include missing values in the first time series but not the second.
NewPatternTimeSeries(Alias="ts1",NewTSID="ts1..Streamflow.Month",
    Description="test data 1",SetStart="2000-01",SetEnd="2001-12",Units="CFS",
    PatternValues=".5,1.5,,3.5,4.5,5.5,6.5,7.5,8.5,9.5,10.5,11.5,
    1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5, 9.5, 10.5, 11.5, 12.5")
NewPatternTimeSeries(Alias="ts2",NewTSID="ts2..Streamflow.Month",
    Description="test data 2",SetStart="2000-01",SetEnd="2001-12",Units="CFS",
    PatternValues="1.5,2.5,3.5,4.5,5.5,6.5,7.5,8.5,9.5,10.5,11.5,12.5,
    2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5, 9.5, 10.5, 11.5, 12.5, 13.5")
# Create an ensemble to hold the above time series
NewEnsemble(TSList=AllTS,NewEnsembleID="TestEnsemble",NewEnsembleName="Test Ensemble")
# Compute the statistic
NewStatisticTimeSeriesFromEnsemble(Alias="Mean",EnsembleID="TestEnsemble",
   NewTSID="Test..Streamflow.Month.Mean",Statistic=Mean)
```

The following figure illustrates the results:

DATE	ts 1, Streamflow, CFS	ts2, Streamflow, CFS	Mean, Streamflow, CFS
2000-01	0.50	1.50	1.00
2000-02	1.50	2.50	2.00
2000-03	1	3.50	3.50
2000-04	3.50	4.50	4.00
2000-05	4.50	5.50	5.00
2000-06	5.50	6.50	6.00
2000-07	6.50	7.50	7.00
2000-08	7.50	8.50	8.00
2000-09	8.50	9.50	9.00
2000-10	9.50	10.50	10.00
2000-11	10.50	11.50	41.00
2000-12	11.50	12.50	12.00
2001-01	1.50	2.50	2.00
2001-02	2.50	3.50	3.00
2001-03	3.50	4.50	4.00
2001-04	4.50	5.50	5.00
2001-05	5.50	6.50	6.00
2001-06	6.50	7.50	7.00
2001-07	7.50	8.50	8.00
2001-08	8.50	9.50	9.00
2001-09	9.50	10.50	10.00
2001-10	10.50	11.50	11.00
2001-11	11.50	12.50	12.00
2001-12	12.50	13.50	13.00

NewStatisticTimeSeriesFromEnsemble() Command Results

This page is intentionally blank.

Command Reference: NewStatisticYearTS() Create a new yearly time series containing a statistic determined from each year of the input time series

The NewStatisticYearTS() command creates a new yearly time series, where each yearly value in the resulting time series contains a statistic determined from the sample of points from the corresponding year in the original time series. For example, if the original time series has a daily time step, then the sample that is analyzed will contain 365 or 366 values (depending on leap year). Calendar years are used by default; however, the OutputYearType parameter can be used to specify that different year types are analyzed. Other commands (e.g., ChangeInterval()) can produce a similar result for a limited number of statistics, for example converting a monthly time series to an annual total or mean. See also the NewStatisticTimeSeries(), NewStatisticTimeSeriesFromEnsemble(), CalculateTimeSeriesStatistic(), and CheckTimeSeries() commands.

For hourly and finer interval, values are considered to be in a year when the year in the date/time matches the year of interested. This may lead to some issues if the last value in a year is actually recorded at hour 0 or later of the following year.

🚯 Edit NewStatisticYearTS() Command					
Create a time series where each value is a statistic calculated from a year of data from the input time series. The output time series has an interval of year.					
It is recommended that new tin	It is recommended that new time series identifier (TSID) information be specified for the output time series to avoid confusing the output with the original.				
Time series to analyze (TSID):	3553.NOAA.TempMin.Day	▼			
Alias to assign:	3553_FrostDateL285 Insert: Select Specifier 💌	Required - use %L for location, etc.			
New time series ID:	3553.NOAA.FrostDateL28S.Year	Recommended - to avoid confusion with TSID from original time series.			
		Edit Clear			
Statistic:	DayOfLastLE 🛛 👻	Required - statistic to calculate.			
Test value:	28	Optional - test value (required for comparison statistics).			
Allow missing count:		Optional - number of missing values allowed in analysis interval (default=allow missing).			
Minimum sample size:		Optional - minimum required sample size (default=determined by statistic).			
Output year type:	×	Optional - to define year span (default=Calendar).			
Analysis start:		Optional - analysis start date/time (default=full time series period).			
Analysis end:		Optional - analysis end date/time (default=full time series period).			
Analysis window:	Month: Day: Hour:	Optional - analysis window within input year (default=full year).			
Search start:	06/30	Optional - search start (needed for some statistics, default=full year).			
Command:	NewStatisticYearTS(TSID="3553.NOAA.TempMin.Day",Alias="3553_FrostDateL28S",NewTSID="355 3.NOAA.FrostDateL28S.Year",Statistic=DayOfLastLE,TestValue=28,SearchStart="06/30")				
Cancel OK					
		NewStatisticYearTS			

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

NewStatisticYearTS() Command Editor

The command syntax is as follows:

```
NewStatisticYearTS(Parameter=value,...)
```

The following older command syntax is updated to the above syntax when a command file is read:

```
TS Alias = NewStatisticYearTS (Parameter=value,...)
```

Parameter	Description	Default
TSID	The time series identifier (or alias) of the time series to	None – must be
	analyze.	specified.
Alias	The alias to assign to the time series, as a literal string or using the special formatting characters listed by the command editor. The alias is a short identifier used by other commands to locate time series for processing, as an alternative to the time series identifier (TSID).	None – must be specified.
NewTSID	The time series identifier to be assigned to the new time series, which is useful to avoid confusion with the original time series.	Use the same identifier as the original time series, with an interval of Year and a scenario matching the statistic.
Statistic	See the Available Statistics table below.	None – must be specified.
TestValue	A test value used when analyzing the statistic.	This parameter is required for some statistics and not used for others. See the statistics table below.
AllowMissing	The number of missing values allowed in the source	Allow any number of
Count	interval(s) in order to produce a result. If an analysis window is specified (default is to analyze full years), then missing values outside of the analysis window are not considered as missing. Gaps at the end of the time series will be considered missing if within the analysis window.	missing values.
Minimum	The minimum sample size in order to compute the	No minimum,
SampleSize	statistic.	although the statistic may have requirements.
OutputYearType	The output year type. For example, an output year type of NovToOct spans November of the previous calendar year to October of the current calendar year. All other parameters should still be specified in calendar year and the AnalysisWindowStart can have a month that is prior to the AnalysisWindowEnd month.	Calendar
AnalysisStart	The starting date/time for the analysis using calendar dates (e.g., 2001-01-01), with precision consistent with	Analyze the full period, extending the

Parameter	Description	Default
	the time series interval. This will limit the data being	period to include full
	analyzed at the ends of the time series and controls the	years.
	length of the output time series. The analysis period is	
	typically set to align with years consistent with the	
	output year type.	
AnalysisEnd	The ending date/time for the analysis using calendar	Analyze the full
	dates (e.g., 2001-01-01), with precision consistent with	period, extending the
	the time series interval. This will limit the data being	period to include full
	analyzed at the ends of the time series and controls the	years.
	length of the output time series. The analysis period is	
	typically set to align with years consistent with the	
	output year type.	
Analysis	The calendar date/time for the analysis start within each	Analyze the full year.
WindowStart	year. Specify using the format MM, MM–DD, MM–DD hh,	
	or MM-DD hh:mm, consistent with the time series	
	interval precision. A year of 2000 will be used	
	internally to parse the date/time. Use this parameter to	
	limit data processing within the year, for example to	
	analyze only a season. Data will be considered missing	
	only if missing within this analysis window. If	
	specifying for other than calendar year, the analysis	
	window start month may be greater than the analysis	
	window end month.	
Analysis	Specify date/time for the analysis end within each year.	Analyze the full year.
WindowEnd	See AnalysisWindowStart for details.	
SearchStart	Within the analysis window, this indicates the starting	Use the analysis
	date/time for the search. Specify using the format MM,	window start and
	MM-DD, MM-DD hh, or MM-DD hh:mm, consistent with	end. Search forward
	the time series interval precision. A year of 2000 will	for most statistics.
	be used internally to parse the date/time. This parameter	Search backward for
	is useful in cases where the processing considers	DayOfLast* and
	seasonal aspects of the analysis window; for example,	MonthOfLast*
	use when determining frost dates (when temperature is	statistics.
	less than or equal to freezing) to ensure that the search	
	starts from the middle of the normal growing season.	
	Searches move forward in time except for the following	
	statistics, in which case SearchStart will be the start	
	of the search window, but will be the last value checked:	
	DayOfLast*,MonthOfLast*.	

Available Statistics

The following statistics are computed from a sample determined using the analysis window. If no analysis window is specified, then the default is to analyze complete years, where the years correspond to the OutputYearType. For example, for OutputYearType=NovToDec, November 1, 2000 to October 31, 2001 from the input corresponds to output year 2001.

Statistic	Description	Limitations
DayOfCentroid	The day of the year (1-366) that is the centroid	Input time series must be

Statistic	Description	Limitations
	of the values, computed as	daily or smaller interval.
	sum(DayOfYear*value)/sum(values).	
DayOfFirstGE	Julian day of the year (1-366, relative to the	Input time series must be
	start of the OutputYearType) for the first	daily or smaller interval.
	data value >= TestValue. Searches start at	
	the start of the analysis window and move	
	forward.	
DayOfFirstGT	Similar to DayOfFirstGE, for values >	Input time series must be
	TestValue.	daily or smaller interval.
DayOfFirstLE	Similar to DayOfFirstGE, for values <=	Input time series must be
	TestValue.	daily or smaller interval.
DayOfFirstLT	Similar to DayOfFirstGE, for values <	Input time series must be
	TestValue.	daily or smaller interval.
DayOfLastGE	Julian day of the year (1-366, relative to the	Input time series must be
	start of the OutputYearType) for the last	daily or smaller interval.
	data value >= TestValue. Searches start at	
	the start of the analysis window and move	
	backward.	
DayOfLastGT	Similar to DayOfLastGE, for values >	Input time series must be
	TestValue.	daily or smaller interval.
DayOfLastLE	Similar to DayOfLastGE, for values <=	Input time series must be
	TestValue.	daily or smaller interval.
DayOfLastLT	Similar to DayOfLastGE, for values <	Input time series must be
_	TestValue.	daily or smaller interval.
DayOfMax	Julian day of the year (1-366, relative to the	Input time series must be
-	start of the OutputYearType) for the first	daily or smaller interval.
	maximum value in the time series.	2
DayOfMin	Julian day of the year (1-366, relative to the	Input time series must be
	start of the OutputYearType) for the first	daily or smaller interval.
	minimum value in the time series.	
GECount	Count of values in a year >= TestValue.	
GEPercent	Percent of values in a year >= TestValue,	
	based on the total number of points in the year.	
GTCount	Count of values in a year > TestValue.	
GTPercent	Percent of values in a year > TestValue,	
	based on the total number of points in the year.	
LECount	Count of values in a year <= TestValue.	
LEPercent	Percent of values in a year <= TestValue,	
	based on the total number of points in the year.	
LTCount	Count of values in a year < TestValue.	
LTPercent	Percent of values in a year < TestValue.	
	based on the total number of points in the year.	
Max	Maximum value in a year.	
Mean	Mean of values in a year.	
Min	Minimum value in a year.	
MissingCount	Number of missing values in a year.	
MissingPercent	Percent of missing values in a year.	

Statistic	Description	Limitations
MonthOfCentroid	The month of the year (1-12) that is the centroid	Input time series must be
	of the values, computed as	monthly or smaller
	sum(MonthOfYear*value)/sum(values).	interval.
MonthOfFirstGE	Month the year (1-12, relative to the start of the	Input time series must be
	OutputYearType) for the first data value >=	monthly or smaller
	TestValue. Searches start at the start of the	interval.
	analysis window and move forward.	
MonthOfFirstGT	Similar to DayOfFirstGE, for values >	Input time series must be
	TestValue.	monthly or smaller
		interval.
MonthOfFirstLE	Similar to DayOfFirstGE, for values <=	Input time series must be
	TestValue.	monthly or smaller
		interval.
MonthOfFirstLT	Similar to DayOfFirstGE, for values <	Input time series must be
	TestValue.	monthly or smaller
		interval.
MonthOfLastGE	Month of the year (1-12, relative to the start of	Input time series must be
	the OutputYearType) for the last data value	monthly or smaller
	>= TestValue. Searches start at the end of	interval.
	the analysis window and move backward.	
MonthOfLastGT	Similar to DayOfLastGE, for values >	Input time series must be
	TestValue.	monthly or smaller
		interval.
MonthOfLastLE	Similar to DayOfLastGE, for values <=	Input time series must be
	TestValue.	monthly or smaller
		interval.
MonthOfLastLT	Similar to DayOfLastGE, for values <	Input time series must be
	TestValue.	monthly or smaller
NorthOfMor	Month of the way (1, 12, relative to the start of	Interval.
MOIICHOIMAX	Month of the year $(1-12)$, feative to the start of the Output $X_{2} = 2 \sqrt{1-12}$, for the first maximum	monthly or smaller
	the Output Year Type) for the first maximum	interval
MonthOfMin	Value in the time series.	Interval.
MOLICIOLMII	Month of the year $(1-12)$, feative to the start of the Output $X_{2} = -12$ for the first minimum	monthly or smaller
	ule output year type) for the first minimum	interval
NonMiggingCount	Number of non-missing volves in a vest	
NonMiggingDowgont	Percent of non-missing values in a year.	
Total	Total of values in a year.	
IULAI	I Otal OI values III a year.	

Example

The following example commands file computes the last spring frost date for 28 degrees and 32 degrees, searching backwards from June 30 each year, and the first fall frost date for 32 and 28 degrees, searching forwards from July 1 each year:

```
StartLog(LogFile="FrostDates_HydroBase.log")
SetOutputPeriod(OutputStart="1950-01",OutputEnd="2004-12")
# 3553 - GREELEY UNC
3553.NOAA.TempMin.Day~HydroBase
NewStatisticYearTS(TSID="3553.NOAA.TempMin.Day",Alias="3553 FrostDateL28S",
  NewTSID="3553.NOAA.FrostDateL28S.Year",
  Statistic=DayOfLastLE,TestValue=28,SearchStart="06/30")
NewStatisticYearTS(TSID="3553.NOAA.TempMin.Day",Alias=" FrostDateL32S",
  NewTSID="3553.NOAA.FrostDateL32S.Year",
  Statistic=DayOfLastLE,TestValue=32,SearchStart="06/30")
NewStatisticYearTS(TSID="3553.NOAA.TempMin.Day",Alias="3553_FrostDateF32F",
  NewTSID="3553.NOAA.FrostDateF32F.Year",
  Statistic=DayOfFirstLE,TestValue=32,SearchStart="07/01")
NewStatisticYearTS(TSID="3553.NOAA.TempMin.Day",Alias="3553_FrostDateF28F",
  NewTSID="3553.NOAA.FrostDateF28F.Year",
  Statistic=DayOfFirstLE,TestValue=28,SearchStart="07/01")
Free(TSID="*.*.TempMin.*")
WriteStateCU(OutputFile="Results/Test.FrostDates")
```

Command Reference: NewTable ()

Create a new table

The NewTable() command creates a table with named columns, each of which is a specified data type. Tables are used to hold information about data objects, such as statistics for time series. Commands like CalculateTimeSeriesStatistic() can add information to tables. Tables can be written as final data products or artifacts of processing. Characteristics of the table are as follows:

- Each column can only contain a single data type
- The default precision for numbers for display and output is 2 digits after the decimal additional formatting features may be available in write commands and may be added later
- Tables are referenced using the TableID
- Cells in tables are referenced using the column name and cell values that identify rows (such as time series identifiers)

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

Edit NewTable() (Command				
This command creates a new table. The table can then be used by other commands, for example to store statistics computed from time series.					
Columns can be defi	Columns can be defined here and can also be added later with other commands.				
Columns should be defined using syntax: name,type;name,type					
Available types are:					
datetime - date a	nd time				
double - double p	recision number				
float - single prec	ision number				
integer - integer					
short - short integ	ger				
string					
Table ID:	Table1 Required - unique identifier for the table.				
	datetime1, datetime; double1, double; float1, float; integer1, integer; short1, sh				
Colored de Colored	ort;string 1,string				
Column derinitions:					
	NewTable(Table1D="Table1",Columns="datetime1,datetime;double1,double;floa				
Command:	tl,float;integer1,integer;short1,short;string 1,string")				
	Cancel OK				
	NewTable				

NewTable () Command Editor

The command syntax is as follows:

```
NewTable (Parameter=Value,...)
```

Parameter	Description	Default
TableID	Identifier for the table – should be unique	None – must be specified.
	among tables that are defined.	
Columns	The column names and data types are	No columns will be defined.
	defined using the format	
	ColumnName,DataType;	
	ColumnName,DataType. Column	
	names can contain spaces; however,	
	simple short names are generally handled	
	better by display features and minimize	
	errors in referencing the columns. Data	
	types are specified using the following	
	strings:	
	 datetime – date and time 	
	• double – double precision number	
	• float – single precision number	
	• integer – integer (-2147483648 to	
	2147483647)	
	• short – short integer (-32768 to	
	32767)	
	• string – Unicode string	

Command Reference: NewTimeSeries()

Create a new time series

Version 10.21.00, 2013-06-10

The NewTimeSeries() command creates a new time series and assigns it an alias. This time series then can be manipulated with other commands. The command is useful, for example, to create a new time series to receive the results of a series of manipulations, rather than having the results accumulate in the first time series. See also the NewPatternTimeSeries() command, which initializes a time series with a repeating pattern of values. Subsequent manipulation of the time series may require use of the SetTimeSeriesProperty() and other commands to ensure that the new time series properties are as desired.

The following dialog is used to edit the command and illustrates the syntax for the command. The new time series identifier, which provides critical information including the data interval, is edited by pressing the *Edit* button.

🗞 Edit NewTimeSeries() Command 🛛 🛛 🔀					
Create a new time series, which can be referenced using the alias or TSID.					
Specify period start	and end date/times using a precision consistent (with the data interval.			
If the start and end	for the period are not set, then a SetOutputPeri	od() command must be specified before the command.			
Alias to assign:	Select Specifier 💙 => ts2	Required - use %L for location, etc.			
New time series ID:	ts2Streamflow.Day	Required - specify unique TSID information to define time series.			
	-	Edit Clear			
Description/Name:	Test time series 2	Optional - description for time series.			
Start:	2000-01-01	Optional - starting date/time for data (default=global start).			
End:	2001-01-14	Optional - ending date/time for data (default=global end).			
Data units:	CFS	Optional - for example: ACFT, CFS, IN (default=no units).			
Missing value:	NaN	Optional - missing data value (default=-999, recommended=NaN).			
Initial value:		Optional - default is to initialize with the missing value.			
Initial function:	~	Optional - function to initialize data.			
Command:	NewTimeSeries(Alias="ts2",NewTSID="ts2Streamflow.Day",Description= "Test time series 2",SetStart="2000-01-01",SetEnd="2001-01-14",Units="CFS",MissingValu e=NaN)				
	Cancel OK				

NewTimeSeries() Command Editor

The command syntax is as follows:

```
NewTimeSeries(Parameter=Value,...)
```

The following older command syntax is updated to the above syntax when a command file is read:

TS Alias = NewTimeSeries(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
Alias	The alias to assign to the time series, as a literal string or using the special formatting characters listed by the command editor. The alias is a short identifier used by other commands to locate time series for processing, as an alternative to the time series identifier (TSID).	None – must be specified.
NewTSID	The time series identifier of the new time series. The editor dialog formats the identifier from its parts.	None – must be specified with at least minimal information (location, data type, and interval).
Description	The description for the time series, used in output.	Blank.
SetStart	The start of the time series data period.	Use the start from SetOutputPeriod().
SetEnd	The end of the time series data period.	Use the end from SetOutputPeriod().
Units	Data units for the time series.	Blank.
MissingValue	Value for missing data values999 is the default for historical reasons; however, NaN (not a number) is being phased in and should be specified if possible.	-999
InitialValue	The value to initialize the time series.	Initialize the time series to missing data.
InitialFunction	 The function to use to initialize time series data values. This parameter can be used to generate data for testing to simplify visual inspection of results. DATE_YYYY - 4-digit year DATE_YYYYMMDD - year, month, and day DATE_YYYYMMDD_hh - year, month, and day, with decimal as hour DATE_YYYYMMDD_hhmm - year, month, and day, with decimal as hour and minute RANDOM_0_1 - random number >= 0 and < 1 RANDOM_0_1000 - random number >= 0 and < 1000 	Initialize the time series to missing data.

The example command file shown below creates a new time series and initializes it to a constant of 20 CFS. Uncommenting the first command would allow the SetStart and SetEnd parameters to be removed from the NewTimeSeries() command. The interval (Month below) must match a recognized type but the other parts of the identifier such as data type are user-defined.

```
#SetOutputPeriod(OutputStart="1950-01",OutputEnd="2002-12")
NewTimeSeries(Alias="station1",NewTSID="Station1.MyModel.Streamflow.Month",
    Description="Example Description",SetStart="1950-01",
    SetEnd="2002-12",Units="CFS",InitialValue=20)
```

Command Reference: NewTreeView()

Create a new tree view using a definition file

Version 09.07.00, 2010-07-20

The NewTreeView() command creates a tree view, which is a hierarchical listing of time series. The resulting view is dispayed in the *Views* section of the TSTool *Results* area and provides interactive access to data. The view is defined using a simple text file, as shown in the following example:

```
# Test data for displaying a tree view of time series results
Label: Top-level label
    TS: ts1*
    Label: Second-level label
        TS: ts2*
    Label: Another second-level label
        TS: ts3*
```

Tree view definition files have the following characteristics:

- Comments are indicated by lines starting with #.
- Indentations indicate the level (branch) in the tree:
 - Use the tab character to indicate indentation
 - The indentation on one row cannot be more than 1 greater than the previous row
- The content for the tree is indicated by keywords:
 - Label: indicates that the string following the colon will be used to label a branch.
 A single top-level label is required
 - TS: indicates that a time series identifier pattern will be used to identify time series in the tree. Wildcard conventions follow rules consistent with the TSList=AllMatchingTSID, TSID=... command parameters.

The following figure illustrates the resulting view that is displayed in TSTool for the above example, using contrived data. The time series in the tree view can be selected and a pop-up menu can be used to generate graphs. Consequently, the view allows the results of processing to be presented in a way that is more customized than a simple list. It is envisioned that additional functionality will be implemented, for example to output the view as HTML with navigation links.

-Results	
Ensembles Output Files Problems Tables Time Series Views View1	
□C⊃ Top-level label ts1Demand.Year	
ts1Supply.Year	
Second-level label	
⊕ — 🛅 Another second-level label	

NewTreeView_Results

Example of Tree View in TSTool Results

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

Edit NewTreeView() Command				
This comma	nd creates a new time series tree view, which is a hierarchical list of time series.			
View ID:	View1 Required - unique identifier for the view.			
Input file:	Data\tree1.txt Browse			
	NewTreeView(InputFile="Data\tree1.txt")			
Command:				
	Add Working Directory To File Cancel OK			
	Ne			

NewTreeView() Command Editor

The command syntax is as follows:

NewTreeView(Parameter=Value,...)

Parameter	Description	Default
ViewID	Identifier to assign to the view, which	None – must be specified.
	allows the view to be used with other	
	commands.	
InputFile	The name of the view definition file to	None – must be specified.
	read, as an absolute path or relative to the	
	command file location.	

Command Reference: Normalize()

Create a normalized time series

Version 10.00.01, 2011-05-15

The Normalize() command creates a new normalized time series from an existing time series, assigning an alias to the result. Normalized time series are useful for analyzing trends and relationships and for allowing time series with different units to be plotted or analyzed together. For example, the range of data values can be normalized to the range 0 to 1. The alias that is assigned to the time series can be referenced by other commands.

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

💧 Edit Normalize() Comr	nand		
Create a new time series by nor	malizing the data f	rom a time series.	
Use the alias to reference the n	ew time series. Da	ita units are set to blank because the re	esult is dimensionless.
Time Series to Normalize:	06730500.USGS.:	5treamflow.Month	✓
Alias to assign:	NormalizedTS	Insert: Select Specifier	Required - use %L for location, etc.
Minimum data value to process:	MinFromTS 🔽		Required.
Minimum output value:	0.0		Required - for example 0.0.
Maximum output value:	1.0		Required - for example 1.0.
Command:	Normalize(7 ormalized79 lue=1.0)	ISID="06730500.USGS.Stre 5",MinValueMethod=MinFro	amflow.Month",Alias="N mTS,MinValue=0.0,MaxVa
		Cancel OK	

Normalize() Command Editor

The command syntax is as follows:

Normalize(Parameter=Value,...)

The following older command syntax is updated to the above syntax when a command file is read:

TS Alias = Normalize(Parameter=Value,...)

Parameter	Description	Default
TSID	The time series identifier or alias for the time series to	None – must be specified.
	be normalized.	
Alias	The alias to assign to the time series, as a literal string or	None – must be specified.
	using the special formatting characters listed by the	
	command editor. The alias is a short identifier used by	
	other commands to locate time series for processing, as	

Parameter	Description	Default
	an alternative to the time series identifier (TSID).	
MinValue Method	Indicates how to determine the minimum data value to process, one of:	None – must be specified.
	• MINFFORTS – get the minimum value from the time series (typical)	
	• MinZero – use zero (e.g., if negative values are to be ignored)	
MinValue	The minimum normalized value (e.g., 0).	None – must be specified.
MaxValue	The maximum normalized value (e.g., 1).	None – must be specified.

A sample command file to process data from the State of Colorado's HydroBase database is as follows:

```
# 06730500 - BOULDER CREEK AT MOUTH, NEAR LONGMONT, CO.
06730500.USGS.Streamflow.Month~HydroBase
Normalize(TSID="06730500.USGS.Streamflow.Month",Alias="NormalizedTS",
MinValueMethod=MinFromTS,MinValue=0.0,MaxValue=1.0)
```

The results are as follows:



Results of Normalize() Command

Command Reference: OpenHydroBase()

Open a connection to a HydroBase database

Version 10.13.00, 2012-09-13

This command will be phased out in the future. Instead, define HydroBase datastores (see the HydroBase Datastore appendix), where the datastore name is equivalent to the InputName parameter. The OpenHydroBase() command opens a connection to a HydroBase database, allowing data to be read from the database (e.g., with ReadHydroBase() commands and time series identifiers that have ~HydroBase input types). This command is not typically used for interactive sessions but may be inserted to run in batch only mode to allow a specific database and commands files to be distributed. It may also be used in cases where time series are read from different HydroBase() commands would be used. When connecting to a SQL Server database, a connection will be tried for SQL Server (Express) and older MSDE databases. If both fail, a warning will be shown.

The following dialog is used to edit this command and illustrates the command syntax. The **Database** *type* is used to control settings for parameters and is not itself a parameter.

👌 Edit OpenHydrof	Base() command	X
This command will be	phased out in the future as HydroB	ase datastore support is phased in.
This command opens a co	nnection to a HydroBase database, closin	ng the previous connection with the same input name.
This command is used, for	r example, when making connections to m	ore than one HydroBase database, or running in batch mode.
The RunMode can also be	e set to control whether the command is ru	un in batch mode, in interactive sessions, or both.
The connection can be ma	ade either by specifying a database serve	er/name for SQL Server,
or an ODBC Data Source	Name (DSN) for a Microsoft Access Hydro	Base database (only for very old HydroBase databases).
Database type:	SQL Server 🔽	Required - indicates whether a database server/name or ODBC DSN is specified below.
Database server:	lonetree\CDSS 🛛 👻	Required - when using SQL Server.
Database name:	HydroBase 😽	Required - when using SQL Server.
ODBC DSN:	×	Optional - only used with Microsoft Access.
Input name:		Optional - input name for connection, to be used with time series identifiers.
Use stored procedures?:	✓	Optional - default is true for SQL Server, ignored for Access.
Run Mode:	✓	Optional - default is GUI and batch mode.
Command:	OpenHydroBase (DatabaseSe)	rver="lonetree\CDSS",DatabaseName="HydroBase")
	[Cancel OK
		OpenHydroBase

OpenHydroBase() Command Editor

The command syntax is as follows:

OpenHydroBase(Parameter=Value,...)

Parameter	Description	Default
Database	Used with a SQL Server HydroBase. Specify the SQL	Required if a SQL
Server	Server database machine name. A list of choices will be	Server database is used.
	shown, corresponding to properties in the CDSS.cfg	

	configuration file. The generic value	
	DatabaseServer=local will automatically be	
	translated to the name of the local computer.	
Database	Used with a SQL Server HydroBase. The name of the	HydroBase
Name	database typically follows a pattern similar to:	
	HydroBase_CO_YYYYMMDD. A list of choices will be	
	shown, corresponding to properties in the CDSS.cfg	
	configuration file.	
OdbcDsn	The ODBC DSN to use for the connection, used only when	Required if a Microsoft
	working with a Microsoft Access database.	Access database is used.
InputName	The input name corresponding to the	Blank (no input name).
	~InputType~InputName information in time series	
	identifiers. This is used when more than one HydroBase	
	connection is used in the same commands file.	
UseStored	Used with SQL Server, indicating whether stored	True (used stored
Procedures	procedures are used. Stored procedures are the default and	procedures).
	should be used except when testing software.	-
RunMode	Indicates when the command should be run, one of:	GUIAndBatch
	• BatchOnly – run the command only in batch mode.	
	• GUIOnly – run the command only in GUI mode.	
	• GUIAndBatch – run the command in batch and GUI	
	mode.	

The following example command file illustrates how to connect to a SQL Server database running on a machine named "sopris":

```
StartLog(LogFile="Results/Example_OpenHydroBase_DatabaseName.TSTool.log")
OpenHydroBase(DatabaseServer="sopris",DatabaseName="HydroBase_CO_20060816")
ReadHydroBase(TSID="BOXHUDCO.DWR.Streamflow.Month",Alias="ts")
```

Example_OpenHydroBase_DatabaseName

The following example command file illustrates how to make two HydroBase database connections, in this case to test whether the stored procedure and SQL queries return the same results (the InputName parameter is used to tell TSTool which connection to use when reading data based on time series identifiers):

```
OpenHydroBase(DatabaseServer="hbserver",RunMode=GUIAndBatch,
UseStoredProcedures=True,InputName="SP")
OpenHydroBase(DatabaseServer="hbserver",RunMode=GUIAndBatch,
UseStoredProcedures=False,InputName="NoSP")
ReadHydroBase(TSID="BOXHUDCO.DWR.Streamflow.Month~HydroBase~SP",Alias="ts_sp")
ReadHydroBase(TSID="BOXHUDCO.DWR.Streamflow.Month~HydroBase~NoSP",Alias="ts_nosp")
Example OpenHydroBase TwoConnections_ISTool
```

The following example commands file illustrates how to connect to a Microsoft Access database (although Microsoft Access databases are no longer supported):

OpenHydroBase(RunMode=GUIAndBatch,OdbcDsn="HydroBase_DIV1_20030701")

Command Reference: PrintTextFile()

Print a text file Version 10.00.03, 2011-07-01

The PrintTextFile() command prints a text file to a physical or virtual (e.g., PDF file) printer. This command is used in testing to verify print features but also can be used in production to automate printing. Printing is highly dependent on the features available from a printer. The command attempts to list printer options to configure command parameters. However, some options are listed based on what a printer <u>can</u> do, but this may require physically changing/selecting paper trays, using manual feed, etc. The command will be enhanced in the future to specify features including tray selection but currently it is intended for use with common printer settings. If advanced settings are needed beyond the properties available in the command, use ShowDialog=True (displaying the dialog will pause command execution).

The following dialog is used to edit the command and illustrates the syntax for the command (in this case printing to a Microsoft XPS file).

👌 Edit PrintTex	tFile() command	
This command prints (a text file using basic formatting.	
Determining supporte	d printer settings may take a few seconds.	
The margins agree wi	th the orientation (e.g., for letter size portrait orientation, left margin	is long edge; for landscape, left margin is for short edge).
It is recommended th	at the file name is relative to the working directory, which is:	
C:\Develop\TSTool	_SourceBuild\TSTool\test\regression\commands\general\PrintTextFile 	
File to print:	Data/50lines.txt	Browse
Printer name:	Microsoft XPS Document Writer C	Optional - printer name (default=default printer).
Paper size:	na-letter - North American Letter (8.50x11.00 in) 🛛 🗸 C	Optional - paper size name (default=printer-specific).
Paper source (tray):	C 0	Optional - paper source (default=default source/tray).
Orientation:	Landscape 🔽 C	Optional - paper orientation (default=portrait).
Left margin:	.5 0	Optional - left margin, inches (default=.75).
Right margin:	.6 0	Optional - right margin, inches (default=.75).
Top margin:	.7 0	Optional - top margin, inches (default=.75).
Bottom margin:	.8 C	Optional - bottom margin, inches (default=.75).
Lines per page:	c	Optional - lines per page (default=depends on page size).
Header:	Header C	Optional - for top of each page (default=none).
Footer:	Footer C	Optional - for bottom of each page (default=none).
Show line count:	True 💙 C	Optional - show line count at left (default=False).
Show page count:	c	Optional - show page count at bottom (default=True).
Pages:	c	Optional - pages to print N,N-N, etc. (default=print all).
Double-sided?:	c c	Optional - print double-sided? (default=False).
Print file:	results/Test_PrintTextFile_Landscape_Letter_1Page_XPS.xps	Browse
Show dialog?:	c	Dptional - show printer dialog? (default=False).
If not found?:	 c	Optional - action if file not found (default=Warn).
Command:	PrintTextFile(InputFile="Data/50lines.txt" Writer",PaperSize="na-letter",Orientation= inTop=.7,MarginBottom=.8,Header="Header",H e="results/Test_PrintTextFile_Landscape_Let	",PrinterName="Microsoft XPS Document =Landscape,MarginLeft=.5,MarginRight=.6,Marg Footer="Footer",ShowLineCount=True,OutputFil etter_1Page_XPS.xps")
	Add Working Directory (Input) Add Working Dire	ectory (Output) Cancel OK PriotTextEllo

PrintTextFile() Command Editor

The command syntax is as follows:

PrintTextFile(Parameter=Value,...)

Parameter	Description	Default
InputFile	The name of the text file to print.	None – must be
		specified.
PrinterName	The name of the printer to use (e.g.,	The default printer
	\\ <u>MyComputer</u> \ <u>MyPrinter</u> or Adobe PDF)	will be used.
PaperSize	The paper size to print. Because there are a number of	The default for the
	standards for paper, the size is specified as standard-	printer will be used.
	sizename (e.g., na-letter for North American Lettter).	
	For information on paper sizes, see:	
	http://en.wikipedia.org/wiki/Paper_size.	
PaperSource	The tray for the paper – currently not enabled. Use	The PaperSize is
	ShowDialog=True to select.	used.
Orientation	The paper orientation.	The default for the
		printer and paper size
		will be used.
MarginLeft	The left margin for the orientation, inches.	See above.
MarginRight	The right margin for the orientation, inches.	See above.
MarginTop	The top margin for the orientation, inches.	See above.
MarginBottom	The bottom margin for the orientation, inches.	See above.
LinesPerPage	The number of lines per page to print. The font size is	An even number is
	chosen accordingly.	chosen with font size
		between 7 and 12
		points.
Header	String included in the top left of every page.	No header is shown.
Footer	String included in the bottom left of every page.	No footer is shown.
ShowLineCount	Indicate whether the line count should be shown to the left	False
	of each line.	
ShowPageCount	Indicate whether the page count should be shown in the	True
	center of the footer.	
DoubleSided	Indicate whether printing should be double-sided, currently	False
	not enabled. Use ShowDialog=True to select.	
OutputFile	Specify an output file, in cases where the printer name	Content is sent to
	corresponds to a file formatter (such as Adobe PDF).	printer device, not a
		file.
ShowDialog	Indicate whether to show the printer dialog, which allows	False
	review and editing of printer parameters. This is useful for	
	testing and for selecting advanced printer features not	
	handled in batch mode by this command.	
IfNotFound	Indicate action if the file is not found, one of:	Ignore
	• Ignore – ignore the missing file (do not warn).	
	• Warn – generate a warning (use this if the file truly is	
	expected and a missing file is a cause for concern).	
	• Fail – generate a failure (use this if the file truly is	
	expected and a missing file is a cause for concern).	

This command can be used to test print features. For example, use a printer that outputs to a PDF, XPS, or other format file rather than a physical printer. If ShowDialog=True and printing to a file is indicated by specifying the OutputFile parameter (such as with Adobe PDF), the **General** tab on the print dialog will be similar to the following:

🎒 Print		
<u>G</u> eneral p	age Setup Appearance	
Print Ser	vice	
Name:	Adobe PDF	Properties
Status:	Accepting jobs	
Туре:		
Info:		V Print To File
Print Ran	ge	Copies
	 All 	Number of copies:
	O Pages	Collate
		Print Cancel

Print Job Dialog General Properties

Note that **Print to File** is checked; however, the name of the file may not be displayed. Instead, the output file is specified by pressing **Print**, which displays the following dialog, with the initial choice matching the value of the OutputFile parameter:

Look in:	C Results		¥ 5		
My Recent Documents Desktop	.svn Test_Print Test_Print Test_Print Test_Print	TextFile_Landscape_Letter_1 TextFile_Portrait_8x11_1Pag TextFile_Portrait_8x11_3Pag TextFile_Portrait_8x11_OneF	IPage_PDF.TSTool.log le.TSTool.log le.TSTool.log Page.TSTool.log		
My Computer					
My Computer	File name:	xxx.pdf		OK	

Print To File Dialog

If running a command to write a PDF file, the following may be displayed:



Adobe PDF Error

Unfortunately, there does not seem to be any way to change the printer setting from the print dialog and consequently PDF cannot be used. An alternate approach such as iText may be implemented for PDF.

If a Microsoft XPS file is printed, the following software may need to be installed, in particular for Windows XP: http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=11816

Command Reference: ProcessTSProduct()

Process a time series product file to produce output

Version 10.16.00, 2013-01-28

The ProcessTSProduct() command automates creation of time series data products. Products are described in time series product description (*.*tsp*) files, which are typically created by using the **Save...Time Series Product** choice in graph windows (a future enhancements may allow creation of text products from summary or table views). See the **TSView Time Series Viewing Tools** appendix for more information about time series products. For example, the following sequence of actions can be used to define and use time series product description files:

- 1. Use TSTool and interactively select time series using the main window. The time series identifiers and/or aliases will be referenced in the time series product.
- 2. Interactively view a graph (e.g., *Results...Graph Line*) and edit its properties by right clicking on the graph and selecting the *Properties* choice (e.g., set titles and legend properties).
- 3. Save the graph as a time series product from the graph window using the **Save...Time Series Product** choice. Typically the product is saved in a location close to the command file. An example time series product file is as follows:

```
[Product]
ProductType = "Graph"
[SubProduct 1]
GraphType = "Line"
MainTitleString = "Streamflow (Monthly Total)"
[Data 1.1]
TSID = "08223000.DWR.Streamflow.Month~HydroBase"
[Data 1.2]
TSID = "08220500.DWR.Streamflow.Month~HydroBase"
```

- 4. Add a ProcessTSProduct () command to the original commands to allow the product to be created automatically. Select the time series product file created in the previous step.
- 5. Save the commands in a file (e.g., named *stations.TSTool*) so that they can be run again. The command file and time series product definition files must be used consistently (e.g., the time series identifiers and directory paths must be consistent).

If the product does not appear as intended, especially for complicated products, it may be necessary to edit the file and make the following corrections:

- Specify Color or other properties so that they are explicitly set and not defaulted.
- Verify that file paths in TSID properties are valid for the machine (may need to convert absolute paths to relative paths).

Time series identifiers in the product file are used as follows:

- If the time series are in TSTool's *Results* area, the time series will be used without rereading.
- Otherwise, the TSID is used to read the time series and must therefore contain enough information to locate and read the time series, such as the ~InputType~InputName information on at the end of the TSID.

If the TSAlias property is found in the product file, then the time series corresponding to the alias must be processed by a command file and be available in TSTool's **Results** area.

It is also possible to create a template time series product file and use the ExpandTemplateFile() command to automate creation of large numbers of graphs, for example to create images for a website.

The following dialog is used to edit the ProcessTSProduct () command and illustrates the command syntax. The path to the file can be absolute or relative to the working directory. The **Browse** button can be used to select the time series product description file (if a relative path is desired, delete the leading path after the select or use the **Remove Working Directory from TSP** button).

Edit ProcessTSProduct() Command				
Process a time series product definition file (typically named *.tsp) to create a graph product.				
Specify paths relative to the working directory, or use an absolute path.				
The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ProcessTSProduct				
TS product file (TSP):	Example_ProcessTSProduct.tsp	Bro	iwse	
Run mode:	×	Optional - when to process products (default=GUIAndBatch).		
View:	✓	Optional - display product in window (default=True).		
Output file:		Optional - output file with *.png or *.jpg extension.		
Default save file:		Optional - file to save during interactive editing (*.dv).		
Visible start:		Optional - start of (initial) visible period (default=all data visible).		
Visible end:		Optional - end of (initial) visible period (default=all data visible).		
	ProcessTSProduct(TSProductFile="Example_ProcessTSProduct.tsp")			
Command:				
Add Working Directory to TSP Cancel OK				
ProcessTSProduc				

ProcessTSProduct() Command Editor

The command syntax is as follows:

ProcessTSProduct(Parameter=Value,...)

Parameter	Description	Default
TSProductFile	The time series product file to process.	None – must be specified.
	The path to the file can be absolute or	
	relative to the working directory. The	
	Browse button can be used to select the	
	file to write (if a relative path is desired,	
	delete the leading path after the select).	
RunMode	Indicate the run mode to process the product, one of:	None – must be specified.
	• BatchOnly – indicates that the product should only be processed in batch mode.	
	• GUIOnly – indicates that the product should only be processed when the TSTool GUI is used (useful when Preview is set to Preview).	
	 GUIAndBatch – indicates that the product should be processed in batch and GUI mode. 	
View	Indicates whether the output should be	None – must be specified.
	previewed interactively, one of:	
	• True – display the graph.	
	• False – do not display the graph	
	(specify the output file instead to	
	automate image creation).	
OutputFile	The absolute or relative path to an output file. Use this parameter with	Graph file will not be created.
	View=False to automate image	
	processing. If the filename ends in	
	"jpg", a JPEG image file will be	
	produced. If the filename ends in "png",	
	a PNG file will be produced	
	(recommended).	
DefaultSaveFile	Used with experimental feature to	Editing is disabled.
	enabling editing in the time series table	
	that corresponds to a graph view.	
	Specify the default DateValue filename	
	to save edits.	
VisibleStart	The starting date/time to zoom for the	Full period is visible.
	initial (and image file) graph.	
VISIDIEE'nd	The ending date/time to zoom for the	Full period is visible.
	initial (and image file) graph.	

A sample command file to process a data product using State of Colorado HydroBase data is as follows:

```
# 08235350 - ALAMOSA RIVER ABOVE JASPER
08235350.USGS.Streamflow.Day~HydroBase
# 08236000 - ALAMOSA RIVER ABOVE TERRACE RESERVOIR
08236000.DWR.Streamflow.Day~HydroBase
# 7337 - SAGUACHE
7337.NOAA.Precip.Month~HydroBase
ProcessTSProduct(TSProductFile="Example_ProcessTSProduct.tsp")
```

After using the above dialog to edit the command, the time series product can be processed from TSTool as follows:

- 1. Interactively load and run the command file:
 - a. Open the command file, in this case containing the above commands file.
 - b. Process the commands using *Run All Commands*. The graph will be displayed for review.
- 2. Load and run the command file in one step:

Use the **Run...Process TSProduct** File menus to select and process the product file. The time series must be in the Results area or must be specified with enough information in the product file to read the time series.

3. Run TSTool in batch mode by specifying an output file (and optionally changing the RunMode parameter to BatchOnly) using:

tstool -commands commands.TSTool

The working directory will be set to the directory for the commands file and output will be relative to that directory.
Command Reference: ProfileCommands()

Profile the commands that have executed, to evaluate performance

Version 10.17.00, 2013-02-18

The ProfileCommands() command summarizes run times and memory use for each command in the command list, and outputs the information to detail (row for each command) and summary (one row for each command name) tables. This command is useful for evaluating which commands are slow or use more memory in a command workflow, so that software and command file logic improvements can occur. The command is usually placed at the end of a command file. The following apply to command profiling:

- Because the command is processed at the time it is encountered in the command list, the command itself and any subsequent commands are not included in the analysis. This generally is not an issue because the command will be used near the end of a workflow or at a strategic location where previous commands need to be examined.
- Currently the memory statistics are rough because the heap size is determined at the start and end of each command's execution and the Java runtime environment may allocate heap memory in blocks. In the future profiling data may be expanded to the estimated memory footprint of each command.
- There is a slight performance and memory hit to collect profiling information. In the future processor property commands may be implemented to control how much profiling data are collected (specifically if memory for each command object is estimated).
- If a command file is causing out of memory exceptions, then placing a ProfileCommands() command at the end of the command file likely will not be helpful. Instead, use a subset of the full command list so the ProfileCommands() command will be executed. Then evaluate the performance of the commands and determine if the command list logic can be optimized. If performance issues appear to be in the software itself, contact the developers to evaluate the software code. Also consider using the Free() and FreeTable() commands to free resources, especially if the results do not need to be available to users via the user interface.
- The runtime percent for each command is calculated as a percentage of the total runtime (ignoring the ProfileCommands() command and subsequent commands).
- The heap memory percentage delta for each command is calculated using the heap memory at the end of the command execution (not the heap memory at the end of the full run). Consequently, the delta reflects the memory use up to that point in time.
- Command profiling currently only applies to run mode. Commands are executed in discovery mode when a command file is loaded. For example, a subset of time series data is retrieved so that time series identifiers can be created and passed to following commands, which allows choices to be populated in command editors. Profiling discover mode is not support but should use a fraction of full runtime resources. For large command files (e.g., those generated by templates), it may be appropriate or necessary to load the commands without running discovery (see the -nodiscovery command line parameter and the *File...Open...Command File (no discovery)* menu item.
- Commands that generate many warning and failure messages will use more memory. Refer to the NumLogRecords column in the detail table to determine if this could be causing memory issues.
- The command currently does not allow sorting output tables by a column. This feature may be added in the future. Use the interactive table view to sort by column (this is how the tables were sorted for the figures below).

If loading or running commands are slow, the following actions might help:

- Use the Free() and FreeTable() commands to free resources. The command will still take up some resources because it has a place in the command list, but data resources used by the command will be freed.
- Review the profiling results to determine if certain commands are major resource users. Evaluate whether changes in the command logic can be implemented. Comment out blocks of commands (# commands will take fewer resources than /* */ blocks) and try to isolate problems. It may be necessary to run smaller subsets of commands, for example by splitting up lists of input time series.
- On Windows, use the Task Manager (run taskmgr) to review memory use by the javaw.exe program. If the memory use approaches the maximum, then the Java Runtime Environment likely will be spending time dealing with short memory and runtimes will increase until memory runs out. If necessary, change the –Xmx parameter in the *TSTool.14j.ini* file located in the system folder under the software install. This parameter indicates the maximum heap memory that can be used by the software. For a typical 32-bit Windows computer with at least 4GB of memory, the –Xmx parameter may be set to as high as 1700mb; however, a number that is too high may not be possible due to memory being used by other applications on the computer.

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

Edit ProfileCommands() Command									
This command profiles commands and saves execution time and memory information in detail and summary tables. If the software is running out of memory before completing the commands, then use this command on a subset of commands to identify memory usage. The profile information for this command and following commands will be incomplete.									
Summary table ID:	CommandProfileDetail	Optional - unique identifier for the summary table.							
Detail table ID:	CommandProfileSummary	Optional - unique identifier for the detail table.							
Command:	Command: ProfileCommands(SummaryTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="CommandProfileDetail",DetailTableID="Command								
Cancel OK									
	Cancel OK	ProfileoCom							

ProfileCommands() Command Editor

The following figure illustrates the output summary table. Because command execution may be very fast, times are shown in milliseconds (1-1000th of a second). The table can be output to a file with other commands.

Command	NumberOfOccurances	TotalTime (ms)	TotalTime (%)	AverageTime (ms)	MaximumTime (ms)	MinimumTime (ms)
ProcessTSProduct	705	88313	65.398	125	219	94
ŧ.	15525	11497	8.514	0	78	16
ReadTimeSeries	705	10375	7.683	14	140	47
ExpandTemplateFile	706	8167	6.048	11	.93	15
VewStatisticYearTS	2820	7542	5,585	2	79	0
CalculateTimeSeriesStatistic	3525	5583	4.134	1	157	0
ietProperty	2115	2097	1,553	0	31	0
ree	705	919	0.681	1	78	0
WriteTableToDelimitedFile	1	265	0.196	265	265	265
ReadTableFromDelimitedFile	1	172	0.127	172	172	172
StartLog	1	94	0.070	94	94	94
CopyTable	4	16	0.012	4	16	0
ProfileCommands	1	0	0.000	0	Q	0
WriteTableToHTML	2	0	0.000	0	0	0

ProfileCommands() Command Summary Output Table

The following figure illustrates the output detail table. Note that the heap memory is increased in blocks by the Java Runtime Environment so only large memory footprint commands trigger immediate heap memory increases.

0	TSTool - Table "CommandProfileDetail"												
.			StartTime		EndTime	RunTime	RunTime	StartHeap	EndHeap	DeltaHeap	DeltaHeap]
	Command	StartTime	(ms)	EndTime	(ms)	(ms)	(%)	(bytes)	(bytes)	(bytes)	(%)	NumLogRecords	
1	#	2013-02-18	1361205076983	2013-02-18	1361205076983	0	0.000	647929856	647929856	0	0.000	(ງົ 🔼
2	#	2013-02-18	1361205076983	2013-02-18	1361205076983	0	0.000	647929856	647929856	0	0.000	() 💻
3	StartLog	2013-02-18	1361205076983	2013-02-18	1361205077077	94	0.070	647929856	647929856	0	0.000	()
4	#	2013-02-18	1361205077077	2013-02-18	1361205077077	0	0.000	647929856	647929856	0	0.000	()
- 5	#	2013-02-18	1361205077077	2013-02-18	1361205077077	0	0.000	647929856	647929856	0	0.000	()
- 6	#	2013-02-18	1361205077077	2013-02-18	1361205077077	0	0.000	647929856	647929856	0	0.000	0)
7	#	2013-02-18	1361205077077	2013-02-18	1361205077077	0	0.000	647929856	647929856	0	0.000)
8	#	2013-02-18	1361205077077	2013-02-18	1361205077077	0	0.000	647929856	647929856	0	0.000	()
_ 9	#	2013-02-18	1361205077093	2013-02-18	1361205077093	0	0.000	647929856	647929856	0	0.000)
10	#	2013-02-18	1361205077093	2013-02-18	1361205077093	0	0.000	647929856	647929856	0	0.000	0)
11	ReadTableFromDel	2013-02-18	1361205077093	2013-02-18	1361205077265	172	0.127	647929856	647929856	0	0.000)
12	#	2013-02-18	1361205077280	2013-02-18	1361205077280	0	0.000	647929856	647929856	0	0.000	()
13	SetProperty	2013-02-18	1361205077280	2013-02-18	1361205077280	0	0.000	647929856	647929856	0	0.000	()
14	SetProperty	2013-02-18	1361205077280	2013-02-18	1361205077280	0	0.000	647929856	647929856	0	0.000	0)
15	SetProperty	2013-02-18	1361205077280	2013-02-18	1361205077280	0	0.000	647929856	647929856	0	0.000	()
16	#	2013-02-18	1361205077280	2013-02-18	1361205077280	0	0.000	647929856	647929856	0	0.000	0)
17	ReadTimeSeries	2013-02-18	1361205077280	2013-02-18	1361205077296	16	0.012	647929856	647929856	0	0.000	()
18	#	2013-02-18	1361205077296	2013-02-18	1361205077296	0	0.000	647929856	647929856	0	0.000	0)
19	NewStatisticYearTS	2013-02-18	1361205077296	2013-02-18	1361205077311	15	0.011	647929856	647929856	0	0.000	0) 🗸
Dis	playing 26816 rows, 1	3 columns.										Ready	
	ProfilkeCommands Detail												

ProfileCommands() Command Detail Output Table

The command syntax is as follows:

ProfileCommand(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
SummaryTableID	The identifier for the summary table.	Summary table will
		not be created.
DetailTableID	The identifier for the detail table.	Detail table will not be
		created.

Command Reference: ReadDateValue()

Read all time series from a DateValue File

Version 10.00.01, 2011-05-15

The ReadDateValue() command reads all the time series in a DateValue file into memory. See the **DateValue Input Type Appendix** for information about the file format.

The following dialog is used to edit the command and illustrates the command syntax. The path to the file can be absolute or relative to the working directory. DateValue files allow each time series to have an alias in addition to the time series identifier (TSID); however, the Alias parameter can be used to assign a new alias as the file is read.

👌 Edit ReadDateV	🕥 Edit ReadDateValue() Command 🛛 🛛 🔀							
Read all the time series	Read all the time series from a DateValue file, using information in the file to assign the identifier.							
Time series in DateValu	e files may also have an alias assigned; however, use the Alias parameter to assign a new alias.							
Specify a full path or re	lative path (relative to the working directory) for a DateValue file to read.							
The working directory is	s: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\ReadDateValue and the test of test							
Specifying the input pe	riod will limit data that are available for fill commands but can increase performance.							
DateValue file to read:	Data\08251500.DWR.Streamflow.IRREGULAR.dv Browse							
Alias to assign:	Insert: Select Specifier 💌 Optional - use %L for location, etc.							
Units to convert to:	Optional - request units different from input.							
Input start:	Optional - overrides the global input start.							
Input end:	Optional - overrides the global input end.							
	ReadDateValue(InputFile="Data\08251500.DWR.Streamflow.IRREGULAR.dv")							
Command:								
	ReadDate/all							

ReadDateValue() Command Editor

The command syntax is as follows:

ReadDateValue(Parameter=Value,...)

The following older command syntax is updated to the above syntax when a command file is read:

TS Alias = ReadDateValue(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
InputFile	The name of the DateValue input file to read, surrounded by double quotes to protect whitespace and special characters. Global property values can be used with the syntax \${PropertyName} (see also the SetProperty() command).	None – must be specified.
Alias	The alias to assign to the time series, as a literal string or using the special formatting characters listed by the command editor. The alias is a short identifier used by other commands to locate time series for processing, as an alternative to the time series identifier (TSID).	The alias in the file will be used if present.
NewUnits	Units to convert data to (must be in the <i>system/DATAUNIT</i> configuration file under the TSTool installation folder).	Use the data units from the file.
InputStart	Starting date/time to read data, in precision consistent with data.	Read all data.
InputEnd	Ending date/time to read data, in precision consistent with data.	Read all data.

A sample command file is as follows:

ReadDateValue(InputFile="Data\08251500.DWR.Streamflow.IRREGULAR.dv")

Command Reference: ReadDelimitedFile()

Read time series from a delimited file

The ReadDelimitedFile() command reads one or more time series from a column-oriented delimited file, where columns contain date/time and values. This command is useful for processing comma-separated-value (CSV) files exported from spreadsheets and mining data from the web(see also the WebGet() and FTPGet() commands). The command processes files that include the following types of information:

- 1. Comments in the header (before data) and embedded in data records (e.g., because bad data values were commented out).
- 2. Column headers as non-commented line at the top of the file.
- 3. Data records, in column format, containing date/time strings, data values, and other information.
- 4. Metadata, such as station identifiers, data types, units, and interval may be read from the file or specified with command parameters.

The mapping of data in the file to data in the time series occurs first by assigning column names, using one of the following methods:

- 1. Read column names from a line in the file, suitable when the column headings are simple strings and agree closely with the contents of the data columns.
- 2. Assign column names with command parameters. The file being read may include metadata within column headings and data records; however, the information can be difficult to extract because of formatting. For example, column headings may include the data type as "Precipitation\n(in)" (where \n indicates a newline). Consequently, the command supports assigning column names via command parameters in order to ensure robust data handling.

In any case, rather than trying to automatically determine other metadata like data type and units from the column heading, the values can be assigned with the DataType and Units parameters. Additional functionality may be added in the future automate metadata discovery. Examples of use for the two cases are shown in the examples below.

The command syntax is as follows:

ReadDelimitedFile(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
InputFile	The name of the delimited input file to read, surrounded by double quotes to protect whitespace and special characters. Global property values can be used with the syntax \${PropertyName} (see also the SetProperty() command).	None – must be specified.
Delimiter	The delimiter character(s) that separate columns.	None – must be specified.
Treat Consecutive Delimiters	Indicate whether consecutive delimiter characters should be treated as a single delimiter, for example, when multiple spaces are used to line up columns.	False (columns are separated by a single delimiter

AsOne		character)
Comment	Character(s) that if found at the start of lines in the file,	#
	indicate that the line is a comment. The characters are	
	interpreted individually (e.g., #\$ indicates that lines starting	
	with $\#$ or $\$$ will be treated as comments).	
SkipRows	Indicate absolute rows (1+) in the file to skip, using single	No rows will be
	numbers and ranges a-b, separated by commas. Rows are	skipped.
	skipped prior to other processing.	
SkipRowsAfter	Indicate the number of rows to skip after header comments.	No rows will be
Comments	Use this parameter to skip column headers prior to the data	skipped.
	lines. This parameter is typically not used if column names	
	are read from the file.	
ColumnNames	The user-specified names for columns in the file, used to	None – must be
	ensure that column headings in files are properly	specified.
	interpreted. These names are used in other parameters to	
	specify columns in the file. Separate column names with	
	commas. Column names can be specified as literal strings	
	or as FC[start:stop] to read columns from the file	
	header (assumed to be the first row after leading comments),	
	where start is 1+ and stop is blank to read all columns	
	or a negative number to indicate the offset from the end	
	column.	
DateTime	The column matching a value in ColumnNames, which	Required if
Column	indicates the date/time column in the file. Date and time are	DateColumn is
	in one column with no separating delimiter characters.	not specified.
DateTime	The format for date/time strings in the date/time column. If	Will automatically
Format	blank, common formats such as ISO YYYY-MM-DD	be determined by
	hh:mm and MM/DD/YYYY will automatically be detected.	examining date/time
	However, it may be necessary to specify the format to	strings.
	ensure proper parsing. This format will be used to parse	
	date/times from the DateTimeColumn or the merged	
	string from the DateColumn and TimeColumn (if	
	specified). The format string will depend on the formatter	
	type. Currently, only the "C" formatter is available, which	
	uses C programming language specifiers. The resulting	
	format includes the formatter and specifiers (e.g.,	
	C:%m%d%y).	
DateColumn	The column matching a string in ColumnNames, which	Required if
	indicates the date column in the file.	DateTimeColumn
		is not specified.
TimeColumn	The column matching a string in ColumnNames, which	A time column is
	indicates the time column in the file. Specify this parameter	required only when
	when DateColumn is specified and time is specified in a	DateColumn 18
	separate column. The DateColumn and TimeColumn	specified and the
	contents are merged with a joining colon character and are	time
	then treated as if DateTimeColumn had been specified.	ume.
ValueColumn	The column(s) matching a string in ColumnNames, which	None – must be
	indicate the data value columns. Separate column names	specified.
	with commas. The FC[start:stop] notation discussed	

	for ColumnNames can also be used.	
FlagColumn	The column(s) matching a string in ColumnNames, which indicate the data flag columns. Separate column names with commas. The FC[start:stop] notation discussed for ColumnNames can also be used. If specified, the number of columns must match the ValueColumn parameter, although blanks are allowed. Double-quotes around flags are not considered part of the flag.	Flags are not read.
Locationid	the value columns (or specify one value to apply to all columns). The FC[start:stop] notation discussed for ColumnNames can also be used.	specified.
Provider	The data provider identifier to assign to time series for each of the value columns (or specify one value to apply to all columns).	No provider will be assigned.
DataType	The data type to assign to time series for each of the value columns (or specify one value to apply to all columns).	Use the value column names for the data types.
Interval	The interval for the time series. Only one interval is recognized for all the time series in the file. Interval choices are provided when editing the command. If it is possible that the date/times are not evenly spaced, then use the IRREGULAR interval.	None – must be specified.
Scenario	The scenario to assign to time series for each of the value columns (or specify one value to apply to all columns).	No scenario will be assigned.
Units	The data units to assign to time series for each of the value columns (or specify one value to apply to all columns).	No units will be assigned.
Missing	Strings that indicate missing data in the file (e.g., "m").	Interpret empty column values as missing data.
Alias	The alias to assign to time series, as a literal string or using the special formatting characters listed by the command editor. The alias is a short identifier used by other commands to locate time series for processing.	No alias will be assigned.
InputStart	The date/time to start reading data.	All data or global input start.
InputEnd	The date/time to end reading data.	All data or global input end.

Example of Column Names Assigned with Command Parameter

The following example for two time series (gate height and discharge) illustrates a format where column headings are complex enough to require assignment of column names using a command parameter:

```
#...
#Data is returned in TAB delimited format. Data miners may find help on automating
#queries and formatting parameters at http://www.dwr.state.co.us/help
#
#Gaging Station: ALVA B. ADAMS TUNNEL AT EAST PORTAL NEAR ESTES PARK (ADATUNCO)
#Retrieved: 3/30/2010 03:04
```

#				
Station Date/Tim	ne GAG	GE_HT (ft)	DISCHRG	(cfs)
ADATUNCO	2006-10-01	00:00	2.34	225
ADATUNCO	2006-10-01	00:15	2.34	225
etc				

The following dialog is used to edit the command and illustrates the syntax for the command. The column headings are skipped because they are assigned with a command parameter. Because the delimiter is a tab, the space between date and time columns is NOT used as a delimiter and the date/time information is treated as one column.

♦ Edit ReadDelimitedFile() Command								
Read all the time series from a column-	riented delimited file, using provided information to assign the time series me	adata.						
Column names are defined by paramete	ers or are determined from the file, and are then used by other parameters to	read data.						
The column name(s), date/time column,	value column(s), and Location $\ensuremath{ID}(s)$ columns can use the notation $\ensuremath{FC}[\ensuremath{start}:s]$:op] to read column headings from the first non-comment file line.						
For example, "Date,FC[2:]" defines the	first column as "Date" and column names 2+ will be read from the file.							
If "FC[" does NOT appear in any param	eters, then a column heading line is NOT automatically read after comments.							
If used, specify input start and end to a	a precision appropriate for the data.							
Specify a full path or relative path (rela	ipecify a full path or relative path (relative to working directory) for a delimited file to read.							
ae working airectory is: c:upevalop is ioo_sourcebuild) is iool(test)regression(commands)general(ReadDelimitedHie								
Delimited hie to read:		Demined definition there there is for a set of the set						
Delimiter:	lt	Required - delimiter character (use (t for tab or (s for space).						
Treat consecutive delimiters as one?:		Optional (default=Faise).						
Comment character(s):		Optional - character(s) that indicate comment lines (default=#).						
Rows to skip (by row number):		Optional - comma-separated numbers (1+) and ranges (e.g., 1,3-/) (default=none).						
Rows to skip (after header comments):		Uptional - number or rows to skip after header comments (default=U).						
Column name(s):	ID,DateTime,GAGE_HT,DISCHRG	Required - column names for file, used below to read data (can use "FC[N:N]").						
Date/time column:	DateTime	Required - if date and time are in the same column (can use "FC[N:N]").						
Date/time format:	Select Specifier V =>	Optional - date/time format MM/DD/YYYY, etc. (default=auto-detect).						
Date column:		Required - if date and time are in separate columns (can use "FC[N:N]").						
Time column:		Required - if date and time are in separate columns (can use "FC[N:N]").						
Value column(s):	GAGE_HT,DISCHRG	Required - specify column names for time series values, separated by commas (can use "FC[N:N]").						
Flag column(s):		Optional - specify column names for time series flags, separated by commas (can use "FC[N:N]").						
Location ID(s):	ADATUNCO	Required - location ID for each value column, separated by commas (can use "FC[N:N]").						
Data provider:	DWR	Optional - data provider (data source) for the data (default=blank).						
Data type(s):	GAGE_HT,DISCHRG	Optional - data type for each value column, separated by commas (default=value column name(s)).						
Data interval:	15Minute	Required - data interval for time series.						
Scenario:		Optional - scenario for the time series (comma-separated, default=blank).						
Units of data:	ft,cfs	Optional - separate by commas (default=blank).						
Missing value(s):		Optional - missing value indicator(s) for file data (default=blank values).						
Alias to assign:	Select Specifier 💙 => %L%T	Optional - use %L for location, etc. (default=no alias).						
Input start:		Optional - overrides the global input start.						
Input end:		Optional - overrides the global input end.						
	ReadDelimitedFile(InputFile="Data\CO-DWR-ADATUNCO-tab.txt",Delimiter="\t",ColumnNames="ID,DateTime,GAGE_H T,DISCHRG",DateTimeColumn="DateTime",ValueColumn="GAGE_HT,DISCHRG",SkipRowsAfterComments="1",LocationID=" ADATUNCO",Provider="DWR",DataType="GAGE_HT,DISCHRG",Interval=15Minute,Units="ft,cfs",Alias="%L%T")							
Command:								
	Add Working Directory Cancel OK							
		BoodDolimitodFilo						

ReadDelimitedFile() Command Editor when Literally Specifying Column Names

The following example command file retrieves real-time time series data from the State of Colorado's website and reads the data:

```
WebGet(URI="http://www.dwr.state.co.us/SurfaceWater/data/export_tabular.aspx?
IDADATUNCO&MTYPEGAGE_HT,DISCHRG&INTERVAL1&START10/1/06&END10/6/06",
LocalFile="Data\CO-DWR-ADATUNCO-tab.txt")
ReadDelimitedFile(InputFile="Data\CO-DWR-ADATUNCO-tab.txt",
```

```
Delimiter="\t",ColumnNames="ID,DateTime,GAGE_HT,DISCHRG",
DateTimeColumn="DateTime",ValueColumn="GAGE_HT,DISCHRG",
SkipRowsAfterComments="1",LocationID="ADATUNCO",Provider="DWR",
DataType="GAGE_HT,DISCHRG",Interval=15Minute,Units="ft,cfs",Alias="%L%T")
```

Example of Column Names Read from the File

The following simple example of annual county population data illustrates a format that allows reading column names from the file. In this case, the rows and columns have been transposed from the original format to be compatible with this command and in the command example shown in the figure below the "County" heading is replaced with "Year" to more clearly indicate the contents.

```
County, COLORADO, Adams, Alamosa, Arapahoe, Archuleta, Baca, Bent, Boulder, Broomfield, Chaffee, ...
2000, 4338793, 366660, 15132, 491134, 10027, 4514, 5991, 296018, 0, 16294, 2229, 9386, ...
2001, 4456408, 360389, 15314, 502567, 10532, 4486, 5911, 282794, 41529, 16382, 2195, 9479, ...
...etc..
```

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

🔷 Edit ReadDelimitedFile() Co		X				
Read all the time series from a column-oriented delimited file, using provided information to assign the time series metadata.						
Column names are defined by paramete	Column names are defined by parameters or are determined from the file, and are then used by other parameters to read data.					
The column name(s), date/time column,	, value column(s), and Location ID(s) columns can use the notation FC[start:s	top] to read column headings from the first non-comment file line.				
For example, "Date,FC[2:]" defines the	First column as "Date" and column names 2+ will be read from the file.					
If includes NOT appear in any param	eters, then a column heading line is NOT automatically read after comments.					
Specify a full path or relative path (rela	tive to working directory) for a delimited file to read.					
The working directory is: C:\Develop\TS	5Tool_SourceBuild\TSTool\test\regression\commands\general\ReadDelimitedF	ïle				
Delimited file to read:	Data\DOLA-counties1yr-trans.csv	Browse				
Delimiter:	,	Required - delimiter character (use \t for tab or \s for space).				
Treat consecutive delimiters as one?:		Optional (default=False).				
Comment character(s):		Optional - character(s) that indicate comment lines (default=#).				
Rows to skip (by row number):		Optional - comma-separated numbers (1+) and ranges (e.g., 1,3-7) (default=none).				
Rows to skip (after header comments):		Optional - number of rows to skip after header comments (default=0).				
Column name(s):	Year,FC[2:]	Required - column names for file, used below to read data (can use "FC[N:N]").				
Date/time column:	Year	Required - if date and time are in the same column (can use "FC[N:N]").				
Date/time format:	Select Specifier V =>	Optional - date/time format MM/DD/YYYY, etc. (default=auto-detect).				
Date column:		Required - if date and time are in separate columns (can use "FC[N:N]").				
Time column:		Required - if date and time are in separate columns (can use "FC[N:N]").				
Value column(s):	FC[2:]	Required - specify column names for time series values, separated by commas (can use "FC[N:N]").				
Flag column(s):		Optional - specify column names for time series flags, separated by commas (can use "FC[N:N]").				
Location ID(s):	FC[2:]	Required - location ID for each value column, separated by commas (can use "FC[N:N]").				
Data provider:	DOLA	Optional - data provider (data source) for the data (default=blank).				
Data type(s):	Population	Optional - data type for each value column, separated by commas (default=value column name(s)).				
Data interval:	Year Y	Required - data interval for time series.				
Scenario:		Optional - scenario for the time series (comma-separated, default=blank).				
Units of data:	Persons	Optional - separate by commas (default=blank).				
Missing value(s):		Optional - missing value indicator(s) for file data (default=blank values).				
Alias to assign:	Select Specifier Y=> %L-pop	Optional - use %L for location, etc. (default=no alias).				
Input start:		Optional - overrides the global input start.				
Input enu:		opuonai - overnues une giobanniput enu.				
	ReadDelimitedFile(InputFile="Data\DOLA-counti	es1yr-trans.csv",Delimiter=",",ColumnNames="Year,FC[2:]",Dat				
	<pre>elimetolumn="Year", Valuetolumn="FC[2:]",Locat l=Vear Units="Persons" lias="%L-non")</pre>	ionip="rel2:j", Provider="pola", patatype="Population", Interva				
Command:	(rear, on recoond , arras on pop)					
	Add Working Directory	Cancel OK				
		ReadDelimitedFile2				

ReadDelimitedFile() Command Editor when Reading Column Names from the File

The following example command file retrieves population forecast data from the State of Colorado's website, transposes the rows and columns using a Python script, and reads the time series data. The Python script is not provided with this example but generates output as shown in the above data file example.

```
StartLog(LogFile="DOLA-county-pop.TSTool.log")
# This command file retrieves population data from the Colorado State Demographer
# website and processes the data into time series for use in analysis.
#
# First retrieve the data from the DOLA web site.
WebGet(URI="http://www.dola.state.co.us/dlg/demog/population/forecasts/countieslyr.csv",
 LocalFile="DOLA-countieslyr.csv")
#
# Transpose the rows/columns to match TSTool time series notation with dates in the
# first column.
SetProperty(PropertyName="ScriptDir",PropertyType=String,
RunPython(InputFile="${InstallDir}\python\table\transpose-csv.py",
 Arguments="\"${WorkingDir}\DOLA-counties1yr.csv\"
  \"${WorkingDir}\DOLA-countieslyr-trans.csv\"",Interpreter="Python")
#
# Read into time series from the delimited CSV file.
# Define column names dynamically based on the first non-comment line in the file
ReadDelimitedFile(InputFile="DOLA-countieslyr-trans.csv",Delimiter=",",
 ColumnNames="Year, FC[2:]", DateTimeColumn="Year", ValueColumn="FC[2:]",
 LocationID="FC[2:]", Provider="DOLA", DataType="Population", Interval=Year, Units="Persons",
 Alias="%L-pop")
```

Command Reference: ReadHecDss()

Read time series from a HEC-DSS File

Version1 0.00.01, 2011-05-15

The ReadHecDss() command reads time series from a HEC-DSS file. See the **HEC-DSS Input Type Appendix** for information about how time series properties are assigned using HEC-DSS file data. Current limitations for the command include:

- Irregular time series cannot be read.
- HEC-DSS uses times through 2400. However, TSTool will convert this to 0000 of the next day. Year, month, and day data are not impacted.

The following dialog is used to edit the command and illustrates the syntax for the command. In the future, it is envisioned that choices for A - F parts will be made available using data from the file.

🕚 Edit ReadHecDs	:() Command								
Read time series from a HEC-DSS file.									
Use * in the A, B, C, E, and F parts to filter the time series that are read (or leave blank to read all).									
The D part (start of period) is handled by specifying the input period.									
Or, instead of specifying	Or, instead of specifying parts, specify the DSS pathname to read a specific time series (the path will be used before the parts).								
The alias can be assigne	for time series based on time series properties,	for example use	e %L for location (A-part:B-part), %T for data type (C-part), %Z for s	cenario (F-part).					
Specify a full path or rela	tive path (relative to working directory) for a HE	C-DSS file to rea	ad.						
The working directory is:	C:\Develop\TSTool_SourceBuild\TSTool\test\reg	ression\comman	nds\general\ReadHecDss						
Specifying the input peri	od will limit data that are available for fill comman	ds but can incre	ase performance.						
HEC-DSS file to read:	Data\sample.dss			Browse					
A part (basin):			Optional - A part to match (default=match all).						
B part (location):			Optional - B part to match (default=match all).						
C part (parameter):			Optional - C part to match (default=match all).						
E part (interval):			Optional - E part to match (default=match all).						
F part (scenario):			Optional - F part to match (default=match all).						
DSS pathname:			Optional - DSS pathname to read (default=use parts from above).						
Input start:			Optional - overrides the global input start (default=read all).						
Input end:			Optional - overrides the global input end (default=read all).						
TSID location to assign:			Optional - use %A for A-part, etc. (default=%A:%B).						
Alias to assign:	Insert: Select Sp	ecifier 🔽	Optional - use %L for location, etc. (default=no alias).						
	ReadHecDss(InputFile="Data\sa	mple.dss")						
Command:									
	Add Work	king Directory	Cancel OK						
		ang biroccory							

ReadHecDss() Command Editor

The command syntax is as follows:

ReadHecDss(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
InputFile	The name of the HEC-DSS input file to read,	None – must be specified.
	surrounded by double quotes to protect whitespace and	
	special characters.	
A	The A part (basin name) to match, using * as a	Match all.
	wildcard. The location part of the TSTool time series	
	Identifier is set to A:B.	
в	The B part (location) to match, using * as a wildcard.	Match all.
	The location part of the 151001 time series identifier is	
C	Set to A B. The C part (perameter) to match using * as a wildcard	Match all
C	The C part (parameter) to match, using * as a windcard.	
म	The E part (interval) to match using * as a wildcard	Match all
<u>ц</u>	The E part (scenario) to match, using * as a wildcard.	Match all
Pathname	The HEC-DSS pathname for a time series as specified	Use the Λ_{-} E parameters
1 a crimanic	in the HEC-DSS documentation. Currently wildcards	Ose the A-r parameters.
	are not allowed. If specified this will be used instead	
	of the A-F parameters.	
InputStart	Starting date/time to read data, in precision consistent	Read all data.
	with data.	
InputEnd	Ending date/time to read data, in precision consistent	Read all data.
	with data.	
Location	The location to assign for the time series identifier. Use	Apart:Bpart (%A:%B).
	%A %F to indicate the Apart Fpart (D part is not	
	available). The assignment will impact the Alias	
	assignment. This is useful when only Bpart is desired	
	as the location identifier.	
Alias	Alias to assign to the output time series. See the	None is assigned.
	LegendFormat property described in the ISView	However, if the location
	Time Series viewing Tools appendix. For example,	contains periods that are in
	SL IS full location, ST is interval, and ST is scenario	identifier conventions, the
	DSS notation), %1 is interval, and %2 is scenario.	alias is set to the identifier
		with periods and the
		periods are replaced with
		spaces in the full time
		series identifier.

A sample command file is as follows:

```
ReadHecDss(InputFile="sample.dss",InputStart="1992-01-01",
InputEnd="1992-12-31",Alias="%L_%T_%Z")
```

Command Reference: ReadHydroBase()

Read time series from a HydroBase database

Version 10.12.00, 2012-09-28

The ReadHydroBase() command reads one or more time series from the HydroBase database (see the **HydroBase Datastore Appendix**). It is designed to utilize query criteria to process large numbers of time series, for example for a specific water district and data type.

The **Data type**, **Data interval**, and **Where** command parameters and input fields are similar to those from the main TSTool interface. However, whereas the main TSTool interface first requires a query to find the matching time series list and then an interactive select for specific time series identifiers, the ReadHydroBase() command reads the time series list and the corresponding data for the time series. This can greatly shorten command files and simplify command logic, especially when processing many time series.

The command supports the old-style input name selection (which corresponds to selecting HydroBase via the TSTool login dialog) and the new-style datastore convention (which corresponds to datastore configuration files). In the future, support for the input name may be phased out; however, this will require resolving how the HydroBase selection dialog is migrated to support datastores. Consequently, both approaches are currently supported during the transition.

Data for the location (station, structure, well, etc.) and time series metadata, as shown in the main TSTool interface, are set as time series properties. For example, the latdecdeg and longdecdeg values from the HydroBase vw_CDSS_StationMeasType view are available as time series properties of the same name. These properties can be transferred to a table with the CopyTimeSeriesPropertiesToTable() command and processed further with other table commands.

Time series corresponding to diversion records, which also include observations for reservoirs and wells, are handled as follows:

- 1. Daily diversion (DivTotal and DivClass) and reservoir release (RelTotal and RelClass) time series have their values automatically carried forward to fill data within irrigation years (November to October). HydroBase only stores full months of daily diversion record data when non-zero observations or non-zero filled values occur in a month. Therefore, this filling action should only provide additional zero values in an irrigation year where a diversion or release was recorded. Irrigation years with no observations remain as missing after the read.
- 2. Daily, monthly, and yearly diversion and reservoir release time series optionally can be filled by the ReadHydroBase() command using diversion comments, which indicate when irritation years should be treated as missing. See the FillUsingDivComments parameter below. Note that diversion comments should not conflict with more detailed records and provide additional information. The separate FillUsingDivComments() command also is available for filling but may be phased out in the future.
- 3. It also may be appropriate to use infrequent data types (IDivTotal, IDivClass, IRelTotal, and IRelClass) to supply data; however, because such values typically are annual values, additional decisions must be made for how to distribute the values to monthly and daily time series. These data, if available, are not automatically folded into the diversion records by TSTool.

4. See the FillHistMonthAverage(), FillPattern(), and other commands, which can be used to fill (estimate) values in data gaps after the initial time series are read.

The following dialog is used to edit the command and illustrates the syntax for the command. Two options are available for matching time series, based on historical software requirements. The following example illustrates how to read a single time series by specifying the time series identifier. This approach is essentially equivalent to using the ReadTimeSeries() command but offers HydroBase-specific parameters such as FillUsingDivComments, which are not available in the more general ReadTimeSeries() command.

👌 Edit Read	lHydroBase Command									
Read 1+ time series from a HydroBase database, using options from the parameter groups below.										
The datastore design is being phased in and the legacy input name parameter will be phased out in the future.										
Refer to the HydroBase documentation for information about data types.										
Specifying the p	ecifying the period will limit data that are available for later commands but can increase performance.									
Speciry databa	ase to read (datastore name will take precedence ir spec	ciried)								
Input name:	Optional - HydroBase co	nnection name (default=HydroBase if no datastore specified).								
Datastore:	Optional - HydroBase da	tastore to read (phasing in datastores).								
Data type:	DivTotal 💌	Required - data type for time series								
Data interval:	Month 💌	Required - data interval (time step) for time series.								
[Indicate how t	to match time series in HydroBase									
Match Single	Time Series Match 1+ Time Series									
Location:	2000812	For example, station ID or structure WDID.								
Data source:	DWR For example: USG5, DWR.									
TSID (full):	2000812.DWR.DivTotal.Month~HydroBase Created from above parameters.									
Alias to assign:	Select Specifier 💉 => %L-%T-%I	Optional - use %L for location, etc. (default=no alias).								
Input start:	1949-11	Optional - overrides the global input start.								
Input end:	2005-10	Optional - overrides the global input end.								
ESpecify how to	o handle diversion comments (only for diversion records)	·								
Fill using d	liversion comments: 💽 💌 Optional - whether to us	e diversion comments to fill more zero values (default=False).								
Fill using diver:	sion comments flag: Optional - string to flag I	filled diversion comment values.								
If missing:	~	Optional - how to handle missing time series (blank=Warn).								
	ReadHydroBase(TSID="2000812.DWR.D	ivTotal.Month~HydroBase",Alias="%L-%								
Command:	T-%I",InputStart="1949-11",InputE	nd="2005-10")								
	Cancel	ОК								

ReadHydroBase() Command Editor to Read a Single Time Series

The following figure illustrates how to query multiple time series.

🚯 Edit ReadHydroBase Command 🛛 🛛 🔀							
Read 1+ time series from a HydroBase database, using options from the parameter groups below.							
The datastore design is being phased in and the legacy input name parameter will be phased out in the future.							
Refer to the HydroBase documentation for information about data types.							
Specifying the period will limit data that are available for later commands but can increase performance.							
Specify database to read (datastore name will take precedence if specified)							
Input name: Optional - HydroBase connection name (default=HydroBase if no datastore specified).							
Datastore: Optional - HydroBase datastore to read (phasing in datastores).							
Data type: Stream - Streamflow Required - data type for time series							
Data interval: Month 💌 Required - data interval (time step) for time series.							
Indicate how to match time series in HydroBase							
Match Single Time Series Match 1+ Time Series							
Lice filters ("upbase" clauses) to limit you it size and increases performance . Filters are AND'ed							
Where County Name w Matches W OTEDO, CO							
Where: Councy Name V Matches V OTERO, CO V							
Where: Station Name Statis with Statis							
where:							
Alias to assign: Select Specifier 😪 => %L-%T-%I Optional - use %L for location, etc. (default=no alias).							
Input start: Optional - overrides the global input start.							
Input end: Optional - overrides the global input end.							
Specify how to handle diversion comments (only for diversion records)							
Fill using diversion comments: 🛛 🔄 Optional - whether to use diversion comments to fill more zero values (default=False).							
Fill using diversion comments flag: Optional - string to flag filled diversion comment values.							
If missing: Optional - how to handle missing time series (blank=Warn).							
ReadHydroBase(DataType="Streamflow",Interval="Month",Where1="County							
Command, Name; Matches; OTERO", Where2="Station Name; Starts							
with; TIMPAS", Alias="%L-%T-%I")							
Cancel							

ReadHydroBase() Command Editor to Read Multiple Time Series

The command syntax is as follows:

```
ReadHydroBase(Parameter=Value,...)
```

The following older command syntax is updated to the above syntax when a command file is read:

TS Alias = ReadHydroBase(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
InputName	The HydroBase database connection input name to use for the	Use the default
-	database connection. as initialized by the	HvdroBase
	OpenHydroBase() command or the HydroBase login dialog	connection.
	shown when TSTool starts. When using this approach the TSID will	
	end in ~HvdroBase~InputName. The input name approach for	
	specifying a HydroBase database connection may be phased out in	
	the future in favor of the datastore approach.	
DataStore	The HydroBase datastore name to use for the database connection, as	Use the default
	per datastore configuration files (see HvdroBase Datastore	(legacy
	appendix). When using this approach the TSID will end in	InputName)
	~DataStore. The datastore approach is being phased in as a more	HydroBase
	flexible design. Configuring a datastore with name HydroBase will	connection, if
	take precedence over InputName=HydroBase.	available.
DataType	The data type to be queried, as documented in the HydroBase	None – must be
	Datastore appendix.	specified.
Interval	The data interval for the time series, as documented in the	None – must be
	HydroBase Datastore appendix (e.g. Day, Month, Year),	specified.
	consistent with the DataType selection.	•
TSID	When reading a single time series, the time series identifier to read.	Use WhereN
	If specified, this parameter will override the WhereN parameters.	parameters to
		read multiple
		time series.
WhereN	When reading 1+ time series, the "where" clauses to be applied. The	If not specified,
	filters matche the values in the <i>Where</i> fields in the command editor	the query will
	dialog and the TSTool main interface. The parameters should be	not be limited
	named Where1, Where2, etc., with a gap resulting in the remaining	and very large
	items being ignored. The format of each value is:	numbers of time
		series may be
	"Item;Operator;Value"	queried.
	Where Item indicates a data field to be filtered on, Operator is	
	the type of constraint, and Value is the value to be checked when	
	querying.	
Alias	The alias to assign to the time series, as a literal string or using the	None – must be
	special formatting characters listed by the command editor. The alias	specified.
	is a short identifier used by other commands to locate time series for	
	processing, as an alternative to the time series identifier (TSID).	
InputStart	Start of the period to query, specified as a date/time with a precision	Read all
	that matches the requested data interval.	available data.
InputEnd	End of the period to query, specified as a date/time with a precision	Read all
	that matches the requested data interval.	available data.
FillUsing	Indicate whether to fill diversion and reservoir release time series	False
Divcomments	using diversion comments.	N. fl.
FillUsing	If specified as a single character, data flags will be enabled for the	INO IIag 18
Flag	une series and each filled value will be tagged with the specified	assigned.
riag	character. The flag can then be used later to label graphs, etc. The	
	I hag will be appended to existing hags if necessary.	1 1

Parameter	Description	Default
IfMissing	Indicate the action to be taken if the requested time series is missing, one of:	Warn
	• Ignore – ignore the time series (do not warn and the time series will not be available)	
	• Warn – generate a failure for the command	

A sample command file is as follows (read all reservoir releases to structure 0300905):

```
ReadHydroBase(DataType="DivClass",Interval="Day",
Where1="District;Equals;3",
Where2="Structure ID;Equals;905",Where3="SFUT;Contains;s:2")
```

This page is intentionally blank.

Command Reference: ReadMODSIM()

Read time series from a MODSIM output file

Version 10.00.00, 2011-03-28

The ReadMODSIM() command reads one or more time series from a MODSIM file. MODSIM is a node/link model used to simulate river basins (see the **MODSIM Input Type Appendix**). Specify a node/link name and data type to read a single time series – if not specified all time series from the file will be read. An alias can be assigned to each time series.

The following dialog is used to edit the command and illustrates the syntax. When a file is selected, the available data types are listed, based on the file extension (the types are not read from the file).

Read 1+ time series from a MODSIM format file. Specify the node/link name and data type to read one time series or leave blank to read all time series.								
Specify the node/link name and data type to read one time series or leave blank to read all time series.								
he node/link name must be consistent with information in the MODSIM file.								
Data types are determined from the file extension but others can be specified.								
Specify a full path or relative path (relative to working directory) for a MODSIM file to read.								
Specifying the period will limit data that are available for fill commands but can increase performance,								
If not specified, the period defaults to the query period.								
The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\Te	estCases\CommandReference\ReadMODSIM							
MODSIM file to read: Data\BIGTOM17.RES	Browse							
Node/link name to read: BIGTOM Optional - default is to read	d all.							
Data type to read: STOR_TRG Optional - default is to read	d all.							
TSID (full): BIGTOMSTOR_TRG Created from node/link nan	me and data type.							
Alias to assign: BIGTOM Insert: Select Specifier 👽 Optional - use %L for locat	tion, etc. (default=no alias).							
Input start: Optional - overrides the glo	obal input start.							
Input end: Optional - overrides the glo	obal input end,							
ReadMODSIM(Alias="BIGTON", InputFile="Data\BIGTOM	117.RES", TSID="BIGTOMST							
Command: OR_TRG")								
Add Working Directory Cancel OK								

ReadMODSIM() Command Editor

The command syntax is as follows:

ReadMODSIM(Parameter=Value,...)

The following older command syntax is updated to the above syntax when a command file is read:

TS Alias = ReadMODSIM(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
InputFile	The name of the MODSIM file to read,	None – must be specified.
	surrounded by double quotes. The path	
	to the file can be absolute or relative to	
	the working directory.	
TSID	A time series identifier pattern to filter	None – must be specified to
	the read – this is constructed in the editor	match a single time series.
	dialog from individual identifier parts –	
	the location and data type are specified	
	and used in the time series identifier.	
Alias	The alias to assign to the time series, as a	None – must be specified.
	literal string or using the special	
	formatting characters listed by the	
	command editor. The alias is a short	
	identifier used by other commands to	
	locate time series for processing, as an	
	alternative to the time series identifier	
	(TSID).	
InputStart	The start of the period to read data –	Use the global query period.
	specify if the period should be different	
	from the global query period.	
InputEnd	The end of the period to read data –	Use the global query period.
	specify if the period should be different	
	from the global query period.	

A sample command file is as follows:

ReadMODSIM(Alias="BIGTOM",	
<pre>InputFile="BIGTOM17.RES",TSID="GREELEYCBTSTOR_TRG")</pre>	

Command Reference: ReadNrcsAwdb()

Read 1+ time series from the NRCS AWDB web service

Version 10.13.00, 2012-11-07

The ReadNrcsAwdb() command reads one or more time series from the Natural Resources Conservation Service (NRCS) Air and Water Database (AWDB) web service (see the **NRCS AWDB Datastore Appendix**), including SNOTEL and Snow Course data.

The NRCS AWDB web service allows station lists to be filtered, both as a convenience and to ensure reasonable web service performance. Many of the choices that are available for limiting queries allow 0+ values to be provided. For example, specifying no requested element (data type) will return all available elements for a location. Specifying a list of elements (separated by commas) will return only stations and time series that have data for the requested elements.

The following dialog is used to edit the command and illustrates the syntax. Some choices are provided as a convenience. However, full listing of choices (such as all the thousands of HUCs) is not provided due to performance issues. Additional query features will be enabled as web service integration is enhanced.

👌 Edit ReadNrcsAwdb Command								
Read one or more time series from the National Resource Conservation Service (NRCS) Air and Water Database (AWDB) web service.								
WARNING - This command can be slow. Constrain the query to improve performance.								
Common choices are provided for convenience but may not apply (additional enhancements to web services may improve intelligent choices in the future).								
Refer to the NRLS AWDO backstore documentation for more information.								
If not specified, the input period defaults to the input period from SetInputPeriod().								
NRCS AWDB Documentation NRCS AWDB Online								
Data store: MrcsAwdb 💌	Required - data store containing data.							
Interval: Day 🔽	Required - data interval for data.							
Location constraints (specify one or more)								
Station ID(s):	List of 1+ station IDs separated by commas.							
State(s): CO	List of 1+ state abbreviations separated by commas.							
Networks(s): Select Network	List of 1+ network codes separated by commas (default=a							
HUC(s):	List of 1+ (8-digit) HUCs separated by commas.							
Bounding box:	Bounding box: WestLon,SouthLat,EastLon,NorthLat							
FIPS counties: Select County 💌 =>	List of 1+ counties separated by commas.							
Element(s): Select Element WTEQ	Optional - list of element codes separated by commas (default=all).							
Elevation, minimum:	Optional - minimum elevation, feet (default=all).							
Elevation, maximum:	Optional - maximum elevation, feet (default=all).							
Input start:	Optional - YYYY-MM-DD, override the global input start.							
Input end:	Optional - YYYY-MM-DD, override the global input end.							
Alias to assign: Select Specifier 💌 => %L-%T	Optional - use %L for location, etc. (default=no alias).							
ReadNrcsAwdb(DataStore="NrcsAwdb",Interval=Day,States="CO",Networks=	"SNTL",Elements="WTEQ",Alias="%L-%T")							
Command								
Cancel OK								
	ReadNrcsAwdb							

ReadNrcsAwdb() Command Editor

The command syntax is as follows:

ReadNrcsAwdb(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
DataStore	The NRCS AWDB datastore to use for queries.	None – must be specified.
Interval	The data interval ("duration" in NRCS AWDB terms) to query. The Irregular interval is used for instantaneous data.	None – must be specified.
Stations	A list of station identifiers to read, separated by commas.	Do not limit the query to a station list.
States	A list of state codes (e.g., AL), separated by commas.	Do not limit the query to a state list.
Networks	A list of data network codes (e.g., SNTL), separated by commas.	Do not limit the query to a network list.
HUCs	A list of 8-digit hydrologic unit codes, separated by commas.	Do not limit the query to a HUC list.
BoundingBox	A bounding box consisting of west longitude, south latitude, east longitude, and north latitude, separated by spaces. Longitudes in the western hemisphere are negative. This feature is not finalized, pending resolution of a web service issue.	Do not limit the query to a bounding box.
Counties	A list of county names, separated by commas. The state must be specified.	Do not limit the query to a county list.
Elements	Data element codes for the stations (e.g., WTEQ for snow water equivalent), separated by commas.	All available elements are returned.
ElevationMin	Minimum station elevation, feet.	Do not limit the query based on elevation minimum.
ElevationMax	Maximum station elevation, feet.	Do not limit the query based on elevation maximum.
InputStart	The start of the period to read data – specify if the period should be different from the global query period. Specify to the precision of the data using the format YYYY-MM-DD hh:mm.	Use the global query period.
InputEnd	The end of the period to read data – specify if the period should be different from the global query period. Specify to the precision of the data using the format YYYY-MM-DD hh:mm.	Use the global query period.
Alias	The alias to assign to the time series, as a literal string or using the special formatting characters listed by the command editor. The alias is a short identifier used by other commands to locate time series for processing, as an alternative to the time series identifier (TSID).	None – must be specified.

Command Reference: ReadPatternFile()

Read the pattern file to be used with FillPattern() commands

Version 08.16.04, 2008-09-19

The ReadPatternFile() command reads pattern time series to be used with FillPattern() commands (see the FillPattern() command for more information). The patterns indicate whether a month is wet, dry, or average, although any number of characterizations can be used. One or more patterns can be included in each pattern file, similar to StateMod time series files (see the **StateMod Input Type** appendix), and multiple pattern files can be used, if appropriate. The following example illustrates the file format. See also the AnalyzePattern() command, which can be used to generate the file.

# # #	Years Shown = W Missing monthly Time series ide Description	Water Years y data fill entifier	B Led by th = =	ne Mixed 09034500 COLORADO	Station).CRDSS_U) RIVER A	Method, JSGS.QME AT HOT SU	USGS 19 MONTH.1 JLPHUR SI	89 PRINGS, (20.					
#	-e-bel	beb	eb	eb	eb	eb	eb	eb	eb	eb	eb-	eb	eb	е
	10/1908 -	9/1996 AC	CFT WYR											
19	09 09034500	AVG	AVG	AVG	WET	WET	AVG	AVG	AVG	WET	WET	WET	WET	
19	10 09034500	WET	WET	WET	WET	WET	WET	AVG	AVG	AVG	AVG	AVG	AVG	
19	11 09034500	AVG	AVG	WET	AVG	AVG	AVG	AVG	WET	WET	WET	AVG	WET	
19	12 09034500	WET	WET	WET	WET	WET	AVG	AVG	WET	WET	WET	WET	WET	
	.ommitted													

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

Read the pattern file used with FillPattern() commands. It is recommended that the location of the files be specified using a path relative to the working directory. The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\ReadPatternFile Pattern file: fill.pat Browse	Edit ReadPatternFile() Command
tt is recommended that the location of the files be specified using a path relative to the working directory. The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\ReadPatternFile Pattern file: fill.pat ReadPatternFile(PatternFile="fill.pat") Command: Add Working Directory To File Cancel OK	Read the pattern file used with FillPattern() commands.
The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\ReadPatternFile Pattern file: fill.pat ReadPatternFile(PatternFile="fill.pat") Command: Add Working Directory To File Cancel OK	It is recommended that the location of the files be specified using a path relative to the working directory.
Pattern file: fill.pat Command: ReadPatternFile(PatternFile="fill.pat") Add Working Directory To File Cancel	$The working \ directory \ is: \ C: \ Develop \ Source Build \ STool \ test \ vegression \ User \ Manual \ Examples \ Test \ Cases \ Command \ Reference \ Read \ Pattern \ File \ Source \ Sou$
ReadPatternFile(PatternFile="fill.pat") Command: Add Working Directory To File OK	Pattern file: fill.pat Browse
Add Working Directory To File Cancel OK	Command:
	Add Working Directory To File Cancel OK

ReadPatternFile() Command Editor

The command syntax is as follows:

ReadPatternFile(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
PatternFile	The path to the pattern file, which can be absolute or relative to the working directory.	None – must be specified.

A sample command file is as follows:

ReadPatternFile(PatternFile="fill.pat")

This page is intentionally blank.

Command Reference: ReadPropertiesFromFile()

Read one or more time series processor properties from a file

Version 10.12.00, 2012-08-02

The ReadPropertiesToFile() command reads the values of one or more time series processor properties from a file. The corresponding WritePropertiesToFile() command can be used to write properties to a file. Processor properties include global defaults such as InputStart, InputEnd, OutputStart, OutputEnd, OutputYearType, WorkingDir, and also user-defined properties set with the SetProperty() command. Internally, properties have a name and a value, which is of a certain type (string, integer, date/time, etc.). Examples of using the command include:

- creating tests to verify that properties are being set
- passing information from another program, such as a Python script, to TSTool
- reading persistent information from a previous use, such as the date/time that data were last downloaded from a web service

A number of property formats are supported as listed in the following table.

Format	Description		
NameValue	Simple format, all properties handled as text:		
	PropertyName=PropertyValue		
	PropertyName="Property value, quoted if necessary"		
NameTypeValue	Same as NameValue format, with non-primitive objects treated as simple		
	constructors:		
	PropertyName=PropertyValue		
	DateTimeProperty=DateTime("2010-10-01 12:30")		
NameTypeValue	Similar to the NameTypeValue format, however, objects are represented		
Python	using "Pythonic" notation, to allow the file to be used directly by Python		
	scripts:		
	PropertyName="PropertyValue"		
	DateTimeProperty=DateTime(2010,10,1,12,30)		

Property File Formats

The format of the file currently is not required when reading the file because the command detects the format for each property and creates an appropriate object type. If this becomes an issue, the command may be enhanced to add a parameter to specify the format (similar to the WritePropertiesToFile() command).

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

💧 Edit ReadProp	pertiesFromFile() Command			
Read one or more properties from a file. The properties will apply globally to subsequent commands. It is recommended that the input file be relative to the current working directory.				
The working directory	is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ReadPropertiesFromFile			
Property file to read:	LastRunPeriod.txt Browse			
Property to read:	Optional - properties to read, separated by commas (default=read all).			
	ReadPropertiesFromFile()			
Command:				
Add Working Directory Cancel OK				
	ReadPropertiesFrom			

ReadPropertiesFromFile() Command Editor

The command syntax is as follows:

```
ReadPropertiesFromFile(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
InputFile	The property file to read, as an absolute path or	None – must be specified.
	relative to the command file.	_
IncludeProperty	The names of properties to read, separated by	If not specified, all
	commas.	processor properties will
		be written.

Command Reference: ReadRccAcis()

Read time series from the RCC ACIS web services

Version 10.11.00, 2012-06-28

The ReadRccAcis() command reads one or more time series from the Regional Climate Center (RCC) Applied Climate Information System (ACIS) web services, in particular to provide access to daily historical and real-time values from the National Climatic Data Center (NCDC). Features and limitations of ACIS are described in the **RCC ACIS Data Store** appendix. Because web services are used to access a remote database, there may be some delay in retrieving data. For data intensive processes, it may be advisable to mine the data, save to a local file or database, and then perform additional processing using the local data.

The following dialog is used to edit the command and illustrates the syntax for the command when reading a single time series. This is appropriate when a specific site is being processed.

S Edit ReadRccAcis Command				
Read one or more time series from the RCC ACIS web service.				
WARNING - This command can be slow. It is recommended that the Where filters be used t	p limit queries when reading multiple time series.			
Refer to the RCC ACIS Data Store documentation for more information.				
If not specified, the input period defaults to the input period from SetInputPeriod().				
Data store: RCC-ACIS-Test 💙	Required - data store containing data.			
Data type: pcpn - Precipitation (daily)	Required - data type for time series.			
Data interval: Day 💌	Required - data interval (time step) for time series.			
Indicate how to match time series in ACIS				
Match Single Time Series Match 1+ Time Series Using Filter				
Specify a site ID when a specific time series is being processed. Site ID: COOP:054719 Required - site type (optional) and identifier (e.g., COOP:052454).				
Input start: 1911-01-01	Optional - YYYY-MM-DD, override the global input start.			
Input end: 1912-03-15	Optional - YYYY-MM-DD, override the global input end.			
Alias to assign: Select Specifier 💌 => %L-Precipitation	Optional - use %L for location, etc. (default=no alias).			
Command: ReadRccAcis(DataStore="RCC-ACIS-Test",DataType="pcpn",Interval="Day",SiteID="COOP:0547				
Cancel OK	PoodPooAcie Sinal			

ReadRccAcis() Command Editor for Reading Single Time Series

The following dialog is used to edit the command and illustrates the syntax for the command when reading multiple time series. This is appropriate when performing bulk processing. Mouse over the *Where* data entry fields to see information about choices.

👌 Edit Read	RccAcis Command				\mathbf{X}
Read one or mor	Read one or more time series from the RCC ACIS web service.				
WARNING - Th	is command can be slow. It is	recommended th	hat the When	e filters be used	to limit queries when reading multiple time series.
Refer to the RCC	C ACIS Data Store documentation f	or more information			
If not specified, I	the input period defaults to the inp	ut period from SetIr	nputPeriod().		
Data store:	RCC-ACIS-Test 💙				Required - data store containing data.
Data type:	pcpn - Precipitation (daily)	*			Required - data type for time series.
Data interval:	Day 💙				Required - data interval (time step) for time series.
[Indicate how to	o match time series in ACIS				
Match Single 1	Time Series Match 1+ Time Series	Using Filter			
	Specify filters when	multiple time series	are being proc	essed (a location c	onstraint must be specified).
	Where: Bounding B	ox 🔽	Matches	3332,38.000001	
	Where:	~	Matches	1	Optional - query filters.
	Where:	*	Matches	/	
Input start:	1911-01-01				Optional - YYYY-MM-DD, override the global input start.
Input end:	1912-03-15				Optional - YYYY-MM-DD, override the global input end.
Alias to assign:	Select Specifier 🛛 💙 => %L	-Precipitation			Optional - use %L for location, etc. (default=no alias).
Command: ReadRccAcis(DataStore="RCC-ACIS-Test",DataType="pcpn",Interval="Day",Where1="Bounding Box;Matches;-103.53334,37.999999,-103.53332,38.000001",InputStart="1911-01-01",InputEn d="1912-03-15",Alias="%L-Precipitation")					
Cancel OK					
					ReadRccAcis

ReadRccAcis() Command Editor for Reading Multiple Time Series

The command syntax is as follows:

ReadRccAcis(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
DataStore	The name of the RCC ACIS data store from which to read.	None – must be
		specified.
DataType	The data type to be queried, as documented in the RCC ACIS Data	None – must be
	Store appendix. For example, use pcpn to request precipitation data	specified.
	(for the older version 1 ACIS, the "Variable Major" number is used,	
	for example 4 for precipitation).	
Interval	The data interval for the time series. Currently only daily time series	None – must be
	can be requested.	specified.
SiteID	Used when reading a single time series. The site ID should be	If not specified,
	specified using the station type and site identifier (e.g.,	the WhereN
	COOP:052454). The station type can be determined by first	filters are used.
	querying the time series using the TSTool main interface or using the	
	WhereN parameter and reviewing the resulting time series identifiers	
	in returned time series. Omitting the station type will assume the	

Parameter	Description	Default
	ACIS identifier, which is internal to the ACIS system and not	
	typically used by users. Specifying the SiteID will override the	
	WhereN parameter.	
WhereN	Used when reading 1+ time series. The "where" clauses to be applied to filter the list of stations, matching the values in the Where fields in the command editor dialog and the TSTool main interface. The parameters should be named Where1, Where2, etc., and a gap in numbering will result in the remaining items being ignored. The format of each value is: "Item;Operator;Value"	If not specified, the query will not be limited and very large numbers of time series may be queried.
	Where Item indicates a data field to be filtered on, Operator is	
	the type of constraint, and Value is the value to be checked when querying.	
InputStart	Start of the period to query, specified as a date/time with a precision	Read all
	that matches the requested data interval.	available data.
InputEnd	End of the period to query, specified as a date/time with a precision	Read all
	that matches the requested data interval.	available data.
Alias	The alias to assign to the time series, as a literal string or using the	None – must be
	special formatting characters listed by the command editor. The alias	specified.
	is a short identifier used by other commands to locate time series for	
	processing, as an alternative to the time series identifier (TSID).	

This page is intentionally blank.

Command Reference: ReadReclamationHDB()

Read time series from a Reclamation HDB database

Version 10.20.00, 2013-04-21

The ReadReclamationHDB() command reads one or more time series from a Reclamation HDB database:

- a single time series (which can be part of an ensemble), indicated by the individual time series identifier:
 - o a "real" time series (observations)
 - o a "model" time series (output from a model)
- all time series in an ensemble, indicated by the ensemble identifier:
 - ensemble trace time series are stored as "model" time series individual ensemble trace time series can be queried by specifying the appropriate "hydrologic indicator" (which is set to the ensemble time series sequence number from TSTool time series)

See the WriteReclamationHDB() command documentation for information about writing the time series that are read by this command. See the **Reclamation HDB Data Store Appendix** for more information about the database features and limitations.

When reading a single time series or ensemble, the choices presented to the user cascade to allow only valid choices.

The following dialog is used to edit the command and illustrates the syntax of the command when reading "real" or "model" data using filters. This approach can be used when reading one or more time series in bulk. *Where* criteria should be specified in sequential order without intervening blank specifiers.

S Edit ReadReclamationHDB Command			
Read one or more time series, or an ensemble, from a Reclamation HDB database.			
Constrain the query by specifying time series metadata to match.			
Specify date/times using the format YYYY-MM-DD hh:mm:ss, to a precision appropriate for the data interval (default=input period from SetInputP	eriod()).		
Datastore: ReclamationHDB-Dev 🗸	Required - datastore containing data.		
Data interval: Day	Required - data interval (time step) for time series.		
Specify how to match HDB time series or ensemble			
Read 1+ time series using filter Read single time series or ensemble			
Use these parameters when reading 1+ time series from HDB.			
Data type: stream gage - flow	Required - data type for time series.		
Where: Site - Common Name V Matches V AAA_DELETE			
Where: Matches			
Where: V Matches V	Onking al any with the second		
Where: V Matches V	Optional - query nicers.		
Where: V Matches V			
Where: Matches V			
Input start:	Optional - override the global input start.		
Input end:	Optional - override the global input end.		
Alias to assign: Select Specifier 💌 => %L-%T	Optional - use %L for location, etc. (default=no alias).		
ReadReclamationHDB(DataStore="ReclamationHDB-Dev", Interval="Day", DataType="stream gage - flow", Where1="Site - Command Common Name; Matches; AAA_DELETE", SiteCommonName="AAA_DELETE", DataTypeCommonName="current air temp", EnsembleTraceID="null", Alias="%L-%T")			
Cancel			

ReadReclamationHDB() Command Editor When Using Filters to Read 1+ Time Series

The following figure illustrates reading a single "real" time series (note that the model parameters are not specified).

Sedit ReadReclamationHDB Command		X
Read one or more time series, or an ensemble, from a Reclamation HDB database.		
Constrain the query by specifying time series metadata to match.		
Specify date/times using the format YYYY-MM-DD hh:mm:ss, to a precision appropriate fo	r the data interval (default=input period from SetInputPeriod()).	
Datastore: ReclamationHDB-Dev 💌	Required - datastore co	ntaining data.
Data interval: Day 💌	Required - data interva	l (time step) for time series.
Specify how to match HDB time series or ensemble		
Read 1+ time series using filter Read single time series or ensemble		
Specify how to match the HDB site_datatype_id-		
Site common name: AAA_DELETE	Required - used with data type common name to determine site_datatype_id.	
Data type common name: current air temp 💌	Required - used with site common name to determine site_datatype_id.	
Matching site_id: 100072 (9 matches)	Information - useful when comparing to database contents.	
Matching site_datatype_id: 101355	Information - useful when comparing to database contents.	
Site data type ID: 🔽 👻	Optional - alternative to selecting above choices.	
Specify how to match HDB model_run_id for single model time series or ensemble of mo	odel time series	
Single model time series Ensemble of model time series		
Use these parameters when reading an individual mo	del time series from HDB.	
Model name:	Required - used to determine the model_run_id.	
Model run name: 🔽	Required - used to determine the model_run_id.	
Model run date: 🔽	Required - YYYY-MM-DD hh:mm, used to determine the model_run_id.	
Hydrologic indicator: 🔽	Required - used to determine the model_run_id.	
Selected model_id: No matches	Information - useful when comparing to database contents.	
Selected model_run_id: No matches	Information - useful when comparing to database contents.	
Model run ID:	Optional - alternative to selecting above choices.	
Input start:	Optional - override the	global input start.
Input end:	Optional - override the	global input end.
Alias to assign: Select Specifier V => %L-%T	Optional - use %L for lo	cation, etc. (default=no alias).
ReadReclamationHDB(DataStore="ReclamationHDE	8-Dev",Interval="Day",DataType="stream gage - flow	J",Where1="Site -
Command: Common Name; Matches; AAA_DELETE", SiteCommonNa	ame="AAA_DELETE",DataTypeCommonName="current air	
temp",EnsembleTraceID="null",Alias="%L-%T")		
	Cancel OK	
		ReadReclamationHDB Rea

ReadReclamationHDB() Command Editor to Read a Single Real Time Series

The following figure illustrates reading a single "model" time series, in which case model parameters are specified in addition to the site and data type parameters.

Sedit ReadReclamationHDB Command				
Read one or more time series, or an ensemble, from a Reclamation HDB database.				
Constrain the query by specifying time series metadata to match.				
Specify date/times using the format YYYY-MM-DD hh:mm:ss, to a precision appropriate for the data interval (default=input period from SetInputPeriod()).				
Datastore: ReclamationHDB-Dev 🔽 Required - datastore co	ntaining data.			
Data interval: Day 🔹 Required - data interval	(time step) for time series.			
Specify how to match HDB time series or ensemble				
Read 1+ time series using filter Read single time series or ensemble				
Specify how to match the HDB site_datatype_id				
Site common name: ABVFLTCO Required - used with data type common name to determine site_datatype_id.				
Data type common name: Total Delivery Volume 👻 Required - used with site common name to determine site_datatype_id.				
Matching site_jd: 100210 (3 matches) Information - useful when comparing to database contents.				
Matching site_datatype_id: 100764 Information - useful when comparing to database contents.				
Site data type ID: Optional - alternative to selecting above choices.				
Specify how to match HDB model_run_id for single model time series or ensemble of model time series				
Single model time series Ensemble of model time series				
Use these parameters when reading an individual model time series from HDB.				
Model name: NWS CBRFC Forecast 🛛 💌 Required - used to determine the model_run_id.				
Model run name: CBRFC ESP Forecast Loader 💙 Required - used to determine the model_run_id.				
Model run date: 2013-01-07 00:00 💌 Required - YYYY-MM-DD hh:mm, used to determine the model_run_id.				
Hydrologic indicator: 1975 💌 Required - used to determine the model_run_id.				
Selected model_id: 5 Information - useful when comparing to database contents.				
Selected model_run_id: No matches Information - useful when comparing to database contents.				
Model run ID: V Optional - alternative to selecting above choices.				
Input start: Optional - override the g	global input start.			
Input end: Optional - override the g	global input end.			
Alias to assign: Select Specifier 🔽 => %L-%T Optional - use %L for lo	cation, etc. (default≕no alias).			
ReadReclamationHDB(DataStore="ReclamationHDB-Dev",Interval="Day",DataType="stream gage - flow	",Where1="Site -			
Common Name; Matches; AAA_DELETE", SiteCommonName="ABVFLTCO", DataTypeCommonName="Total Delivery				
Volume", ModelName="NWS CBRFC Forecast", ModelRunName="CBRFC ESP Forecast Loader", ModelRunDate=	2013-01-07			
00:00", HydrologicIndicator="1975", Alias="%L-%T")				
Cancel OK				
Re	adReclamationHDB Model			

ReadReclamationHDB() Command Editor to Read a Single Model Time Series
The following figure illustrates reading n ensemble of "model" time series, in which case ensemble/model parameters are specified in addition to the site and data type parameters.

👌 Edit Read	dReclamationHDB (Command			X
Read one or mo	ore time series, or an er	nsemble, from a Reclamation HDB data	base.		
Constrain the q	juery by specifying time	series metadata to match.			
Specify date/tin	mes using the format YY	/YY-MM-DD hh:mm:ss, to a precision ap	opropriate for I	the data interval (default=input period from SetInputPeriod()).	
Datastore:	ReclamationHDB-Dev	*		Required - datastore c	ontaining data.
Data interval:	Day 🖌			Required - data interva	al (time step) for time series.
F ^{Specify} how to	o match HDB time series	s or ensemble			
Read 1+ time	e series using filter Re	ad single time series or ensemble			
Specify how	I to match the HDB site	_datatype_id			
Site	common name: AAA_C	DELETE	~	Required - used with data type common name to determine site_datatype_id.	
Data type	common name: curren	nt air temp 🔽		Required - used with site common name to determine site_datatype_id.	
м	latching site_id: 100072	2 (9 matches)		Information - useful when comparing to database contents.	
Matching site	e_datatype_id: 101355	5		Information - useful when comparing to database contents.	
Site	e data type ID:	▼		Optional - alternative to selecting above choices.	
Single mode	to match HDB model_ru	un_id for single model time series or en le of model time series	nsemble of mod	del time series	
Use these p	arameters when readin	ng an ensemble of model time series fro	om HDB. If the	e run date is specified, the ensemble time series will be uniquely identified with th	ne run date (to the minute).
	Ensemble name:	Test 🔽	Requi	ired - used to determine the ensemble model_run_id.	
E	insemble model name:	CBT AOP RiverWare	🗾 🔽 Requi	ired - used to determine the ensemble model_run_id.	
Ense	emble model run date:	v	Optio	nal - YYYY-MM-DD hh:mm, used to determine the ensemble model_run_id (defa	ult=run date not used).
<u> </u>	Selected ensemble_id: N	No matches	Infor	mation - useful when comparing to database contents.	
Selecte	ed ensemble model_id: 1	10	Infor	mation - useful when comparing to database contents.	
Selected en	isemble model_run_id: [Determined for trace when command is	run		
]
Input start:				Optional - override the	global input start.
Input end:				Optional - override the	global input end.
Alias to assign:	Select Specifier	✓ => %L-%T		Optional - use %L for l	ocation, etc. (default=no alias).
	ReadReclamati	.onHDB(DataStore="Reclam	ationHDB-	-Dev", Interval="Dav", DataTvpe="stream gage - flo	w",Where1="Site -
Common de	Common Name;Ma	atches; AAA DELETE", Site	CommonNan	ne="AAA DELETE",DataTypeCommonName="current air	
Command:	temp",ModelNa	me="CBT AOP RiverWare",	ModelRunN	Name="CBT AOP Test",ModelRunDate="2012-07-01	
	00:00:00.0",E	nsembleName="Test",Ense	mbleModel	lName="CBT AOP RiverWare",Alias="%L-%T")	
			C	Cancel OK	
				Read	ReclamationHDB Ensemble

ReadReclamationHDB() Command Editor to Read an Ensemble of Model Time Series

The command syntax is as follows:

ReadReclamationHDB(Parameter=Value,...)

Parameter	Description	Default			
DataStore	Reclamation HDB data store name indicating	None – must be			
	database from which to read time series.	specified.			
Interval	The data interval to read (Hour, Day, Month,	None – must be			
	Year, Irregular). Irregular is used for	specified.			
	instantaneous data and internally results in data				
	with date/times to minute precision. 2Hour, 3Hour,				
	4Hour, 6Hour, 12Hour, and 24Hour can also be				
	included, but how can HDB be queried to limit				
	choices to these intervals? This interval is				
	important because it tells TSTool how to allocate				

Parameter	Description	Default
	memory for data values, and iterate through data.	
	Use the following parameter when reading 1+ time	
	series using filters	
DataType	The data type to read as ObjectType -	None – must be
	DataTypeCommonName. The object type is	specified.
	shown to help with selections. * can be specified	
	to read all data types.	
WhereN	The "where" clauses to be applied when querying	If not specified, the
	data, which match the values in the Where fields in	query will not be
	the TSTool main interface. The parameters should	limited and very large
	be specified as Where1, Where2, etc., with no	numbers of time series
	intervening gaps in numbering. All clauses are	may result from the
	joined as "and" and are therefore cumulative in	query (which may
	limiting the query. The format of each parameter	require a long time to
	value is:	perform the query).
	"Itom: Operator: Walue"	
	item/operator/varue	
	Where Item indicates a data field to be filtered on,	
	Operator is the type of constraint, and Value is	
	the value to be checked when querying.	
	Use the following parameters when reading a single	
	time series or an ensemble of time series.	
Site	The site common name for the time series location;	None – must be
CommonName	used with the data type common name to determine	specified unless
	the site_datatype_id in the database.	SiteDataTypeID is
		specified.
DataType	The data type common name for the time series;	None – must be
CommonName	used with the site common name to determine the	specified unless
	site_datatype_id in the database.	SiteDataTypeID is
		specified.
SiteDataTypeID	The site_datatype_id value to match the time series.	
	If specified, the value will be used instead of the	
	site_datatype_id determined from	
	SiteCommonName and	
	DataTypeCommonName.	
	Use the following parameters when reading a single model time series	
ModelName	The model name for the time series: used with the	Nona must ha
Moderivanie	model run name, hydrologic indicator(s), and model	specified unless
	run date to determine the model run id in the	Model Run ID is
	database.	specified.
ModelRunName	The model run name for the time series: used with	None – must be
_	the model name, hydrologic indicator(s), and model	specified unless
	run date to determine the model_run_id in the	ModelRunID is
	database.	specified.
ModelRunDate	The model run date (timestamp) to use for the time	None – must be
	series; used with the model name, model run name,	specified unless

Parameter	Description	Default
	and hydrologic indicator(s) to determine the	ModelRunIDis
	model run id in the database. The run date should	specified.
	be specified using the format YYYY-MM-DD	1
	hh:mm (zero-padded with hour 0-23, minute 0-59,	
	seconds and hundredths of seconds will default to	
	0). Need to implement tests to make sure this is	
	properly handled, including formatting and	
	listing existing values.	
Hydrologic	The hydrologic indicator(s) to use for the time	None – must be
Indicator	series; used with the model name, model run name,	specified unless
	and model run date to determine the model_run_id	ModelRunID is
	in the database.	specified.
ModelRunID	The model_run_id value to match the time series.	•
	If specified, the value will be used instead of the	
	model_run_id determined from ModelName,	
	ModelRunName, ModelRunDate, and	
	HydrologicIndicator.	
	Use the following parameters when reading an	
	ensemble of model time series.	
EnsembleName	The name of the ensemble to write. The	Must be specified if
	TSList=EnsembleID and EnsembleID	writing an ensemble.
	parameters also should be specified.	
EnsembleTraceID	Indicate how to identify time series trace identifiers.	The HDB trace number
	This parameter may be implemented in the future.	is used for the TSTool
		ensemble trace
		sequence number.
EnsembleModelName	The model name corresponding to the ensemble.	Must be specified if
		writing an ensemble.
EnsembleModel	When writing an ensemble, the model run date for	If not specified, the
RunDate	the ensemble, specified using format:	ensemble identifier in
	• YYYY-MM-DD hh:mm (zero-padded with	HDB will not include
	hour 0-23)	the model run date.
	• \${TS:property} - use a run date from a	
	time series property, truncated to minute	
	Need to implement tests to make sure this is	
	properly handled, including formatting and	
	listing existing values.	
	The following parameters are always appropriate.	
InputStart	Start of the period to query, specified in format	Read all available data.
	YYYY-MM-DD HH, with a precision appropriate	
	for the interval.	
InputEnd	End of the period to query, specified in format	Read all available data.
	YYYY-MM-DD HH, with a precision appropriate	
	for the interval.	
Alias	Indicate an alias to assign to time series, which can	No alias is assigned.
	result in shorter identifiers for time series when	
	referenced with other commands.	

This page is intentionally blank.

Command Reference: ReadRiversideDB()

Read time series from a RiversideDB database

Version 10.06.00, 2012-04-04

The ReadRiversideDB() command reads one or more time series from a RiversideDB database (see the **Riverside Data Store Appendix** for more information). It is designed to utilize query criteria to process large numbers of time series. The RiversideDB design is highly consistent with TSTool conventions and therefore time series properties in RiversideDB, including time series identifier information, map closely to TSTool internal data representations.

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

🔷 Edit Rea	🔹 Edit ReadRiversideDB Command						
Read 1+ time s	Read 1+ time series from a RiversideDB database.						
Specifying the	period will limit data tha	it are availat	le h	or fill commands bu	it can in	crease performance.	
Data store:	RiversideDB_TSTool	*				Required - data store containing data.	
Data type:	QIN - INSTANTANEO	JS OBSERVE	D RJ	IVER DISCHARGE	~	Required - data type for time series.	
Data interval:	15MINUTE 🔽					Required - data interval (time step) for time series.	
Where: Station	n Identifier (ID) 👘 🔽	Matches	¥	01350101			
Where: Scena	rio 🔽	Matches	*	RAW			
Where:	*	Matches	¥				
Where:	~	Matches	*				
Where:	~	Matches	¥				
Where:	~	Matches	¥				
Input start:	2011-10-08 00:00]			Optional - override the global input start.	
Input end:	2011-11-03 08:15]			Optional - override the global input end.	
Alias to assign:	%L-%T	Insert:	S	ielect Specifier	*	Optional - use %L for location, etc. (default=no alias).	
Missing value:	NaN					Optional - missing data value (default=-999, recommended=Na	iN).
	ReadRiverside	DB (Data	Sto	ore="Rivers:	ideDB	TSTool", DataType="QIN - INSTANTANEOUS	
OBSERVED RIVER DISCHARGE", Interval="15MINUT							
(ID); Matches; 01350101		1",	Where2="Sca	enari	;Matches;RAW",InputStart="2011-10-08		
	00:00", InputE	nd="201	1-1	L1-03 08:15'	',Ali	as="%L-%T",MissingValue="NaN")	
				Cancel			
						ReadRiversi	deDB

ReadRiversideDB() Command Editor

The **Data type**, **Data interval**, and **Where** input fields are similar to those from the main TSTool interface. However, whereas the interactive interface first requires a query to find the matching time series list and then an interactive select for specific time series identifiers, the ReadRiversideDB() command reads all matching time series in one step. This can greatly shorten command files and simplify command logic, especially when processing large amounts of data. It may be necessary to specify more criteria where a single time series is needed.

ReadRiversideDB(Parameter=Value,...)

Parameter	Description	Default
DataStore	The data store name, indicating the RiversideDB database to query.	None – must be
		specified.
DataType	The data type to be queried, determined from time series that are	None – must be
	available in the database.	specified.
Interval	The data interval for the time series, determined from time series	None – must be
	that are available in the database matching the DataType.	specified.
WhereN	The "where" clauses to be applied when querying data, matching	If not specified,
	the values in the <i>Where</i> fields in the command editor dialog and	the query will not
	the TSTool main interface. The parameters should be named	be limited and
	Where1, Where2, etc., with a gap resulting in the remaining	very large
	items being ignored. The format of each value is:	numbers of time
		series may be
	"Item;Operator;Value"	queried.
	Where Item indicates a data field to be filtered on, Operator is the	
	type of constraint, and Value is the value to be checked when	
	querying.	
InputStart	Start of the period to query, specified as a date/time with a	Read all available
	precision that matches the requested data interval.	data.
InputEnd	End of the period to query, specified as a date/time with a precision	Read all available
	that matches the requested data interval.	data.
Alias	The alias to assign to the time series, as a literal string or using the	
	special formatting characters listed by the command editor. The	
	alias is a short identifier used by other commands to locate time	
	series for processing, as an alternative to the time series identifier	
	(TSID).	
MissingValue	Value to use to indicate missing data values within the time series.	-999
	-999 is the default for historical reasons; however, NaN (not a	
	number) is being phased in and should be specified if possible.	
	Null values in the database will be converted to the missing data	
	value.	

Command Reference: ReadRiverWare()

Read a single time series from a RiverWare file

Version 10.00.00, 2011-03-28

The ReadRiverWare() command reads a single time series from a RiverWare file (see the **RiverWare Input Type Appendix**) and assigns an alias to the result.

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

() Edit ReadRiver	S Edit ReadRiverWare() Command					
Read a single time serie	Read a single time series from a RiverWare time series file.					
It is assumed that the fi	lename follows the convention	on ObjectName.SlotNam	е.			
The ObjectName and Sk	otName will be used for the t	ime series identifier local	tion and dal	a type, respectively.		
Specify a full path or rel	ative path (relative to worki	ng directory) for a River	Nare file to	read.		
Specifying the period wi	ll limit data that are available	for fill commands but ca	in increase	performance.		
If not specified, the per	iod defaults to the global inp	ut period.				
The working directory is	: C:\Develop\TSTool_Source	Build\TSTool\test\regres	sion\UserM	anualExamples\TestCases\CommandReferen	ce\ReadRiverWare	
RiverWare file to read:	SouthHolstonData.SOGPool	Elevation			Browse	
Alias to assign:	ts1	Insert: Select Specif	ier 🔽	Required - use %L for location, etc.		
Input start:				Optional - overrides the global input start.		
Input end:				Optional - overrides the global input end.		
	ReadRiverWare(Al:	ias="ts1", Input	File="2	outhHolstonData.SOGPoolElev	ration")	
Command:						
		Aud working Directory				
					ReadRiverWare	

ReadRiverWare() Command Editor

ReadRiverWare(Parameter=Value,...)

The following older command syntax is updated to the above syntax when a command file is read:

TS Alias = ReadRiverWare(Parameter=Value,...)

Parameter	Description	Default
InputFile	The name of the RiverWare file to read,	None – must be specified.
	surrounded by double quotes. The path	
	to the file can be absolute or relative to	
	the working directory.	
Alias	The alias to assign to the time series, as a	None – must be specified.
	literal string or using the special	
	formatting characters listed by the	
	command editor. The alias is a short	
	identifier used by other commands to	
	locate time series for processing, as an	
	alternative to the time series identifier	
	(TSID).	
Units	The units for the time series. The data	Use the units read from the file.
	values will be converted to these units.	This parameter is not yet
	TSTool by default understands certain	enabled.
	units abbreviations and attempting to	
	convert to or from unknown units may	
	not be possible. The ability to handle	
	user-defined units is being evaluated.	
	See the Scale() and	
	ConvertDataUnits() commands.	
InputStart	The start of the period to read data –	Use the global query period.
	specify if the period should be different	
	from the global query period.	
InputEnd	The end of the period to read data –	Use the global query period.
	specify if the period should be different	
	from the global query period.	

Command Parameters

A sample command file is as follows:

ReadRiverWare(Alias="ts1",InputFile="SouthHolstonData.SOGPoolElevation")

Command Reference: ReadStateCU()

Read time series from a StateCU time series or report File

Version 09.07.02, 2010-08-20

The ReadStateCU() command reads all the time series in a StateCU time series file (e.g., frost dates) or report file (e.g., IWR, WSL) (see the **StateCU Input Type Appendix**).

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

🛃 Edit ReadStateC	U() Command		X			
Read 1+ (or all) time ser	ies from one of the followi	ng file types, using data in the file to assign the identifier:				
Crop pattern time ser	ies file (StateCU input file)).				
Irrigation practice tim	practice time series file (input).					
StateCU IWR or WSL	report file (output).					
Specify a full or relative	path (relative to working d	lirectory).				
The working directory is	C:\Develop\TSTool_Source	ceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandRi	eference\ReadStateCU			
The time series identifier	pattern, it specified, will h	ilter the read ONLY for IWR and WSL files;				
specify blank to read all	or use * wildcards to matci	h a time series identifier,				
ABC * TWP Month	monunity twick time series fo	or locations starting with ABC, specify:				
Location can be X. X*. o	*. Data type and interva	al can be * or combinations as follows:				
CropArea-AllCrons (Ve	ar)					
IWR or WSL (Month, Y	ear)					
IWR_Depth or WSL_D	epth (Month, Year)					
StateCU file to read:	Data\ym2004.iwr		Browse			
Input start:		Optional - overrides the global input start (default=read all).			
Input end:		Optional - overrides the global input end (default=read all)				
Time series ID:		Optional - Loc.Source.Type.Interval to filter (default=read	all).			
New scenario:		Optional - to help uniquely identify time series (default=nor	ne).			
Automatically adjust?:	~	Optional - convert data type with "," to "-" to work in TSID	(default=True).			
Check data after read?:	×	Optional - check data integrity after read? (default=True).				
	ReadStateCU(Inp	outFile="Data\ym2004.iwr")				
Command:						

ReadStateCU() Command Editor

ReadStateCU(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
InputFile	The name of the StateCU time series or	None – must be specified.
	report file to read, surrounded by double	
	quotes.	
InputStart	The starting date/time to read, specified	Read all the data.
	to a precision (month or year) that	
	matches the data file.	
InputEnd	The ending date/time to read, specified to	Read all the data.
	a precision (month or year) that matches	
	the data file.	
TSID	A time series identifier pattern that will	Read all time series.
	be used to filter the list of time series that	
	are read. See the figure above for	
	examples.	
NewScenario	A new scenario to use for the TSID.	No scenario.
	This is useful when reading data from	
	multiple model runs that otherwise	
	would have the same TSIDs.	
AutoAdjust	Indicate whether to automatically adjust	True
	time series identifiers to use a dash "-"	
	instead of period "." in the data type,	
	necessary because StateCU data types	
	(e.g., crop types that include CU method)	
	have a period that interferes with the	
	normal TSID convention.	
CheckData	Indicate whether to check the data for	True
	integrity after reading. Currently only	
	the irrigation practice time series can be	
	checked, to verify that the acreage totals	
	are the sum of the parts.	

A sample commands file is as follows:

ReadStateCU(InputFile="Data\ym2004.iwr")

Command Reference: ReadStateCUB()

Read time series from a StateCU binary output time series file

Version 08.17.00, 2008-10-02

The ReadStateCUB() command reads time series from a StateCU binary output time series file (see the **StateCUB Input Type Appendix**). The actual reading occurs as the commands are being processed. For this reason and because the number of time series in the binary file is usually large, if any other commands reference the StateCU binary file time series, the time series identifiers must be specified manually or use wildcards in identifiers (identifiers are not available to list in dialogs). Only data types that contain floating point numbers will be read.

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

🛃 Edit ReadStateCUB() C	Command	×			
Read all the time series from a	Read all the time series from a StateCU binary output file, using information in the file to assign the identifier.				
Due to the large number of tim	ne series in StateCU binary files, the list of time series identifiers in the file will NOT be available	e in other command editors.			
Specify a full or relative path ((relative to working directory).				
The working directory is: $\ensuremath{C:UD}$	evelop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ReadStateCUB				
The time series identifier patte	ern, if specified, will filter the read.				
Use blank or * to read all time	e series.				
Use A* to read all time series	s with alias or location starting with A.				
Use *.*.XXXXXX.*.* to read all	time series with data type XXXXX.				
Currently, data source, interv	val, and scenario are internally defaulted to *.				
StateCU binary file to read:	eata\SP2008_OutOfOrder.BD1	Browse			
Time series ID:	Optional - specify a TSID pattern to match.				
Input start: 1	970-01 Optional - default is global input start or all data.				
Input end: 1	971-12 Optional - default is global input end or all data.				
Command: 1	eadStateCUB(InputFile="Data\SP2008_OutOfOrder.BD1",InputStart="19 971-12")	970-01",InputEnd="			
	Add Working Directory Cancel OK	ReadState			

ReadStateCUB() Command Editor

ReadStateCUB(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
InputFile	The name of the StateCU binary time	None – must be specified.
	series file to read, surrounded by double	
	quotes. The path to the file can be	
	absolute or relative to the working	
	directory.	
TSID	Time series identifier pattern to filter the	Read all time series.
	read.	
InputStart	The starting date/time to read data,	Read all data.
	specified to Month precision.	
InputEnd	The ending date/time to read data,	Read all data.
	specified to Month precision.	

The following example command file illustrates how to read all CU Shortage time series:

ReadStateCUB(InputFile="Data\farmers.BD1",TSID="*.*.CU Shortage.*.*")

The following example illustrates how to read all time series from a binary file with debug turned on to echo all information that is read.

```
StartLog(LogFile="commands.TSTool.log")
SetDebugLevel(LogFileLevel=1)
ReadStateCUB(InputFile="Data\farmers.BD1")
```

Command Reference: ReadStateMod()

Read all the time series from a StateMod time series file

Version 09.05.03, 2009-11-17

The ReadStateMod() command reads all the time series in a StateMod time series file (see the **StateMod Input Type Appendix**). Single time series can be read by using time series identifier (TSID) commands.

Water rights files also can be read and converted to time series – this is useful for visualization, water supply analysis, and is used to test well right processing. Considering all water rights for a location based on the administration number results in a step function of decree over time. Monthly and yearly time series use calendar year and a right is active if it is turned on anywhere in the month or year. Free water rights (e.g., those having administration numbers > 90000.00000 are treated like other rights and therefore may not impact the results in the current period because the corresponding appropriation date is in the future (additional parameters may be added in the future to allow more ways to process these rights). If processing well rights and multiple years of parcel data are processed, this command executes the same logic as the StateDMI MergeWellRights() command.

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

😹 Edit ReadStateMod() Command				
Read all the time series from a StateMod time series or water right file, using information in the file to assign the identifier. The data source and data type will be blank in the resulting time series identifier (TSID). Specify the interval and parcel year only for well rights, as appropriate. Specify a full or relative path (relative to working directory). The working directory is: C:\Develop\TSTool SourceBuild\TSTool\test\regression\commands\general\ReadStateMod				
StateMod file to read:	ym2004.ddh	Browse		
Input start:	Optional - overrides the global input start.			
Input end:	Optional - overrides the global input end.			
Alias to assign:	Insert: %A - Alias 🛛 🗸 Optional - use %L for location, etc. (default=no alias).			
Interval:	Optional for rights - interval for resulting time series (default=Year).			
Spatial aggregation:	Optional for rights - spatial aggregation (default=Location).			
Parcel year:	Optional for well rights - read a single irrigated lands year (default=read all).			
	ReadStateMod(InputFile="ym2004.ddh")			
Command:				
Add Working Directory Cancel OK				

ReadStateMod() Command Editor

ReadStateMod(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
InputFile	The name of the StateMod time series file to read, surrounded by double quotes. The path to the file can be absolute or relative to the working directory. Global property values can be inserted using the syntax \${PropertyName} (see also the SetProperty() command).	None – must be specified.
InputStart	The start of the period to read data – specify if the period should be different from the global query period. Specify to a precision that matches the data. If reading water rights, the output time series will start on this date.	Use the global query period or if not specified read all data. The default for water rights is the date of the first right.
InputEnd	The end of the period to read data – specify if the period should be different from the global query period. Specify to a precision that matches the data. If reading water rights, the output time series will end on this date.	Use the global query period or if not specified read all data. The default for water rights is the date of the last right.
Alias	The alias to assign to the time series that are read. Use the format choices and other characters to define a unique alias.	No alias is assigned.
Interval	When reading a water right file, specify the interval for the resulting time series, one of Day, Month, or Year.	Year
Spatial Aggregation	 When reading a water right file, indicate how time series are to be aggregated spatially, one of: Location - aggregate by the station identifier. Parcel - (only used with well rights) aggregate based on the parcel number and parcel year. None - do not aggregate spatially, which will result in constant value time series for each water right. 	Location
ParcelYear	When processing a well water right file, indicate the year of parcel data to process. Parcel configurations change from year to year, and a single year of parcel data can be processed if desired.	Process all parcel years.

A sample command file is as follows:

ReadStateMod(InputFile="ym2004.ddh")

Command Reference: ReadStateModB()

Read time series from a StateMod binary output time series file

Version 09.06.00, 2010-01-05

The ReadStateModB() command reads time series from a StateMod binary output time series file (see the **StateModB Input Type Appendix**). The identifiers (or aliases) from the time series will be available as choices when editing other commands. If this causes performance issues due to the large number of time series that may be read, limit the time series that are read using the TSID parameter.

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

😹 Edit ReadStateModB	() Command		
Read time series from a State Specify a full or relative path The working directory is: C:\D The time series identifier path Use blank or * to read all tim Use A* to read all time serie: Use *.*.XXXXX.*.* to read a Currently, data source, inter	Mod binary output file, using information in the file (relative to working directory). vevelop\TSTool_SourceBuild\TSTool\test\regression er, if specified, will filter the read. e series. s location starting with A. all time series with data type XXXXX. rval, and scenario are internally defaulted to *.	to assign the identifier. \commands\general\ReadStateModB	
StateMod binary file to read:	Data/ex7_Version12.29.b43		Browse
Time series ID:	*.*.Available_Flow.*.*	Optional - specify a TSID pattern to match (default is all).	
Input start:		Optional - override the global input start.	
Input end:		Optional - override the global input end.	
StateMod version:		Optional - for files prior to StateMod version 11 (default is current version)	
Alias to assign:	%L-%T Insert: %A - Alias 💌	Optional - use %L for location, etc. (default=no alias).	
Command:	ReadStateModB(InputFile="Data/ex7_Version12.29.b43",TSID="*.*.Available_Flow.*.*",Al ias="%L-%T")		
Add Working Directory Cancel OK			

ReadStateModB() Command Editor

```
ReadStateModB(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
InputFile	The name of the StateMod binary time series file	None – must be
	to read, surrounded by double quotes. The path to	specified.
	the file can be absolute or relative to the working	
	directory. Global property values can be inserted	
	using the syntax \${PropertyName} (see also	
	the SetProperty() command).	
TSID	Time series identifier pattern to filter the read.	Read all time series.
	Use periods to indicate separate TSID parts and	
	use * to match patterns within the parts.	
InputStart	The starting date/time to read data, specified to	Read all data.
	Day or Month precision based on whether a	
	daily or monthly model run.	
InputEnd	The ending date/time to read data, specified to	Read all data.
	Day or Month precision based on whether a	
	daily or monthly model run.	
Version	StateMod version number using the form NN.NN	Detect from the file if
	(padded with leading zero for version 9)	possible.
	corresponding to the file, necessary because the	
	file version number (and consequently	
	parameters) cannot be automatically detected in	
	older versions. Changes in binary file format	
	occurred with version 9.01 and 9.69, mainly to	
	add new data types. The StateMod file version	
- 7 1	for version 11+ is automatically detected.	
Alias	The alias to assign to the time series that are read.	No alias is assigned.
	Use the format choices and other characters to	
	define a unique alias.	

The following example command file illustrates how to read all Available_Flow time series for identifiers starting with 44 (e.g., to extract all such time series for a water district):

ReadStateModB(InputFile="..\StateMod\ym2002b.b43",TSID="44*.*.Available_Flow.*")

The following example illustrates how to read all time series from a binary file that was created with StateMod version 9.53. As shown in the example, debug can be turned on for the log file to evaluate issues with the file format.

```
StartLog(LogFile="commands.TSTool.log")
SetDebugLevel(0,1)
ReadStateModB(InputFile="COLOFB.B43",Version="09.53")
```

Command Reference: ReadTableFromDataStore()

Read a table from a datastore

Version 10.21.00, 2013-06-21

The ReadTableFromDataStore() command executes a database query for a datastore that is associated with a database, and places the result in a TSTool table, which can subsequently be processed with other TSTool commands. This command cannot be used with web service datastores because the underlying software relies on a database to perform the query. If database datastore support is not specifically provided by TSTool, a generic datastore can be used (see the **Generic Database DataStore** appendix). This command is useful when the database can provide results with a simple query and tight integration with TSTool is not required or has not been implemented. The query can be specified in the following ways:

- Specify a single table/view to query:
 - the list of tables is filtered to remove internal database tables; however, this capability varies by database product and in some cases internal tables will be listed
 - \circ $\,$ the query is constructed from the provided database table/view name and column names
 - the output can be sorted by specifying column names
 - o "where" clauses currently are not supported but may be added in the future
 - the top N rows of the result can be returned to allow "peeking" at tables (may not be available for all database software)
- Specify a SQL select statement:
 - o SQL must be valid for the database (syntax may vary based on database software)
 - Use \${Property} notation to insert processor property values set with SetPropety().
 - SQL syntax is not checked for validity and therefore error messages from the database may be more difficult to interpret
- Specify an SQL select statement in a file:
 - o Similar to the above option; however, the SQL statement is read from a file
 - Useful if the SQL statement is also used by other tools
- Specify a procedure to run:
 - Available procedures are listed and can be selected
 - Currently, only procedures that do not require parameters can be run

General constraints on the query are as follows:

- the table, views, and procedures being queried must be readable (some databases restrict direct access to data and require using stored procedures)
- the resulting table in TSTool will have columns with names that match the database query results
- data types for columns will closely match the database results:
 - o data will be treated as strings if unable to match the database column type
 - o the precision of floating point numbers for displays is defaulted to 6 digits
 - null values in the database will transfer to null values in the TSTool table and will display as blank table cells
 - date/time columns in the database will be represented as such in the TSTool table; however, it may not be possible to limit the precision of the date/time (i.e., hours, minutes, and seconds may be shown with default zero values in output)

Future enhancements will add additional features to intelligently map database results to TSTool tables.

The following dialog is used to edit the command and illustrates the syntax for the command, in this case reading a small table from the State of Colorado's HydroBase.

👌 Edit R	eadTabl	eFromDataStore() Command		
This commar The query c 1) Specify 2) Specify 3) Similar The wo 4) Specify The resulting	This command reads a table from a database datastore table, view, or procedure. The query can be specified in one of four ways: 1) Specify a single table or view, related columns, and sort order. 2) Specify a free form SQL select statement (allows joins and other SQL constructs supported by the database software). 3) Similar to 2; however, the SQL statement is read from a file, which can be specified relative to the working directory. The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ReadTableFromDataStore 4) Specify a database procedure to run (under development). The resulting table columns will have data types based on the query results.			
Datastore:	HydroBas	e Required - data store containing data to read.		
-Specify ho Table and	ow to query d columns	y the datastore		
Datasto Datastore (Tor	ore table: : columns: Order by: p N rows:	vw_CDSS_Cropchar Required - database table/view to read. Optional - database table/view columns, separated by commas (default=all). Optional - columns to sort by, separated by commas (default=no sort). Optional - return top N rows (default=return all).		
Table ID:	CropChar	Required - unique identifier for the output table.		
ReadTableFromDataStore(DataStore="HydroBase", DataStoreTable="vw_CDSS_Cropchar", TableID="CropChar") Command:				
	Remove Working Directory Cancel OK			
		ReadTableFromDataStore Table		

ReadTableFromDataStore() Command Editor When Querying a Single Table

The corresponding output table is as shown below:

ROPNUM	METHOD_DESC	CROPNAME	TEMPEARLYMOISTURE	TEMPLATEMOISTURE	INITIALROOT	MAXROO
	1 BLANEY-CRIDDLE_TR-21	ALFALFA.TR21	50.000000	28.000000	4.900000	
	8 BLANEY-CRIDDLE_TR-21	GRASS_PASTURE.TR21	45.000000	45.000000	3.300000	-
	9 BLANEY-CRIDDLE_TR-21	ORCHARD_WITH_COVER.TR21	50.000000	45.000000	5.000000	
	10 BLANEY-CRIDDLE_TR-21	ORCHARD_WO_COVER.TR21	50.000000	45.000000	5.000000	
	7 BLANEY-CRIDDLE_TR-21	GRAPES.TR21	55.000000	50.000000	4.100000	
	6 BLANEY-CRIDDLE_TR-21	DRY_BEANS.TR21	60.000000	32.000000	2.500000	
			15 000000		0 500000	3

Example ReadTableFromDataStore() Command Output Table

The following example illustrates using an SQL query string, in this case to read diversion records for a specific structure in HydroBase:

S Edit ReadTableFromDataStore() Command		
This command reads a table from a database datastore table, view, or procedure. The query can be specified in one of four ways: 1) Specify a single table or view, related columns, and sort order. 2) Specify a free form SQL select statement (allows joins and other SQL constructs supported by the database software). 3) Similar to 2; however, the SQL statement is read from a file, which can be specified relative to the working directory. The working directory is: C:QDvelop[TSTool_SourceBuild]TSTool[test]vregression[commands]general[ReadTableFromDataStore 4) Specify a database procedure to run (under development). The working table columns will have data types based on the query results. Datastore: !vydroBase Table and columns SQL string SQL file Procedure Select * from vw_CDSS_AnnualAmt where meas_num = (select meas_num from vw_CDSS_StructureStructMeasType where wd = 1 and id = 501 and meas_type = 'DivTotal' and time_step='Annual') and irr_year >= 1970 and irr_year < 1980 order by irr_year		
Table ID: DivTotal Required - unique identifier for the output table.		
Command: ReadTableFromDataStore(DataStore="HydroBase",Sql="select * from vw_CDSS_AnnualAmt where meas_num(select meas_num from vw_CDSS_StructureStructMeasType where wd1 and id501 and meas_typeDivTotal and time_stepAnnual) and irr_year >1970 and irr_year < 1980 order by irr_year",TableID="DivTotal")		
Remove Working Directory Cancel OK		

ReadTableFromDataStore() Command Editor When Specifying a SQL Query String

ReadTableFromDataStore(Parameter=Value,...)

Command Parameters				
Parameter	Description	Default		
DataStore	The name of a database datastore to read.	None – must be specified.		
DataStoreTable	The name of the database table or view to	None.		
	read when querying a single table or view.			
	If specified, do not specify Sql or			
	SqlFile.			
DataStoreColumns	When reading a single table/view, the	All columns from		
	names of the columns to read, separated	DataStoreTable are		
	by commas.	read.		
OrderBy	When reading a single table/view, a list of	Default database sort order		
	column names separated by commas to	will be used.		
	control the order of output.			
Тор	Indicate that Top rows should be returned.	Return all rows.		
	This functionality may not be			
	implemented for all databases (SQL is not			
	fully standardized for this feature). This			
	parameter is useful to determine the			
	columns for a table prior to using the Sql			
	or SqlFile parameters.			
Sql	The SQL string that will be used to query	None.		
	the database, optionally using			
	\${Property} notation to insert			
	processor property values. If specified, do			
	not specify DataStoreTable or			
	SqlFile.			
SqlFile	The name of the file containing an SQL	None.		
	string to execute, optionally using			
	\${Property} notation in the SQL file			
	contents to insert processor property			
	values. If specified, do not specify			
	DataStoreTable or Sql.			
DataStoreProcedure	The name of the database procedure to	None.		
	run. Currently, only procedures that do			
	not require parameters can be run.			
TableID	Identifier to assign to the output table in	None – must be specified.		
	TSTool, which allows the table data to be			
	used with other commands. A new table			
	will be created.			

Command Reference: ReadTableFromDBF() Read a table from a dBASE file

Version 09.09.00, 2010-09-23

The ReadTableFromDBF() command reads a table from a dBASE file, such as the files used with ESRI GIS shapefiles. dBASE files are self-contained binary database files.

Handling of dBASE files is limited and support for newer features may not be included.

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

Edit ReadTableFromDBF() Command			
This command reads a table from a dBASE file. The table can then be used by other commands. An example of a dBASE file is the attribute table file used with ESRI GIS shapefiles. The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ReadTableFromDBF			
Table ID:	ClimateStations	Required - unique identifier for the table.	
Input file:	Data\div1_climatestations.dbf	Browse	
Command:	ReadTableFromDBF(TableID="ClimateStations", InputFile="Data\div1 climatestations.dbf")		
Add Working Directory To File Cancel OK			
		ReadTableFror	

ReadTableFromDBF() Command Editor

ReadTableFromDBF(Parameter=Value,...)

Parameter	Description	Default
TableID	Identifier to assign to the table that is	None – must be specified.
	read, which allows the table data to be	
	used with other commands.	
InputFile	The name of the file to read, as an	None – must be specified.
	absolute path or relative to the command	
	file location.	

Command Reference: ReadTableFromDelimitedFile()

Read a table from a delimited file

Version 10.03.00, 2012-01-09

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.
- 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. This command currently does not deal with """text"" notation although support may be added in the future (see information about comma-separated-value [CSV] standards: http://en.wikipedia.org/wiki/Comma-separated_values).

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

Edit ReadTableFromDelimitedFile() Command			
his 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 (user-specified delimiters will be added in the future). An example data file is shown below (line and data row numbers are shown on the left for illustration):			
<pre>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 "Headerl", "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</pre>			
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:\Develop\ISTool.SourceBuild\ISTool\test\regression\commands\general\BeadTableEropDelimitedFile			
Table ID: Table1 Required - unique identifier for the table.			
Input file: Sample.csv Browse			
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).			
Command: ReadTableFromDelimitedFile(TableID="Table1",InputFile="Sample.csv",SkipLines="2"			
Add Working Directory To File Cancel OK			

ReadTableFromDelimitedFile() Command Editor

The command syntax is as follows:

ReadTableFromDelimitedFile(Parameter=Value,...)

Parameter	Description	Default
TableID	Identifier to assign to the table that is read, which allows the table data to be used with	None – must be specified.
	other commands.	
InputFile	The name of the file to read, as an absolute	None – must be specified.
	path or relative to the command file location.	
SkipLines	Indicates the number of lines in the file to	No lines are skipped.
	skip, which otherwise would interfere with	
	reading row data. Individual row numbers	
	and ranges can be specified, for example:	
	1,5-6,17	
HeaderLines	Indicate the rows that include header	If the first non-comment line
	information, which should be used for	contains quoted field names,
	column names. Currently this should only be	they are assumed to be
	one row, although a range may be fully	headers. Otherwise, no
	supported in the future.	headers are read.

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

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
```

This page is intentionally blank.

Command Reference: ReadTableFromExcel()

Read a cell range from a Microsoft Excel file and create a new Table

Version 10.18.00, 2012-02-25

The ReadTableFromExcelFile() command reads a table from a Microsoft Excel file, more specifically from a worksheet in an Excel workbook file. A contiguous block of cells (rectangle) must be specified in one of the following ways:

- Specify a range of cells using Excel address notation (e.g., A1:D10) (TODO figure out if worksheet can be specified in this address, in which case the Worksheet parameter is not required).
- Specify the name of an Excel named range.
- Specify a table name (essentially a named range).

Table column types (number, text, etc.) are determined from the cells in the first data row being read (NOT the column name row) – data types must be consistent for all cells in a column, although blanks are allowed. Table column names are determined according to the ExcelColumnNames command parameter.

TSTool uses the Apache POI software, version 3.9 (http://poi.apache.org) to read the Excel file and consequently functionality is constrained by the features of that software package. The software reads and writes Excel files. It does not communicate with a running Excel program, as does other software tools (for example IronPython using Excel interoperability libraries). POI does not fully implement Excel functionality and consequently some formula capabilities are not available, which will generate errors getting values for some cells. One solution, for example to create test data in Excel, is to copy cells with "paste special" and then paste the values. It is expected that updates to POI will continue to add more formula support.

Table columns must contain consistent data types (all strings, all numeric, etc.). The following table describes how column types are determined and data values are transferred to the table. Column type determination uses the first data row in the specified address range. If a column is determined to be a type and then cell values in the column are different, conversions are made to maintain the intent of the values if possible. For example, a Boolean value stored in a cell will get converted to 1.0 if the table column has been determined to be for double precision numbers. Errors in processing cells may result in empty cell values in the output table.

Excel Cell Format	
("Number Category"	Conversion from Excel to TSTool Table
Number:	• If Excel cell is internally a "numeric", convert to a double-precision
• General	number, where the format "Decimal places" is used in the TSTool table
• Number	for formatting. The number of decimal places in Excel is fixed for some
Currency	of the number categories shown on the left (e.g., Special=Zip Code).
Accounting	Excel internally stores integers as numbers with zero decimals. Need to
• Percentage	figure out how to get the Excel cell formatting number of decimals to
• Fraction	similarly set in the output table – but DO NOT assume zero decimals
Scientific	should convert to an integer.
• Special	• See the ExcelIntegerColumns parameter, which specifies the
• Custom	output table to use integers.
	• If Excel cell is internally a "Boolean", convert to an integer having

Excel Data Type Conversion to Table

Excel Cell Format			
("Number Category"	Conversion from Excel to TSTool Table		
	values 0 or 1. Need to evaluate having a parameter		
	ExcelBooleanColumns to transfer to a Boolean column in the		
	output table. Excel seems to handle Booleans as text with values True		
	or False.		
Date:	TSTool does not generally deal with only time and therefore implementation		
• Date	is limited. The POI library does not seem to have all date/time functions		
• Time	implemented.		
Text:	Converts to a string.		
• Text			
Blank	• Treated as Text (may in the future scan down the column to determine		
	data type from first non-blank cell).		
	• Blank cells found once the column type is determined are set to empty		
	strings in text columns, and null in number and date columns.		
Error	• Treated as Text (may in the future scan down the column to determine		
	data type from first non-error cell).		
	• Blank cells found once the column type is determined are set to empty		
	strings in text columns, and null in number and date columns.		
Formula	Expanded internally and the resulting cell value is set in the output table.		
	POI does not support all formulas and errors may be generated, which result		
	in empty output table cells.		

Consider the following Excel worksheet example, which is equivalent to a comma-separated-value (CSV) file that has comments at the top and four columns:

	А	В	С	D	E	F
1	# INSIGHT dat	a interva	al types			
2	# Mainly used	to ensu	re constrair	nt on time s	eries	
3	Abbreviation	Base	Multiplier	IsIrregular		
4	Day	Day	1	0		
5	Month	Month	1	0		
6	Year	Year	1	0		
7						

Example Excel Workshet With Comments, Column Names, and Text and Integer Columns

Although it is possible to use comments in Excel (annotation on cells), these comments cannot be saved in simple text files like CSV files. Consequently, for transparency and automation of a full process, embedding comments in the worksheet may make sense. Note also that the numeric cells are formatted as type "Number" with 0 decimals in Excel. Internally, Excel does not have an integer data type and consequently it is difficult for the ReadTableFromExcel() command to know when to convert a zero-decimal number in Excel to a floating point or integer number in the output table (it therefore defaults to a floating point number in output). To make this conversion more explicit, use the ExcelIntegerColumns command parameter. The comment lines in the above example will be ignored in determining the headings, and any data rows that have a first cell value starting with the comment character will be ignored. The following dialog is used to edit the command and illustrates the syntax for the command when reading the above Excel worksheet.

Edit ReadTableFromExcel() Command				
This command reads a table from a worksheet in a Microsoft Excel workbook file (*.xls, *.xlsx). A contiguous block of cells must be specified using one of the address methods below. It is recommended that the location of the files be specified using a path relative to the working directory. The working directory is: 1//Spanshots/2012/TSTool				
Table ID:	Excel_InsightDataIntervalTypes	Required - unique identifier for the created table.		
Input (workbook) file:	\DataForImport\Definitions\Insi	ghtDefinitions.xlsx Browse		
Worksheet:	InsightDataIntervalTypes	Required (if not in address) - worksheet name (default=first sheet).		
CSpecify the address f	or a contigous block of cells the in	Excel worksheet		
By Excel Address	Range By Table Name			
Excel address:		Excel cell block address in format A1:B2.		
Excel column names:	FirstRowInRange 💌	Optional - how to define column names (default=ColumnN).		
Comment character:	#	Optional - character that indicates comment lines (default=none).		
Excel integer columns:	Multiplier, IsIrregular	Optional - columns that are integers, separated by commas.		
Read all as text?:	~	Optional - read all cells as text? (default=False).		
Command:	<pre>ReadTableFromExcel(TableID="Excel_InsightDataIntervalTypes",Inpu tFile="\DataForImport\Definitions\InsightDefinitions.xlsx",Wor ksheet="InsightDataIntervalTypes",ExcelColumnNames=FirstRowInRan ge,Comment="#",ExcelIntegerColumns="Multiplier,IsIrregular")</pre>			
Add Working Directory To File Cancel OK				

ReadTableFromExcel() Command Editor

The command syntax is as follows:

ReadTableFromExcelFile(Parameter=Value,...)

Parameter	Description	Default
TableID	Identifier to assign to the table that is read,	None – must be specified.
	which allows the table data to be used with	
	other commands.	
InputFile	The name of the Excel workbook file	None – must be specified.
	(*. <i>xls</i> or *. <i>xlsx</i>) to read, as an absolute path	_
	or relative to the command file location.	
Worksheet	The name of the worksheet in the	Read the first worksheet. If
	workbook to read. Currently this is	no address parameter is
	required if a specific sheet is read but in	specified, read the entire
	the future it may be made optional because	worksheet.

Parameter	Description	Default
	the sheet can be determined from named range and table names (global resources in the workbook) and absolute Excel	
	addresses that include the sheet name.	
ExcelAddress	Indicates the block of cells to read into the table, using Excel address notation (e.g., A1:D10).	Must specify address using one of available address parameters.
ExcelNamedRange	Indicates the block of cells to read into the table, using an Excel named range.	Must specify address using one of available address parameters.
ExcelTableName	Indicates the block of cells to read into the table, using an Excel named range.	Must specify address using one of available address parameters.
ExcelColumnNames	Indicate how to determine the column	ColumnN, <mark>or</mark>
	names for the table, one of:	FirstRowInRange when
	• ColumnN – column name will be	ExcelTableName is
	Column1, Column2, etc.	specified?
	 FirstRowInRange - column 	
	names are taken from the first non- comment row in the address range	
	• RowBeforeRange – column names are taken from the first non-comment row before the address range	
Comment	Specify the character that if found at the	No comments are used.
	start of the first column in a row (not just	
	the specified address range) indicates that	
	in transforring data to the output table	
	Comments are particularly useful when	
	processing entire data sheets	
ExcelIntegerColumns	Indicate the names of columns (separated	Numeric columns are treated
	by commas) that should be treated as	as double-precision values in
	integer columns in the output table.	the output table.
ReadAllAsText	Indicate with True or False whether all	False – set table column
	columns in the Excel address block should	types using the first data row
	be treated as text columns.	

Command Reference: ReadTimeSeries()

Read a single time series using a full time series identifier

Version 10.21.00, 2013-05-17

The ReadTimeSeries() reads a single time series using the time series identifier to uniquely identify the time series. This generalized command is useful for converting time series identifiers from the TSTool interface into read commands that assign an alias to a time series. Because the command is generic, it does not offer specific parameters that may be found in read commands for specific input types. Use the specific read commands where available for additional functionality and more specific error handling. See also the **ReadTimeSeriesList()** command.

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

ſ	👌 Edit ReadTimeSeries	() Command			23
	This command is a general time series read command.				
	Its main purpose is to assig	gn an alias to a time series, which	is more convenient to u	se than the long time series identifier.	
	Read commands for specif	ic input types generally offer more	e options and should be	used if available.	
1	The alias should be descrip	otive and should not contain space	s, periods, or parenthe	ses.	
	Specify the period to read	using the SetInputPeriod() comma	and.		
1	See also the ReadTimeSeri	iesList() command.			
1	Time series identifier:	08235350.USGS.Streamflow.Day	~HydroBase		
l	Alias to assign:	Select Specifier ▼ => Al	amosa	Required - use %L for location, etc.	
l	If time series not found?:	Warn 👻		Required - how to handle time series that are no	t found.
l	Default units:			Optional - units when IfNotFound=Default.	
	Command:	Command: ReadTimeSeries(TSID="08235350.USGS.Streamflow.Day~HydroBase",Alias="Alamosa",IfNotFound=Warn)			as="
	Cancel OK				

ReadTimeSeries() Command Editor

```
ReadTimeSeries(Parameter=Value,...)
```

The following older command syntax is updated to the above syntax when a command file is read:

TS Alias = ReadTimeSeries (Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSID	The time series identifier of the time series to read. The identifier should	None – must be specified.
	include the input type (and input name, if	
	required). See the input type appendices	
	for examples of time series identifiers for	
	various input types.	
Alias	The alias to assign to the time series, as a	None – must be specified.
	literal string or using the special	
	formatting characters listed by the	
	command editor. The alias is a short	
	identifier used by other commands to	
	locate time series for processing, as an	
	(TSID)	
	(ISID). Indiactos horreto hondle missing time	Mara
TINOCFOUND	series one of:	Walli
	• Waren generate fatal warnings and	
	do not include in output.	
	• Ignore – generate non-fatal	
	warnings and do not include in	
	output.	
	• Default – generate non-fatal	
	warnings and create empty time	
	series for those that could not be	
	found. This requires that a	
	SetOutputPeriod() command	
	be used before the command to	
	define the period for default time	
	series.	
DefaultUnits	Default units when	Blank – no units.
	IfNotFound=Default.	

A sample command file to read data from the State of Colorado's HydroBase is as follows:

ReadTimeSeries(TSID="08235350.USGS.Streamflow.Day~HydroBase",Alias=TS1)

Command Reference: ReadTimeSeriesList()

Read one or more time series using location identifiers from a table

Version 10.21.00, 2013-05-17

The ReadTimeSeriesList() command reads one or more time series using location identifiers from a table, an example of which is shown below as a comma-separated value file:

```
# Example list file. Comments start with the # character.
# Column headings can be specified in the first non-comment row using quotes.
"Structure ID", "Structure Name"
500501,Ditch 501
500502,Ditch 502
# Invalid ID (see IfNotFound parameter)
509999,Ditch 9999
```

The command typically is used when reading time series from a single source and can streamline processing in the following situations:

- A list of identifiers may have been generated from a database query
- A list of identifiers may have been extracted from a model data set

TSTool uses the location identifiers in the table with the command parameters and internally creates a list of time series identifiers. The time series are of the standard form:

Location.DataSource.DataType.Interval[.Scenario]~DataStore[~InputName]

where the brackets indicate optional information. TSTool then queries each time series, which can be processed further by other commands. See also the ReadTimeSeries() command, which performs essentially the same functionality but only reads one time series.

Although it is possible to specify a datastore (or "input type") that reads from files by also using the InputName, this is not generally recommended because the ReadTimeSeriesList() command can only specify one input file name and the file will be reopened for each time series read. Instead, read commands for specific file formats should be used because these commands are typically optimized to read multiple time series from the files. Use the SetInputPeriod() command to set the period to read.

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

Read a list of time series	ising location identifiers in a table	
The information specified I	below is used with the location identifiers	s to create time series identifiers, which are then used to read the time series.
The time series identifiers	(TSIDs) are of the form:	
LocationID.DataSource.I	DataType.Interval.Scenario~DataStore~	~InputName
The term "DataStore" is us	sed generically to mean a database, web	o service, or file supplying time series data (also called "Input Type" elsewhere
Jse the SetInputPeriod()	command to specify the period to read.	
Table ID:	StationList	 Required - table containing list of location IDs.
Location ID column:	TSID	Required - name of column containing location IDs.
Data source:	NWIS	Optional or required depending on datastore.
Data type:	00060-00003	Optional or required depending on datastore.
Data interval:	Day 👻	Required - data interval (time step) for time series.
Scenario:		Optional.
Datastore:	DateValue	Required - needed to identify input database, file, etc.
Input name:	Data/testdata.dv	Optional - file name if required for datastore.
f time series not found?:	Warn 👻	Required - how to handle time series that are not foun
Default units:		Optional - units when IfNotFound=Default.
	ReadTimeSeriesList (Table	eID="StationList",LocationColumn="TSID",DataSou
Comments	rce="NWIS", DataType="000	060-00003", Interval="Day", DataStore="DateValue"
Command:	,InputName="Data/testdat	ta.dv",IfNotFound=Warn)
	6	ancel OK

ReadTimeSeriesList() Command Editor

The command syntax is as follows:

ReadTimeSeriesList(Parameter=Value, ...)

Parameter	Description	Default
TableID	The identifier for the table that provides the	None – must be specified.
	list of location identifiers.	
LocationColumn	The column in the table containing the	None – must be specified.
	location identifiers to use in time series	
	identifiers.	
DataSource	The data source in the time series identifier.	May or may not be required,
	For example, if using the State of	depending on the datastore or input
	Colorado's HydroBase, USGS indicates that	type. Refer to the input type
	data are from the United States Geological	appendices.
	Survey. See the datastore and input type	
	appendices for more information on	
	available data types.	
DataType	The data type in the time series identifier.	Usually required. Refer to the

Parameter	Description	Default
	For example, if using the State of	datastore and input type
Colorado's HydroBase, DivTotal is used		appendices.
	for diversion totals. See the input type	
	appendices for more information on	
	available data types.	
Interval	Data interval in the time series identifier,	None – must be specified.
	using standard values such as 15Minute,	
	6Hour, Day, Month, Year.	
Scenario	Scenario in the time series identifier.	Usually not required.
DataStore	The data store (or input type) in the time	None – must be specified.
	series identifier. Refer to the datastore and	
	input type appendices or the TSTool main	
	GUI for options.	
InputName	The input name in the time series identifier,	
	when a file name is required.	
IfNotFound	Indicates how to handle missing time series,	Warn
	one of:	
	• Warn – generate fatal warnings and do	
	not include in output.	
	• Ignore – generate non-fatal warnings	
	and do not include in output.	
	• Default – generate non-fatal	
	warnings and create empty time series	
	for those that could not be found. This	
	requires that a SetOutputPeriod()	
	command be used before the	
	command to define the period for	
	default time series.	
DefaultUnits	Default units when	Blank – no units.
	ItNotFound=Default.	

IfNotFound=Default)

A sample command file to process monthly diversion data from the State of Colorado's HydroBase database is as follows:

```
# Read monthly diversion total from HydroBase for the structures in the list
# file. The data source is set to DWR because data source is saved in
# HydroBase.
ReadTimeSeriesList(TableID="Diversions.csv",LocationColumn="WDID",
DataSource=DWR,DataType=DivTotal,Interval=Month,InputType=HydroBase,
```
Command Reference: ReadUsgsNwisDaily()

Read 1+ time series from the USGS NWIS Daily Value web service

Version 10.12.00, 2012-08-06

The ReadUsgsNwisDaily() command reads one or more time series from the United States Geological Survey (USGS) National Water Information System (NWIS) Daily Value web service (see the **UsgsNwisDaily Data Store Appendix**). The command provides parameters to constrain the web service query and also allows the result to be saved as an output file. For example, if WaterML is chosen as the time series format, a WaterML file can be saved and can be read later using the ReadWaterML() command. See also the WebGet() command, which also can be used to retrieve data files from the USGS website.

The USGS NWIS web service allows station and time series data type information to be filtered, both as a convenience and to maintain reasonable web service performance. Many of the choices that are available for limiting queries allow 0+ values to be provided. For example, specifying no requested parameter will return all available parameters for a location. Specifying a list of parameters (separated by commas) will return only the requested parameters.

USGS codes are used in order to generate a unique time series identifier (TSID). For example, the TSID data type is formed from the parameter code, a dash, and the statistic code. The numerical codes currently are used to ensure uniqueness but in the future the string name may be allowed as an option. In order to have more human-friendly identifiers for time series, one strategy is to request only a specific parameter and statistic and then use the alias to specify a text equivalent to the numeric codes. For example, specify Parameters=00060 (for streamflow discharge) and Statistics=00003 and assign the alias with Alias=%L.Streamflow-Mean.

The following dialog is used to edit the command and illustrates the syntax. Note that some choices are provided as a convenience. However, full listing of choices (such as all the thousands of streamflow stations that are available) is not provided due to performance issues. Additional query features will be enabled as web service integration is enhanced.

Sedit ReadUsgsNwisDaily	Command			
Read one or more time series from	Read one or more time series from the USGS NWIS daily value web service.			
WARNING - This command ca	WARNING - This command can be slow. Constrain the query to improve performance.			
Common choices are provide	d for convenience but may not apply (additional enhance	ements to web services may improve i	ntelligent choices in the future).	
Refer to the USGS NWIS Daily Dat	a Store documentation for more information.			
Constrain the query by specifying	time series metadata to match. A location constraint must be	e specified.		
If not specified, the input period d	efaults to the input period from SetInputPeriod().			
Optionally, also write time series to The working directory in Cultured	o a file, which can be specified using a full or relative path (relative	e to the working directory).		
USGS NWIS Documentation	USGS NWIS Online	(Readosysiawisbaily		
Data store:			Required - data store containing data.	
Location constraint (specify only	one constraint)			
Site number(s): 06752000		List of 1+ site numbers separated by comma	as.	
State(s):		List of 1+ state abbreviations separated by	commas.	
HUC(s):		List of 1+ (1 2-digit and/or up to 10 8-digit)	HUCs separated by commas.	
Bounding box:		Bounding box: WestLon,SouthLat,EastLon,I	NorthLat	
FIPS counties: Select County	/ 💌 =>	List of 1+ counties separated by commas.		
Parameter(s):	Select Parameter	✓ => 00060	Optional - list of parameter codes separated by commas (default=all).	
Statistic(s):	Select Statistic 💙 => 00003		Optional - list of statistic codes separated by commas (default=all).	
Site status:	▼		Optional - site status (default=All).	
Site types(s):			Optional - list of site types separated by commas (default=all).	
Agency:			Optional - agency code (default=all).	
Input start:	2007-01-01		Optional - YYYY-MM-DD, override the global input start.	
Input end:	2007-09-30		Optional - YYYY-MM-DD, override the global input end.	
Alias to assign:	Select Specifier 💉 => %L-%T		Optional - use %L for location, etc. (default=no alias).	
Format:	✓		Optional - data format (default=WaterML).	
Output file to write:	${\it Results/Test_ReadUsgsNwisDaily_SingleSite_Alias_out.waterml}$		Browse	
Command: ReadUsgsNwisDaily(DataStore="UsgsNwisDaily",Sites="06752000",Parameters="00060",Statistics="00003",InputSta rt="2007-01-01",InputEnd="2007-09-30",Alias="%L-%T",OutputFile="Results/Test_ReadUsgsNwisDaily_SingleSite_A lias_out.waterml")				
	Add Working Directory Cancel OK			
			ReadUsosNwisDaily	

ReadUsgsNwisDaily() Command Editor

ReadUsgsNwisDaily(Parameter=Value,...)

Parameter	Description	Default
Sites	A list of site numbers to read, separated	None – one of the locational
	by commas.	parameters must be provided to
		constrain the query.
States	A list of state codes (e.g., AL), separated	None – see above.
	by commas.	
HUCs	A list of hydrologic unit codes, separated	None – see above.
	by commas. See the limitations on the	
	NWIS site for more information.	
BoundingBox	A bounding box consisting of west	None – see above.
	longitude, south latitude, east longitude,	
	and north latitude, separated by spaces.	
	Longitudes in the western hemisphere	
	are negative.	
Counties	A list of Federal Information Processing	None – see above.
	Standards (FIPS) county codes, separated	
	by commas.	
Parameters	Data parameter codes for the stations	All available parameters are
	(e.g., 00060 for stream discharge),	returned.
	separated by commas.	
Statistics	Statistic codes (e.g., 00003 for mean),	All available statistics are
	separated by commas.	returned.
SiteStatus	Filter for stations, one of:	All
	• All – all stations are returned	
	• Active – only active stations are	
	returned	
	• Inactive – only inactive stations	
	are returned	
SiteTypes	Site types to return, separated by	All available site types are
	commas.	returned.
Agency	Agency code to return (e.g., USGS).	All available agencies are
		returned.
InputStart	The start of the period to read data –	Use the global query period.
	specify if the period should be different	
	from the global query period.	
InputEnd	The end of the period to read data –	Use the global query period.
	specify if the period should be different	
	from the global query period.	
Alias	The alias to assign to the time series, as a	None – must be specified.
	literal string or using the special	
	formatting characters listed by the	
	command editor. The alias is a short	
	identifier used by other commands to	
	locate time series for processing, as an	

Parameter	Description	Default
	alternative to the time series identifier (TSID).	
Format	 The data format for output, one of: JSON – JavaScript Object Notation (currently used only for downloads but will not result in time series in TSTool) RDB – tab-delimited format (also see ReadUsgsNwisRDB() command; currently used only for downloads but will not result in time series in TSTool). WaterML – XML format (also see the ReadWaterML() command). 	WaterML
OutputFile	The name of the output file to create. The path to the file can be absolute or relative to the working directory.	No output file will be created.

Command Reference: ReadUsgsNwisGroundwater()

Read 1+ time series from the USGS NWIS groundwater web service

The ReadUsgsNwisGroundwater() command reads one or more time series from the United States Geological Survey (USGS) National Water Information System (NWIS) groundwater web service (see the **UsgsNwisGroundwater Datastore Appendix**). The USGS data are historical manually recorded values and data may be sparse over the full period. The command provides parameters to constrain the web service query and also allows the result to be saved as an output file. For example, if WaterML is chosen as the time series format, a WaterML file can be saved and can be read later using the ReadWaterML() command. See also the WebGet() command, which can be used to retrieve data files from the USGS website.

The USGS NWIS web service allows well and time series data type information to be filtered, both as a convenience and to ensure reasonable web service performance. Many of the choices that are available for limiting queries allow 0+ values to be provided. For example, specifying no requested parameter will return all available parameters for a location. Specifying a list of parameters (separated by commas) will return only the requested parameters.

USGS codes are used in order to generate a unique time series identifier (TSID). For example, the TSID data type is formed from the parameter code. The numerical codes currently are used to ensure uniqueness but in the future the string name may be allowed as an option. In order to have more human-friendly identifiers for time series, one strategy is to request only a specific parameter and then use the alias to specify a text equivalent to the numeric codes. For example, specify Parameters=72019 (for depth to water level) and assign the alias with Alias=%L.WaterLevel.

Although the NWIS groundwater web service may return date/times with precision to minute, this command treats all data as daily values and returns a daily time series. The daily interval time series therefore may have many missing values, but often is easier to process with other TSTool commands. In the future, the command may be updated to allow the option of returning other data intervals, including irregular (which would have only non-missing values but typically must be converted to a regular interval to use with other commands).

The following dialog is used to edit the command and illustrates the syntax. Some choices are provided as a convenience. However, full listing of choices (such as all the thousands of wells that are available) is not provided due to performance issues. Additional query features will be enabled as web service integration is enhanced.

👌 Edit ReadUsgsNwisGrou	ndwater Command		X
Read one or more time series from	the USGS NWIS groundwater (histo	orical manually recorded values) web service.	
WARNING - This command car	n be slow. Constrain the query	to improve performance.	
Daily time series are created	 in the future other data inter 	rvals may be supported.	
Common choices are provided	d for convenience but may not	t apply (additional enhancements to web services may improve intelligent choices in the h	uture).
Constrain the query by specifying	ater Datastore documentation for m time series metadata to match. A k	nore information. An an	
If not specified, the input period d	efaults to the input period from Set.	InputPeriod().	
Optionally, also write time series to	a file, which can be specified using	g a full or relative path (relative to the working directory).	
The working directory is: C:\Develo	op\TSTool_SourceBuild\TSTool\test\	\regression\commands\general\ReadUsgsNwisGroundwater	
USGS NWIS Documentation	USGS NWIS Online		
Data store:	UsgsNwisGroundwater 💌	Required - data store containing data.	
Location constraint (specify only	one constraint)		
Site number(s): 4001251043700	01	List of 1+ site numbers separated by commas.	
State(s):		List of 1+ state abbreviations separated by commas.	
HUC(s):		List of 1+ (1 2-digit and/or up to 10 8-digit) HUCs separated by commas.	
Bounding box:		Bounding box: WestLon,SouthLat,EastLon,NorthLat	
FIPS counties: Select County	/ 💙 =>	List of 1+ counties separated by commas.	
Parameter(s):	Select Parameter		ted by commas (default=all).
Site status:	Active 💌	Optional - site status (default=All).	
Site types(s):		Optional - list of site types separated by	commas (default=all).
Agency:		Optional - agency code (default=all).	
Interval:	Day 🖌	Required - data interval for data.	
Input start:		Optional - YYYY-MM-DD, override the glo	oal input start.
Input end:		Optional - YYYY-MM-DD, override the glo	oal input end.
Alias to assign:	Select Specifier 💌 => %L	L-%T Optional - use %L for location, etc. (defa	ult=no alias).
Format:	×	Optional - data format (default=WaterMi	.),
Output file to write:	Results/Test_ReadUsgsNwisGroun	ndwater_SingleSite_Day_out.waterml	Browse
Command:	ReadUsgsNwisGroundwat SiteStatus="Active",i ingleSite_Day_out.wat	ter(DataStore="UsgsNwisGroundwater",Sites="400125104370001",Para Interval="Day",Alias="%L-%T",OutputFile="Results/Test_ReadUsgsNu term1")	ameters="72019", visGroundwater_S
Add Working Directory Cancel OK			

ReadUsgsNwisGroundwater() Command Editor

ReadUsgsNwisGroundwater(Parameter=Value,...)

Parameter	Description	Default
Sites	A list of site numbers to read, separated	None – one of the locational
	by commas.	parameters must be provided to
		constrain the query.
States	A list of state codes (e.g., AL), separated	None – see above.
	by commas.	
HUCs	A list of hydrologic unit codes, separated	None – see above.
	by commas. See the limitations on the	
	NWIS site for more information.	
BoundingBox	A bounding box consisting of west	None – see above.
	longitude, south latitude, east longitude,	
	and north latitude, separated by spaces.	
	Longitudes in the western hemisphere	
	are negative.	
Counties	A list of Federal Information Processing	None – see above.
	Standards (FIPS) county codes, separated	
	by commas.	
Parameters	Data parameter codes for the stations	All available parameters are
	(e.g., 72019 for depth to water level),	returned.
	separated by commas.	
SiteStatus	Filter for stations, one of:	All
	• All – all stations are returned	
	• Active – only active stations are	
	returned	
	• Inactive – only inactive stations	
	are returned	
SiteTypes	Site types to return, separated by	All available site types are
	commas.	returned.
Agency	Agency code to return (e.g., USGS).	All available agencies are
		returned.
Interval	The interval to use for the created time	None – must be specified.
	series. Groundwater measurements in	
	NWIS may be recorded for the day or	
	may have more precise date/time. Using	
	an interval of Day results in a regular	
	interval time series that is easier to	
	process by other commands, but may not	
	be suitable when values change	
	significantly within a day.	
InputStart	The start of the period to read data –	Use the global query period.
	specify if the period should be different	
	from the global query period.	
InputEnd	The end of the period to read data –	Use the global query period.
	specify if the period should be different	

Parameter	Description	Default
	from the global query period.	
Alias	The alias to assign to the time series, as a	None – must be specified.
	literal string or using the special	
	formatting characters listed by the	
	command editor. The alias is a short	
	identifier used by other commands to	
	locate time series for processing, as an	
	alternative to the time series identifier	
	(TSID).	
Format	The data format for output, one of:	WaterML
	• JSON – JavaScript Object Notation	
	(currently used only for downloads	
	but will not result in time series in	
	TSTool)	
	• RDB – tab-delimited format (also see	
	ReadUsgsNwisRDB() command;	
	currently used only for downloads	
	but will not result in time series in	
	TSTool).	
	• WaterML – XML format (also see	
	the ReadWaterML() command).	
OutputFile	The name of the output file to create.	No output file will be created.
	The path to the file can be absolute or	_
	relative to the working directory.	

Command Reference: ReadUsgsNwisInstantaneous()

Read 1+ time series from the USGS NWIS Instantaneous Values web service Version 10.13.00, 2012-10-30

The ReadUsgsNwisInstantaneous() command reads one or more time series from the United States Geological Survey (USGS) National Water Information System (NWIS) Instantaneous Values web service (see the **UsgsNwisInstantaneous Datastore Appendix**). The command provides parameters to constrain the web service query and also allows the result to be saved as an output file. For example, if WaterML is chosen as the time series format, a WaterML file can be saved and can be read later using the ReadWaterML() command. See also the WebGet() command, which also can be used to retrieve data files from the USGS website.

The USGS NWIS web service allows station and time series data type information to be filtered, both as a convenience and to maintain reasonable web service performance. Many of the choices that are available for limiting queries allow 0+ values to be provided. For example, specifying no requested parameter will return all available parameters for a location. Specifying a list of parameters (separated by commas) will return only the requested parameters.

The data interval for returned time series is set to 15Min. A check is performed to ensure that data line up with this interval. If the data do not line up, values are set by rounding time and warnings will be generated. Another option is to save the time series as a WaterML file and then use ReadWaterML() command, which allows the time series interval to be specified.

USGS codes are used in order to generate a unique time series identifier (TSID). For example, the TSID data type is formed from the parameter code. The numerical codes currently are used to ensure uniqueness but in the future the string name may be allowed as an option. In order to have more human-friendly identifiers for time series, one strategy is to request only a specific parameter and then use the alias to specify a text equivalent to the numeric codes. For example, specify Parameters=00060 (for streamflow discharge) and and assign the alias with Alias=%L.Streamflow.

ReadUsgsNwisInstantaneous

The following dialog is used to edit the command and illustrates the syntax. Note that some choices are provided as a convenience. However, full listing of choices (such as all the thousands of streamflow stations that are available) is not provided due to performance issues. Additional query features will be enabled as web service integration is enhanced.

Edit ReadUsgsNwisInsta	ntaneous Command			¥
Read one or more time series from	the USGS NWIS instantaneous values web service.			
WARNING - This command car Common choices are provide	n be slow. Constrain the query to improve performance I for convenience but may not apply (additional enhan	coments to meh services may improve intelligent ch	nices in the future)	
Refer to the USGS NWIS Instantar	neous Values Datastore documentation for more information.	cements to web services may improve intelligent ch	bices in the facare).	
Constrain the guery by specifying	time series metadata to match. A location constraint must l	be specified.		
If not specified, the input period d	efaults to the input period from SetInputPeriod().			
Optionally, also write time series to	a file, which can be specified using a full or relative path (relativ	ve to the working directory).		
The working directory is: C:\Develo	pp/TSTool_SourceBuild\TSTool\test\regression\commands\gener	al\ReadUsgsNwisInstantaneous		
USGS NWIS Documentation	USGS NWIS Online			
Data store:	UsgsNwisInstantaneous 💌		Required - data store contai	ining data.
Location constraint (specify only	one constraint)	_		
Site number(s): 06752260		List of 1+ site numbers separated by commas.		
State(s):		List of 1+ state abbreviations separated by commas.		
HUC(s):		List of 1+ (1 2-digit and/or up to 10 8-digit) HUCs separate	d by commas.	
Bounding box:		Bounding box: WestLon,SouthLat,EastLon,NorthLat		
FIPS counties: Select County	💌 =>	List of 1+ counties separated by commas.		
Parameter(s):	Select Parameter	✓ => 00060	Optional - list of parameter (codes separated by commas (default=all
Site status:	 Image: A start of the start of		Optional - site status (defau	ilt=All).
Site types(s):			Optional - list of site types s	eparated by commas (default=all).
Agency:			Optional - agency code (def	ault=all).
Input start:	2010-01-01		Optional - YYYY-MM-DD, ove	erride the global input start.
Input end:	2010-03-15		Optional - YYYY-MM-DD, ove	erride the global input end.
Alias to assign:	Select Specifier 💌 => %L.%T		Optional - use %L for location	on, etc. (default=no alias).
Format:	×		Optional - data format (defa	ault=WaterML).
Output file to write:	${\tt Results/Test_ReadUsgsNwisInstantaneous_SingleSite_Alias_output the test_stantaneous_SingleSite_Alias_output test_stantaneous_SingleSite_Alias_output test_stantaneous_Site_Alias_output test_stantaneous_Site_Alias_output test_stantaneous_Site_Alias_output test_stantaneous_Site_Alias_output test_stantaneous_Site_Alias_output test_stantaneous_Site_Alias_output test_stantaneous_Site_Alias_Site_Alias_Site_Alias_Site_Alias_Site_Alias_Site_Alias_$	ıt.watermi]	Browse
Command: ReadUsgsNwisInstantaneous(DataStore="UsgsNwisInstantaneous",Sites="06752260",Parameters="00060",InputStart="2010-01- 01",InputEnd="2010-03-15",Alias="%L.%T",OutputFile="Results/Test_ReadUsgsNwisInstantaneous_SingleSite_Alias_out.wate rml")				
	Add W	orking Directory Cancel OK		

ReadUsgsNwisInstantaneous() Command Editor

ReadUsgsNwisInstantaneous(Parameter=Value,...)

Parameter	Description	Default
Sites	A list of site numbers to read, separated	None – one of the locational
	by commas.	parameters must be provided to
		constrain the query.
States	A list of state codes (e.g., AL), separated	None – see above.
	by commas.	
HUCs	A list of hydrologic unit codes, separated	None – see above.
	by commas. See the limitations on the	
	NWIS site for more information.	
BoundingBox	A bounding box consisting of west	None – see above.
	longitude, south latitude, east longitude,	
	and north latitude, separated by spaces.	
	Longitudes in the western hemisphere	
	are negative.	
Counties	A list of Federal Information Processing	None – see above.
	Standards (FIPS) county codes, separated	
	by commas.	
Parameters	Data parameter codes for the stations	All available parameters for the
	(e.g., 00060 for stream discharge),	sites are returned.
	separated by commas.	
SiteStatus	Filter for stations, one of:	All
	• All – all stations are returned	
	• Active – only active stations are	
	returned	
	• Inactive – only inactive stations	
	are returned	
SiteTypes	Site types to return, separated by	All available site types are
	commas.	returned.
Agency	Agency code to return (e.g., USGS).	All available agencies are
		returned.
InputStart	The start of the period to read data to 15-	Use the global query period.
	minute precision – specify if the period	
	should be different from the global query	
	period.	
InputEnd	The end of the period to read data to 15-	Use the global query period.
	minute precision – specify if the period	
	should be different from the global query	
	period.	
Alias	The alias to assign to the time series, as a	None – must be specified.
	literal string or using the special	
	formatting characters listed by the	
	command editor. The alias is a short	
	identifier used by other commands to	
	locate time series for processing, as an	

Parameter	Description	Default
	alternative to the time series identifier (TSID).	
Format	 The data format for output, one of: JSON – JavaScript Object Notation (currently used only for downloads but will not result in time series in TSTool) RDB – tab-delimited format (also see ReadUsgsNwisRDB() command; currently used only for downloads but will not result in time series in TSTool). WaterML – XML format (also see the ReadWaterML() command). 	WaterML
OutputFile	The name of the output file to create. The path to the file can be absolute or relative to the working directory.	No output file will be created.

Command Reference: ReadUsgsNwisRdb()

Read a single time series from a USGS NWIS RDB file

Version 10.05.00, 2012-02-27

The ReadUsgsNwisRdb() command reads a single time series from a USGS NWIS RDB file (see the UsgsNwisRdb Input Type Appendix) and assigns an alias to the result. This command replaces the older ReadUsgsNwis() command – legacy ReadUsgsNwis() commands are automatically translated to ReadUsgsNwisRdb() commands. Currently only the daily streamflow format is supported and the file being read must contain only one time series. The data type is assigned as Streamflow, with units CFS. See also the WebGet() command, which can be used to retrieve data files from the USGS website. See also the ReadUsgsNwisDaily() and ReadWaterML() commands.

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

👌 Edit ReadUsgsNwisRdb() Command 🛛 🛛 🔀				×
Read a single time series from	tead a single time series from a USGS NWIS RDB format file. Currently only daily streamflow is supported.			
Specify a full path or relative	path (relative to working directory)	for a USGS NWIS RDB f	ile to read.	
Specifying the period will limit	data that are available for fill comm	hands but can increase p	erformance.	
If not specified, the period de	faults to the global input period, o	r read all.		
The working directory is: C:\D	evelop\TSTool_SourceBuild\TSTool	\test\regression\commar	nds\general\ReadUsgsNwisRdb	
USGS NWIS RDB file to read:	Data\G03451500.txt			Browse
Alias to assign:	nwis Insert:	Select Specifier 💊	Required - use %L for location, etc.	
Input start:			Optional - overrides the global input start.	
Input end:			Optional - overrides the global input end.	
	ReadUsgsNwisRdb(Inpu	tFile="Data\GO3	451500.txt",Alias="nwis")	
Command:				
	Add Working Directory Cancel OK			
			B	eadUsosNwisRdb

ReadUsgsNwisRdb() Command Editor

ReadUsgsNwisRdb(Parameter=Value,...)

The following legacy command syntax is updated to the above syntax when a command file is read:

TS Alias = ReadUsgsNwis (Parameter=Value,...)

Command Parameters

Parameter	Description	Default
InputFile	The name of the USGS NWIS RDB file	None – must be specified.
	to read, surrounded by double quotes.	
	The path to the file can be absolute or	
	relative to the working directory.	
Alias	The alias to assign to the time series, as a	None – must be specified.
	literal string or using the special	
	formatting characters listed by the	
	command editor. The alias is a short	
	identifier used by other commands to	
	locate time series for processing, as an	
	alternative to the time series identifier	
	(TSID).	
InputStart	The start of the period to read data –	Use the global query period.
	specify if the period should be different	
	from the global query period.	
InputEnd	The end of the period to read data –	Use the global query period.
	specify if the period should be different	
	from the global query period.	

A sample command file is as follows:

ReadUsgsNwisRdb(Alias="ts1",InputFile="G03451500.txt")

Command Reference: ReadWaterML()

Read 1+ time series from a WaterML file

Version 10.13.00, 2012-10-31

The ReadWaterML() command reads one or more time series from a WaterML XML time series file (see the **WaterML Input Type Appendix**). WaterML version 1.1 is supported. WaterML files can be created using the ReadUsgsNwisDaily(), ReadWaterOneFlow(), and WriteWaterML() commands, and can be saved from web sites that provide WaterML using the WebGet() command. This command may be enhanced in the future to read a subset of the time series in the WatermL file (currently all time series in the file are read), and additional WaterML versions may be supported.

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

🕈 Edit ReadWaterML Command				
Read all the time series from a W	ead all the time series from a WaterML file, using information in the file to assign the time series identifier.			
Specify a full path or relative pat	h (relative to the working directory) fo	for a WaterML file to read.		
The working directory is: C:\Deve	elop\TSTool_SourceBuild\TSTool\test\r	\regression\commands\general\ReadWaterML		
Specifying the input period will lin	it data that are available for fill comm	mands but can increase performance.		
WaterML file to read:	Data\03451500_SingleSite_UsgsNwis	visDaily.waterm	rowse	
Alias to assign:	Select Specifier 💉 => %L.%	%T Optional - use %L for location, etc.		
Interval:	Day 🖌	Required - data interval for data.		
Require data to match interval?:	~	Optional - require data/interval alignment (default=True).		
Input start:		Optional - overrides the global input start.		
Input end:		Optional - overrides the global input end.		
Command:	ReadWaterML(InputFile="Data\03451500_SingleSite_UsgsNwisDaily.waterml", Alias="%L.%T",Interval="Day")			
Add Working Directory Cancel OK				

ReadWaterML() Command Editor

ReadWaterML(Parameter=Value,...)

Parameter	Description	Default
InputFile	The name of the WaterML file to read.	None – must be specified.
	The path to the file can be absolute or	
	relative to the working directory.	
Alias	The alias to assign to the time series, as a	No alias is assigned.
	literal string or using the special	_
	formatting characters listed by the	
	command editor. The alias is a short	
	identifier used by other commands to	
	locate time series for processing, as an	
	alternative to the time series identifier	
	(TSID).	
Interval	The data interval for the file, necessary	None – must be specified.
	because WaterML 1.1 does not have a	
	data element indicating the interval (time	
	step for the data) and using irregular by	
	default would be inefficient for data	
	management. This issue is being further	
	evaluated.	
RequireData	Indicate whether the date/time for each	True
ToMatchInterval	data value must align with the interval:	
	• True – For example, if	Parameter is not used for
	Interval=15Min for USGS	irregular data.
	instantaneous data, then values a	
	warning will be generated.	
	• False – Date/times that do not	
	align result in time series values	
	being assigned using a truncated	
	date/time. For example, USGS	
	groundwater web service values read	
	with Interval=Day will be	
	assigned to the nearest day (by	
	ignoring more precise time	
	information).	
	This parameter and the Interval	
	parameter will continue to be evaluated.	
InputStart	The start of the period to read data –	Use the global query period.
	specify if the period should be different	
	from the global query period.	
InputEnd	The end of the period to read data –	Use the global query period.
	specify if the period should be different	
	from the global query period.	

Command Reference: ReadWaterOneFlow()

Read 1+ time series from a WaterOneFlow web service

Version 10.06.00, 2012-03-29

This command is under development.

The ReadWaterOneFlow() command reads one or more time series from WaterOneFlow web service (see the **WaterOneFlow Data Store Appendix**) and optionally assigns an alias to the time series. WaterML version 1.0 is supported for time series transfer; however, the WaterML response currently cannot be saved to a file (and therefore output cannot be used with the ReadWaterML() command).

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

Need to generate screen shot.

ReadWtaerOneFlow

ReadWaterOneFlow() Command Editor

ReadWaterOneFlow(Parameter=Value,...)

Parameter	Description	Default
Alias	The alias to assign to the time series, as a literal string or using the special formatting characters listed by the command editor. The alias is a short identifier used by other commands to locate time series for processing, as an alternative to the time series identifier (TSID).	No alias is assigned.
InputStart	The start of the period to read data – specify if the period should be different from the global query period.	Use the global query period.
InputEnd	The end of the period to read data – specify if the period should be different from the global query period.	Use the global query period.

Command Reference: RelativeDiff()

Create a relative difference time series Version 10.00.01, 2011-05-15

A RelativeDiff() command creates a new relative difference time series, computed by subtracting the time series and then dividing by one of the time series. This is useful when analyzing the relative magnitudes of two time series over time. Most of the properties for the new time series are the same as the first time series. The alias for the result can be referenced by other commands. The divisor can be either of the time series. The result is set to missing if either time series value is missing or the divisor is zero.

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

👌 Edit Relative	Diff() Command		X
Create a new unitles	s time series with data valu	Jes:	
TS = (TS1 - TS2)/I	Divisor		
where the Divisor is	either TS1 or TS2.		
Attempting to divide	by zero sets the data point	t to the missing value.	
The metadata for th	e result is a copy of TS1 ex	cept for the alias.	
Time series 1 (TS1):	DelNorte		×
Time series 2 (TS2):	Alamosa		×
Divisor:	DivideByTS1 💌		Required - specify time series for divisor.
Alias to assign:	RelativeDiff	Insert: Select Specifier	Required - use %L for location, etc.
Command:	RelativeDiff(TS] eByTS1,Alias="Re	[D1="DelNorte",TSI elativeDiff")	D2="Alamosa",Divisor=Divid
Cancel OK			

RelativeDiff() Command Editor

```
RelativeDiff(Parameter=Value,...)
```

The following older command syntax is updated to the above syntax when a command file is read:

TS Alias = RelativeDiff(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSID1	The time series identifier or alias for the first time series.	None – must be
		specified.
TSID2	The time series identifier or alias for the second time series	None – must be
	(subtracted from the first).	specified.
Divisor	Indicates whether the first time series is the divisor	None – must be
	(DivideByTS1) or the second time series is the divisor	specified.
	(DivideByTS2).	
Alias	The alias to assign to the time series, as a literal string or using	None – must be
	the special formatting characters listed by the command editor.	specified.
	The alias is a short identifier used by other commands to	
	locate time series for processing, as an alternative to the time	
	series identifier (TSID).	

A sample command file to process data from the State of Colorado's HydroBase database is as follows:



The input time series for the command are shown in the following figure:

Data for the RelativeDiff() Command



The results of processing the commands are shown in the following figure:

Results of the RelativeDiff() Command

Command Reference: RemoveFile()

Remove a file Version 09.02.00, 2009-04-03

The RemoveFile() command removes a file from the file system. This command is used in testing software to remove results files before attempting to regenerate the results.

A failure will be generated if the file exists and cannot be removed (e.g., due to file permissions or being locked by another process).

Even read-only files may be removed by this command, depending on how the operating system and computer environment handle access permissions.

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

🛃 Edit RemoveFile() command 🛛 🔀			
This command r	removes a file. The file to be removed does not need to exist when editing this command.		
It is recommend	led that the file name be relative to the working directory, which is:		
C:\Develop\T	STool_SourceBuild\TSTool%test%regression%UserManualExamples\TestCases%CommandReferenc	e\RemoveFile	
File to remove:	Results/output.dv	Browse	
If not found?:	Optional - action if file not found (default=lgnore)		
	RemoveFile(InputFile="Results/output.dv")		
Command:			
	Add vvorking Directory Cancel OK		
		RemoveFi	

RemoveFile() Command Editor

RemoveFile(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
InputFile	The name of the file to delete.	None – must be specified.
IfNotFound	 Indicate action if the file is not found, one of: Ignore - ignore the missing file (do not warn). Warn - generate a warning (use this if the file truly is expected and a missing file is a cause for concern). Fail - generate a failure (use this if the file truly is expected and a 	Ignore
	missing file is a cause for concern).	

The following example command file illustrates how to remove a file:

RemoveFile(InputFile="Results/output.dv")

Command Reference: RemoveTableRowsFromDataStore()

Remove rows from a datastore table

Version 10.18.00, 2013-02-26

The RemoveTableRowsFromDataStore() removes rows from a database datastore table by executing an SQL DELETE statement. If database datastore support is not specifically provided by TSTool, a generic datastore can be used (see the **Generic Database DataStore** appendix). This command cannot be used with web service datastores and use with Excel datastores has not been tested. This command is useful in particular for bulk data processing such as to remove records in a table before (re)loading in bulk (see WriteTableToDataStore() command).

General constraints are as follows:

- the table or views being processed must be writeable by the user specified for the database connection (some databases restrict direct access to data and/or require using stored procedures)
- currently, only the ability to delete all rows is supported (see RemoveAllRows command parameter); in the future functionality will be implemented to delete rows matching rows in a TSTool table

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

👌 Edit RemoveTableRowsFromDataStore() Command					
This command remove:	This command removes rows from a datastore table.				
Currently the Remove/	AllRows parameter is the only way to control what is removed.				
Datastore:	INSIGHT-FABAnalysis-2012	Required - database datastore to receive data.			
Datastore table/view:	InsightTimeSeriesData_Year	 Required - database table/view to receive data. 			
Remove all rows?:	True 💌	Required - remove all rows? (default=False).			
	RemoveTableRowsFromDataStore(DataStore="INSIG	GHT-FABAnalysis-2012",DataStor			
	eTable="InsightTimeSeriesData_Year",RemoveAl.	lRows="True")			
Command:					
Cancel OK					
		RemoveTableRowsFromDataStore			

RemoveTableRowsFromDataStore() Command

RemoveTableRowsFromDataStore(Parameter=Value,...)

Parameter	Description	Default
DataStore	The name of a database datastore to process.	None – must be specified.
DataStoreTable	The name of the database table or view	None – must be specified.
	being processed.	
RemoveAllRows	Indicate whether all rows should be removed	False – only rows
	(True) or only a subset (False). False	matching TSTool table are
	is the default as a safeguard and future	removed.
	enhancements will enable removing only	
	rows that match TSTool table rows or a	
	constraint.	

Command Reference: ReplaceValue()

Replace time series data value(s)

Version 10.06.00, 2012-03-22

The ReplaceValue() command replaces a range of values in a time series with a constant value, sets the values to missing, or removes the values (if an irregular time series). If the missing value indicator is a number in the range, missing values also will be replaced. The time series data flag can be checked in place of or addition to checking the numerical values. This command is useful for filtering out erroneous data values. See also the CheckTimeSeries() command, which provides for a variety of checks and also allows values to be set to missing or removed.

The following dialog is used to edit the command and illustrates the syntax of the command:

👌 Edit ReplaceValue() Comm	and	
Replace a single data value or range o	f data values with a constant.	
The data values and/or flags are matc	hed to determine values to replace.	
Optionally, set missing or remove the v	values entirely (if an irregular interval time series).	
If the missing value indicator is a numb	er in the given range, missing values also will be replaced.	
Specify dates with precision appropria	te for the data.	
TS list:	AllMatchingTSID	Optional - indicates the time series to process (default=AllTS).
TSID (for TSList=AllMatchingTSID):	08235700.DWR.Streamflow.Month	×
EnsembleID (for TSList=EnsembleID):		
Minimum value to replace:	-100000	Optional (maximum value also can be specified).
Maximum value to replace:	0	Optional - use when specifying range.
Flag to match for replace:		Optional - use when matching flags.
Constant value to replace with:	0	Required - or specify action.
Action:	×	Optional - action for matched values (default=just replace).
Replacement start:		Optional - start of replacement (default is all).
Replacement end:		Optional - end of replacement (default is all).
	[Start] [End]	
🔄 Analysis window:	Month: Day: Hour: Minute: Month: Day: Hour: Minute:	Optional - analysis window within each year (default=full year).
Set flag:		Optional - string to mark replaced data.
	ReplaceValue(TSList=AllMatchingTSID.TSID="08235700.DWR.Stre	amflow.Month".MinValue=-100000.MaxVal
	ue=0.NewValue=0)	
Command:		
	Cancel	
		ReplaceValue

ReplaceValue() Command Editor

```
ReplaceValue(Parameter=Value,...)
```

Parameter	Description	Default		
TSList	Indicates the list of time series to be processed,	Allts		
	one of:			
	• AllMatchingTSID – all time series that			
	match the TSID (single TSID or TSID with			
	wildcards) will be processed.			
	• AllTS – all time series before the command.			
	• EnsembleID – all time series in the			
	ensemble will be processed.			
	• FirstMatchingTSID – the first time			
	series that matches the TSID (single TSID or			
	TSID with wildcards) will be processed.			
	 LastMatchingTSID – the last time series 			
	that matches the TSID (single TSID or TSID)			
	with wildcards) will be processed			
	 SelectedTS – the time series are those 			
	selected with the Select TimeSeries ()			
	command.			
TSID	The time series identifier or alias for the time	Required if		
	series to be processed, using the * wildcard	TSList=*TSID.		
	character to match multiple time series.	1220 12_21		
EnsembleID	The ensemble to be processed, if processing an	Required if		
	ensemble.	TSList=EnsembleID.		
MinValue	The minimum value to replace.	The minimum value		
		and/or MatchFlag must		
		be specified.		
MaxValue	The maximum value to replace.	If not specified, only data		
		values that exactly match		
		the minimum value will be		
		replaced.		
MatchFlag	The flag to match. If specified in addition to	The minimum value		
	MinValue, then the value and flag must be	and/or MatchFlag must		
	matched in order to perform the replacement. A	be specified.		
	case-sensitive comparison is made and the data			
	value flag must exactly match MatchFlag. In			
	the future additional flexibility may be added to			
	match a substring, etc.			
	If Action=SetMissing. the original data flag			
	value will remain. Specifying SetFlag will			
	result in the original data flag being modified.			
NewValue	The new data value.	Required, unless the		
		Action parameter is		

Parameter	Description	Default
		specified.
Action	An additional action to take with values that are	No additional action is
	matched:	taken and the NewValue
	• Remove – remove the data points. This can	parameter must be
	only be specified for irregular interval time	specified.
	series and will be interpreted as	
	SetMissing for regular interval time	
	series.	
	 SetMissing – set values to missing. 	
SetStart	The date/time to start filling, if other than the full	Check the full period.
	time series period.	
SetEnd	The date/time to end filling, if other than the full	Check the full period.
	time series period.	
AnalysisWindow	The starting date/time within the calendar year to	Process each full year.
Start	replace data. The window CANNOT cross	
	calendar year boundaries (this may be allowed in	
	the future). Use multiple commands if necessary.	
AnalysisWindow	The ending date/time within the calendar year to	Process each full year.
End	replace data.	
SetFlag	A string to assign to data values that are replaced.	Do not assign a string flag.

A sample command file to process from the State of Colorado's HydroBase database is as follows:

08235700 - ALAMOSA RIVER BELOW CASTLEMAN GULCH NEAR JASPER 08235700.DWR.Streamflow.Month~HydroBase ReplaceValue(TSList=AllTS,MinValue=-100000,MaxValue=0,NewValue=0) This page is intentionally blank.

Command Reference: ResequenceTimeSeriesData()

Resequence time series data (shuffle years of data)

The ResequenceTimeSeriesData() command resequences data in time series by shifting/shuffling/repeating values from one year to another, creating new time series for each time series. For example, January 1950 might be shifted to January 1990. This command is useful for generating synthetic time series by resequencing historical data. The following constraints apply to the command as currently implemented:

- 1. Processing by default occurs by calendar year, with the sequence specified as calendar years. If an alternate output year type is used (see the OutputYearType parameter). The OutputStart year is considered to be consistent with the output year type.
- 2. The sequence of years must currently be supplied as a column of years in a table (rows of years may be added in a future enhancement).
- 3. Full start and end years are required, matching the output year type.
- 4. Currently the command can only be applied to month interval data. For a daily data interval, several technical issues must be resolved before the feature can be implemented:
 - a. If a short year (i.e., non-leap year with 365 days) is transferred to a long year (i.e., a leap year with 366 days), the first day after the short year is used for the 366th day during the transfer. What to do if the year being transferred is the last in the data set and no more years are available for the 366th day repeat the last day?
 - b. If a long year (i.e., leap year with 366 days) is transferred to a short year (i.e., a non-leap year with 365 days), the 366th day in the leap year is not transferred.
- 5. The original period is by default retained in the output time series. For example, if the original data are 1937 to 1997, the resequenced data will also be in a time series with a period 1937 to 1997. The OutputStart parameter can be used to shift the start year of output.

The command is designed to work with a table that provides sequence information. For example, see the ReadTableFromDelimitedFile() command and the example shown below.

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

🕥 Edit ResequenceTimeSeriesData() Command 🛛 🛛 🔀					
Resequence time series data by "shuffling" th An identifier for the table with the new year s	Resequence time series data by "shuffling" the original years of data, creating new time series. An identifier for the table with the new year sequence must be specified, and each sequence of years must be provided in a column with a unique column heading.				
The resulting time series will start in the indica	ted year and hav	e a time series id	lentifier that is the same	e as the original time series, additionally with the indicated scenario.	
If not specified, the output start is taken from	n the global outpu	t start or the tim	ie series.	Optional - indicates the time series to process (default=AllTS)	
				optional - indicates the time series to process (default—Ain 5).	
TSLD (FOR TSLISC=AllMatchingTSLD):					
EnsembleID (for TSList=EnsembleID):				<u>```</u>	
Table ID for year sequence:	KNN_Seq			¥	
Column name in table for year sequence:	1			Required - column name for year sequence.	
New scenario:	KNN01			Required - for TSID of new time series.	
First row number in table for year sequence:	1			Optional - default is first row in column.	
Last row number in table for year sequence:	30			Optional - default is last row in column.	
Output year type:	~			Optional - output year type (default=Calendar).	
Output start:	1908]	Optional - starting year of resequenced time series, consistent with output year type.	
Alias to assign:	%L-KNN01	Insert:	Select Specifier	✓ Optional - use %L for location, etc. (default=no alias).	
	Resequent	ter a combinatio	n of literal strings and/o	or format specifiers from the list on the right. Column="1", TableRowStart=	
Commandi	"1",TableR	owEnd="30"	,OutputStart="	"1908",NewScenario="KNN01",Alias="%L-KNN01")	
Command.					
	Cancel OK				
				Deserving Time Order Date	

ResequenceTimeSeriesData() Command Editor

The command syntax is as follows:

```
ResequenceTimeSeriesData(Parameter=Value,...)
```

Parameter	Description	Default
TSList	 Indicates the list of time series to be processed, one of: AllMatchingTSID - all time series that match the TSID (single TSID or TSID with wildcards) will be processed. AllTS - all time series before the command will be processed. EnsembleID - all time series in the ensemble will be processed. FirstMatchingTSID - the first time series that matches the TSID will be processed. LastMatchingTSID - the last time series that matches the TSID (single TSID or TSID with wildcards) will be processed. SelectedTS - the time series selected with the SelectTimeSeries() command will be processed. 	AllTS
TSID	The time series identifier or alias for the time series to be modified, using the * wildcard character to match multiple time series.	Required when TSList=*TSID.
EnsembleID	The ensemble to be modified, if processing an ensemble.	Required when TSList=

Parameter	Description	Default
		EnsembleID.
TableID	The identifier for the sequence table to use, which	None – must be
	indicates the dates to use when resequencing data (e.g.,	specified.
	list of years for data sequence). For example, see the	
	ReadTableFromDelimitedFile() command.	
	The years should be consistent with the	
	OutputYearType.	
TableColumn	The column name containing the sequence	None – must be
	information. Note that the input table must have	specified.
	column names in a header record.	
TableRowStart	The first data row number (1+) containing the first year	Use all rows.
	in the new sequence.	
TableRowEnd	The last data row number (1+) containing the first year	Use all rows.
	in the new sequence.	
OutputYearType	The output year type, indicating the year extent for the	Calendar
	resequencing, one of:	
	Calendar – January to December	
	• NovToDec – November of previous calendar year	
	to October of current year.	
	• Water – October of previous calendar year to	
	September of current year.	
OutputStart	The output start as a four-digit year that is consistent	Same as the original
	with OutputYearType. For example, if processing	input data or use the
	water years, the OutputStart would be the first	global output start if
	water year in the output (and start in October of the	specified. The output
	previous calendar year). The output end is relative to	months will be
	the output start and includes the number of years in the	adjusted for the
	sequence.	output year type.
NewScenario	The new scenario to assign to the created time series,	Not specified, but a
	resulting in a unique TSID.	new scenario and/or
		alias must be
		specified.
Alias	Alias to assign to the output time series. See the	Not specified, but a
	LegendFormat property described in the TSView	new scenario and/or
	Time Series Viewing Tools appendix. For example,	alias must be
	%L is full location, %T is data type, %I is interval, and	specified.
	βZ is scenario.	

The following example:

- 1. Reads a list of time series from a StateMod model file.
- 2. Reads a sequence of years from a delimited file.
- 3. Resequences the StateMod time series data.
- 4. Writes the resequenced file to a new StateMod file.

```
# Read all demand time series...
ReadStateMod(InputFile="..\StateMod\gunnC2005.xbm")
# Read the sequence of years to use...
Table 0001HK0101 = ReadTableFromDelimitedFile(InputFile="0001HK0101.csv")
# Resequence the StateMod time series...
ResequenceTimeSeriesData(TSList=AllTS,TableID="0001HK0101",
TableColumn="Trace1",NewScenario="KNN0101",Alias="%L-KNN0101")
# Write the resequenced data for StateMod
WriteStateMod(TSList=AllMatchingTSID,TSID="*KNN*",
OutputFile="..\StateMod0101\gunnC2005.xbm")
```

The year sequence is specified in a file similar to the following.

```
# Some comments
"Trace1","Trace2",...
1905,1967,...
1920,1943,...
etc.
```

Variations on the example can be implemented, for example, to process output products after the run.

Command Reference: RunCommands()

Run a command file

The RunCommands () command runs a command file using a separate command processor as a "child" of the main processor. This command can be used to manage workflow where multiple commands files are run, and is also used extensively for testing, where a test suite consists of running separate test case command files.

Command files that are run can themselves include RunCommands () commands. Each command file that is run has knowledge if its initial working directory and relative paths referenced in the command file are relative to this directory. This allows a master command file to reside in a different location than the individual command files that are being run. The current working directory is reset to that of the command file being run.

Data stores from the parent command processor are by default passed to the child command processor. Consequently, database connections can be opened once and shared between command files.

Currently the properties from the parent command file are NOT applied to the initial conditions when running the command file. Therefore, global properties like input and output period are reset to defaults before running the command file. A future enhancement may implement a command parameter to indicate whether to share the properties with the parent processor. The output from the command is also not added to the parent processor. Again, a future enhancement may be to append output so that one final set of output is generated.

There is currently no special handling of log files; consequently, if the main command file opens a log file and then a command file is run that opens a new log file, the main log file will be closed. This behavior is being evaluated.

TTL - C - 11 ' ' - 1 '	- $ +$ $+$ $+$ $+$ $+$ $+$ $+$ $+$ $+$ $+$
The following dialog is lised to edif the command and ill	listrates the syntax for the command
The following dialog is used to call the communa and m	usuales are syntax for the communa.

🜌 Edit RunCommands	s() Command	$\mathbf{\times}$		
Read a command file and run the commands using a separate command processor.				
Parent command processor data stores can be shared with the processor for the command file.				
Results are cleared before processing each command file.				
The (success/warning/failure) status from the command file is used for the RunCommands() command status.				
Specify a full or relative path (relative to working directory).				
The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\RunCommands				
Command file to run:	\\commands\general\ReadStateMod\Test_ReadStateMod_1.TSTool Browse			
Expected status:	Optional - use for testing (default=Success).			
Share parent data stores?:	Optional - share data stores (default=Share).			
Command:	RunCommands(InputFile="\\commands\general\ReadStateMod\Test_Read StateMod_1.TSTool")	d		
Add Working Directory Cancel OK				
	RunComr	mands		

RunCommands() Command Editor

RunCommands(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
InputFile	The name of the command file to run, enclosed in double	None – must be
	quotes if the file contains spaces or other special	specified.
	characters. A path relative to the master command file	
	can be specified.	
ExpectedStatus	Used for testing – indicates the expected status from the	Success
	command, one of:	
	• Unknown	
	• Success	
	• Warning	
	• Failure	
ShareDataStores	Indicate whether data stores in the parent should be	Share
	shared with the child command processor. Normally this	
	should be done so that databases can be opened once.	
	Note that opening data stores in the child command file	
	will not make the data stores available in the parent.	

The following example illustrates how the RunCommands () command can be used to test TSTool software (or any implementation of commands). First, individual command files are implemented to test specific functionality, which will result in warnings if a test fails:

```
StartLog(LogFile="Results/Test_ReadStateMod_1.TSTool.log")
NewPatternTimeSeries(NewTSID="MyLoc..MyData.Day",Alias="TS",
    Description="Test data",SetStart="1950-01-01",
    SetEnd="1951-03-12",Units="CFS",PatternValues="5,10,12,13,75")
# Uncomment the following command to regenerate the expected results file.
# WriteStateMod(TSList=AllTS,
# OutputFile="ExpectedResults\Test_ReadStateMod_1_out.stm")
ReadStateMod(InputFile="ExpectedResults\Test_ReadStateMod_1_out.stm")
CompareTimeSeries(Precision=3,Tolerance=".001",DiffFlag="X",
    WarnIfDifferent=True)
```

Next, use the RunCommands () command to run one or more tests:

```
StartRegressionTestResultsReport(
    OutputFile="RunRegressionTest_commands_general.TSTool.out.txt")
...
RunCommands(InputFile="..\..\commands\general\ReadStateMod\Test_ReadStateMod_1.TSTool")
...
```

Each of the above command files should produce expected results, without warnings. If any command file unexpectedly produces a warning, a warning will also be visible in TSTool. The issue can then be evaluated to determine whether a software or configuration change is necessary. See the StartRegressionTestResultsReport() command documentation for an explanation of how to disable a command file with #@enabled False or indicate its expected status for testing (e.g., @#expected Status Warning).
Command Reference: RunDSSUTL()

Run the DSSUTL and other utility programs from the US Army Corps of Engineers

Version 09.03.00, 2009-04-10

The RunDSSUTL() command runs the Army Corps of Engineers' DSSUTL program and other utility programs, which are used with HEC-DSS files. See also the **HEC-DSS Input Type** appendix. This command formats the command line for the program, runs the program, and checks the exit value. A non-zero exit value will result in a failure status for the command.

TSTool internally maintains a working directory that is used to convert relative paths to absolute paths in order to locate files. The working directory is by default the location of the last command file that was opened. The location of the program being run (e.g., *DSSUTL.EXE*) is determined by the operating system using the PATH environment variable; therefore, use the $\{WorkingDir\}$ property in the command line if the program location is not in PATH. Use " in the command line or arguments to surround whitespace.

It is not clear whether DSSUTL and other program have limits on path or filename length, but if this appears to be the case, use shorter names. If a program is not provided with correct input, it may go into interactive mode, in which case TSTool may appear to stop when running the command. Currently there is no way to kill the process and TSTool must be stopped and restarted.

The following table summarizes how the command treats input for various utility programs. Required arguments are for the RunDSSUTL() command but may be optional if the program is run on the command line.

Progam	Description	DSSFILE=	INPUT=	OUTPUT=
		Argument	Argument	Argument
DSSUTL	Data Storage System Utility Program	Required	Required	Optional
DSPLAY	Data Storage System Graphics Utility	Required	Required	Optional
DSSMATH	Utility Program for Mathematical	Not used – use	Required	Optional
	Manipulation of HEC-DSS Data	OPEN()	_	_
		command.		
REPGEN	Report Generator – not fully			
	supported due to different			
	command line argument			
	conventions.			
DSSTS	Regular Interval Time-Series Data	Required	Required	Optional
	Entry Program			
DSSITS	Irregular Interval Time-Series Data	Required	Required	Optional
	Entry Program			
DSSPD	Paired Data Entry Program	Required	Required	Optional
DSSTXT	Text Data Entry Program	Required	Required	Optional
DWINDO	Interactive Data Entry and Editing	This interactive p	rogram is not s	upported by
		RunDSSUTL()	command.	
WATDSS	Watstore to DSS Data Entry Program	Required	Required	Optional
	– not fully supported due to			
	different command line argument			

RunDSSUTL() Command Handling of HEC-DSS Utility Program Input

Progam	Description	DSSFILE=	INPUT=	OUTPUT=
		Argument	Argument	Argument
	conventions.			
NWSDSS	National Weather Service to Data	Required	Required	Optional
	Storage System Conversion Utility –			
	not fully supported due to different			
	command line argument			
	conventions.			
PREAD	Functions, Macros, and Screens – not			
	fully supported due to interactive			
	prompts.			

The following dialog is used to edit the command and illustrates the command syntax. Note that the *DSSUTL.EXE* location is in this case not included in the PATH environment variable and is specified with the DssutlProgram parameter, using \${WorkingDir}. The HEC-DSS and input files are relative to the working directory.

🔯 Edit RunDSSUTL() Command	×
This command run runs the DSSUTL program, developed by the US Army Corps of Engineers.	
DSSUTL processes data stored in a HEC-DSS binary database file, using DSSUTL commands in the input file.	
The DSSUTL program location must be in the PATH environment variable, specified using an absolutepath, or specified using \${WorkingDir}, which is the location of the command file.	
The HEC-DSS file must exist because otherwise the DSSUTL software prompts for the filename, and interaction with DSSUTL from TSTool is not enabled.	
DSSUTL will be run in TSTool's working directory (command file location).	
Specify a full or relative path to HEC-DSS, input, and output files (relative to working directory).	
The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\RunDSSUTL	
HEC-DSS file: Data\sample.dss Browse	
Input file: Data\sample_TA_Results.mco Browse	
Output file: Browse	
DSSUTL program: \${WorkingDir}\DSSUTL.EXE Browse	
Command: RunDSSUTL(DssFile="Data\sample.dss", InputFile="Data\sample_TA_Results.mco", DssutlProgram="\$(WorkingDir)'	\DSSUTL.
Add Working Directory (HEC-DSS) Add Working Directory (Input) Add Working Directory (Output) Add Working Directory (Program) Cancel O	ж

RunDSSUTL() Command Editor when Specifying Command Line

RunDSSUTL(Parameter=Value...)

Command Parameters

Parameter	Description	Default
DssFile	The HEC-DSS filename as an absolute path or	None – must be
	relative to the working directory. The file must	specified for most
	exist because TSTool does not interface with the	programs.
	program interactive mode prompts. The	
	parameter is passed to the program using the	
	DSSFILE= command line argument.	
InputFile	The DSS utility program command file to run.	None – must be
	The file must exist because TSTool does not	specified.
	interface with the utility program interactive mode	
	prompts. The input file name is passed to the	
	program using the INPUT= command line	
	argument.	
OutputFile	The DSS utility program output file, which	Not required – output
	contains logging information. This is passed to	will be to screen if
	the program using the OUTPUT= command line	command shell window
	argument. Specifying the argument will cause	is shown.
	output to be printed to the file and not the screen.	
	Note that some utility program commands write to	
	other output files (controlled by the command file	
	or other command line arguments), which should	
	not be confused with the output file for this	
D = = + 1 D = = = = = = =	argument.	
DSSUCIProgram	The DSS utility program to run. The PATH	If not specified,
	environment variable is used to locate the	DSSUIL.EXE will be
	executable if a full path is not specified. Specify	used and must be
	the specific DSS utility program to run if the	located in a directory
	default value is not appropriate.	instea in the PATH
	default value is not appropriate.	environment variable.

This page is intentionally blank.

Command Reference: RunningAverage()

Convert time series data to running average values

Version 10.13.00, 2012-10-25

The RunningAverage() command converts a time series' raw data values to a running average, resulting in data that are smoothed. New time series are NOT created (note that the newer RunningStatisticTimeSeries() command has more flexibility and the RunningAverage() command may be phased out in the future). There are several approaches to computing the running average (as specified by the AverageMethod command parameter):

- The centered running average requires that the number intervals on each site of a point be specified (e.g., specifying 1 will average 3 values at each point).
- The previous/future running average requires that the number of intervals prior to or after the current point be specified.
- The N-year running average is computed by averaging the current year and N 1 values from previous years, for a specific date. An average value is produced only if N non-missing values are available. Currently N-year running average values for Feb 29 for daily or finer data will always be missing because a sufficient number of values will not be found an option may be added in the future to allow Feb 29 values to be computed based on fewer than N values.
- A special case of the N-year running average (NAllYear) is to use all previous years' and the current value.

The following dialog is used to edit the command and illustrates the centered running average command syntax.

👌 Edit RunningAverage() Con	imand		
Convert a time series to a running ave	Convert a time series to a running average. Units, data type, etc., are not changed.		
A centered running average averages	the values at a date;	e/time and on either side.	
Previous and future running averages	use points only on or	one side of the current point, and optionally inclusive of the current point	
An N-year running average averages t	he values for the dat	ate/time and previous years (N years total).	
The RunningStatisticTimeSeries() command replac	ces this command and has more flexibility.	
TS list:	AllMatchingTSID	 Optional - indicates the time series to process (default=AllTS) 	
TSID (for TSList=AllMatchingTSID):	Center	×	
EnsembleID (for TSList=EnsembleID):		×	
Type of average:	Centered 🔹 🗸	 Required. 	
Number of intervals on each side:	3	Required (except for NAIIYear).	
	RunningAvera	age(TSList=AllMatchingTSID,TSID="Center",Av	
Commond.	erageMethod=	=Centered,Bracket=3)	
Command:			
Cancel OK			
		RunningAverage cente	

RunningAverage() Command Editor for Centered Running Average

The following dialog illustrates the N-year running average command syntax.

👌 Edit RunningAverage() Con	imand	×
Convert a time series to a running average. Units, data type, etc., are not changed.		
A centered running average averages	the values at a date	/time and on either side.
Previous and future running averages	use points only on o	ne side of the current point, and optionally inclusive of the current point.
An N-year running average averages t	he values for the da	te/time and previous years (N years total).
The RunningStatisticTimeSeries() command repla	ces this command and has more flexibility.
TS list:	AllMatchingTSID	 Optional - indicates the time series to process (default=AllTS).
TSID (for TSList=AllMatchingTSID):	NYear	▼
EnsembleID (for TSList=EnsembleID):		×
Type of average:	NYear 💽	 Required.
Number of years:	5	Required (except for NAIIYear).
	RunningAvera	ge(TSList=AllMatchingTSID,TSID="NYear",Ave
	- rageMethod=N	Year, Bracket=5)
Command:		
Cancel OK		
		RunningAverage nyea

RunningAverage() Command Editor for N-Year Running Average

RunningAverage(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSList	 Indicates the list of time series to be processed, one of: AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will be modified. AllTS – all time series before the command. EnsembleID – all time series in the ensemble will be modified. LastMatchingTSID – the last time series that matches the TSID (single TSID or TSID with wildcards) will be modified. SelectedTS – the time series are those selected with the SelectTimeSeries() command. 	AllTS
TSID	The time series identifier or alias for the time series to be modified, using the * wildcard character to match multiple time series.	Required if TSList=*TSID.
EnsembleID	The ensemble to be modified, if processing an ensemble.	Required if TSList= EnsembleID.
AverageMethod	 The method used to create the running average, one of: Centered - values on each side of a date/time are averaged. Future - average the next N (bracket) values but do not include the current value. FutureInclusive - average the next N (bracket) values and also include the current value. NYear - values for the current year and (N - 1) preceding years, for the same date/time, are averaged. NAllYear - values for the current year and all preceding years, for the same date/time, are averaged (missing values are allowed) Previous - average the previous N (bracket) values but do not include the current value. 	None – must be specified.
Bracket	For centered running average, the bracket is the number of points on each side of the current point (therefore a value of 1 will average 3 data values). For N-year running average, the bracket is the total number of years to average, including the current year.	None – must be specified.

A sample command file to convert State of Colorado HydroBase diversion time series to running averages is as follows:

```
# 0100501 - EMPIRE DITCH
ReadTimeSeries(Alias="Center","0100501.DWR.DivTotal.Month~HydroBase")
RunningAverage(TSList=AllMatchingTSID,TSID="Center",
AverageMethod=Centered,Bracket=3)
ReadTimeSeries(Alias="NYear","0100501.DWR.DivTotal.Month~HydroBase")
RunningAverage(TSList=AllMatchingTSID,TSID="NYear",
AverageMethod=NYear,Bracket=5)
0100501.DWR.DivTotal.Month~HydroBase
```

The resulting graph is as follows:



Results from RunningAverage() Commands

Command Reference: RunningStatisticTimeSeries()

Create a new time series containing running statistics computed from input

The RunningStatisticTimeSeries() command uses a sample of values from a time series to compute a running statistic, resulting in new time series. The two main purposes of the command are:

- 1. Compute a running statistic around a moving point, in order to smooth the time series, for example to focus on underlying short-term forcings rather than variability or noise
- 2. Compute a statistic by using values from the historical period, for example to illustrate how a daily value compares to historical values for the same day

The sample is computed relative to a date/time in the time series and consequently the resulting statistic may vary at each date/time in the time series. The resulting time series will have a time series identifier (TSID) that is the same as the original, with "-Running-" and the statistic appended to the data type (an alias can be assigned to customize the identifier that is used for processing). There are several approaches to determining the sample for the running statistic (as specified by the SampleMethod command parameter):

- The centered running statistic requires that the number intervals on each site of a point be specified (e.g., specifying 1 will use 3 values at each point).
- The previous/future running statistic requires that the number of intervals prior to or after the current point be specified.
- The N-year running statistic is computed by processing the current year and N 1 values from previous years, for a specific date. A resulting value is produced only if N non-missing values are available. Currently N-year running statistic values for Feb 29 for daily or finer data will always be missing because a sufficient number of values will not be found an option may be added in the future to allow Feb 29 values to be computed based on fewer than N values.
- A special case of the N-year running statistic (NAllYear) is to use all previous years' and the current value.

Statistics may be calculated directly from the sample or may be derived from an additional calculation. For example, the Mean statistic is computed by computing the mean of the values in the sample, and is assigned as the output time series value for the date/time that defines the sample. However, the PercentOfMean statistic is computed first by computing the Mean statistic and then dividing the original time series value by the mean, for each date/time in the time series. Derived statistics could be computed for many statistics but are provided only for cases that have common use.

The following dialog is used to edit the command and illustrates the centered running average command syntax.

Sedit RunningStatistic TimeS	Series() Command		
Create running statistic time series, where each new value is a statistic determined from a moving window of sample data (e.g., a running average).			
A centered running statistic is compute	ed from the values at a date/time and on eith	ner side.	
Previous and future running statistics	use points only on one side of the current po	pint, and optionally inclusive of the current point.	
An NYear running statistic uses the va	lues for the date/time and previous years (N	I years total).	
An NAIIYear running statistic uses the	values for the date/time and all previous yea	ars.	
TS list:	AllMatchingTSID	Optional - indicates the time series to process (default=AllTS).	
TSID (for TSList=AllMatchingTSID):	0100501.DWR.DivTotal.Month	×	
EnsembleID (for TSList=EnsembleID):		×	
Statistic:	Mean 💌	Required - statistic to calculate.	
Sample method:	Centered 🔽	Required - how to determine sample to analyze.	
Number of intervals on each side:	3	Required (except for NAIIYear).	
Allow missing count:		Optional - number of missing values allowed in sample (default=no limit).	
Alias to assign:	Select Specifier 🛛 🚽 => Centered	Optional - use %L for location, etc. (default=no alias).	
Probability units:	✓	Optional - units for probability statistic (default=Fraction).	
	RunningStatisticTimeSeries	(TSList=AllMatchingTSID,TSID="0100501.DWR.Div	
Command:	Total.Month",Statistic=Mea	n,SampleMethod=Centered,Bracket=3,Allas="Cent	
	Canter		

RunningStatisticTimeSeries() Command Editor for Centered Running Average

The command syntax is as follows:

RunningStatisticTimeSeries(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of:	AllTS
	1. AllMatchingTSID – all time series that match	
	the TSID (single TSID or TSID with wildcards)	
	2. AllTS – all time series generated before the	
	command	
	3. EnsembleID – all time series in the ensemble	
	4. FirstMatchingTSID – the first time series that	
	matches the TSID (single TSID or TSID with	
	wildcards)	
	5. LastMatchingTSID – the last time series that	
	matches the TSID (single TSID or TSID with	
	wildcards)	
	6. SelectedTS – the time series selected with the	
	SelectTimeSeries() command	
TSID	The time series identifier or alias for the time series to	Required if
	be processed, using the * wildcard character to match	TSList=*TSID.

Parameter	Description	Default
	multiple time series.	
EnsembleID	The ensemble to be processed, if processing an	Required if
	ensemble.	TSList=
		EnsembleID.
Statistic	The statistic to compute for each point in the created	None – must be
	time series, one of:	specified.
	 ExceedanceProbability – the probability 	
	that the value will be exceeded, best-suited for the	
	N* sample methods (see discussion below about	
	how statistics are computed)	
	 GeometricMean – geometric mean value 	
	• Lag-1AutoCorrelation – the autocorrelation	
	between values and the those that follow in the next	
	time step, given by:	
	$r_k = \sum_{i=1}^{N} \frac{N(X_i - Y_{mean})(Y_i + k - Y_{mean})}{\sum_{i=1}^{N} \frac{N(X_i - Y_{mean}$	
	$\sum_{i=1}^{n} (Y_i - Y_{mean})^2$	
	• Max – maximum value	
	• Mean – arithmetic mean value	
	 Median – median value 	
	• Min – minimum value	
	 NonexceedanceProbability - the 	
	probability that the value will not be exceeded, 1-	
	ExceedanceProbability, best-suited for the	
	N* sample methods (see discussion below about	
	how statistics are computed)	
	 PercentOfMax – percent of the Max statistic 	
	output	
	• PercentOfMean – percent of the Mean statistic	
	output	
	 PercentOfMedian – percent of the Median 	
	statistic output	
	 PercentOfMin – percent of the Min statistic 	
	output	
	• Skew – skew coefficient, as follows:	
	$Cs = \frac{N \sum_{i=1}^{N} (Y_i - Y_{mean})^3}{(N - 2)^3}$	
	$(n-1)(n-2)s^{2}$	
	where $s = standard deviation$	
	• Stadev – standard deviation	
	• Total – sum of values	
	Variance – variance	NT - 1
SampleMethod	The method used to determine the data sample for each	None – must be
	statistic calculation, one of:	specified.
	• Centered – N (bracket) values on each side of a	
	date/time and the center value	
	• Future – average the next N (bracket) values but	
	do not include the current value	1

Parameter	Description	Default
	 FutureInclusive – average the next N (bracket) values and also include the current value NYear – values for the current year and (N – 1) preceding years, for the same date/time in each year NAllYear – values for the current year and all preceding years, for the same date/time in each year (missing values are allowed) Previous – the previous N (bracket) values but do not include the current value PreviousInclusive – the previous N (bracket) values and also include the current value If a sample method such as NAllYear is desired, but including previous, current, and future values, then the NewStatisticTimeSeries() command can be used. 	
Bracket	For centered SampleMethod, the bracket is the number of points on each side of the current point (therefore a value of 1 will average 3 data values). For future and previous SampleMethod, the bracket is the number of previous or future values. For N-year SampleMethod, the bracket is the total number of years to process, including the current year.	None – must be specified.
AllowMissing Count	The number of values allowed to be missing in the sample and still compute the statistic. Care should be taken to specify a value that is relatively small for the sample size.	0 – no missing values are allowed in the sample
Alias	The alias to assign to the time series, as a literal string or using the special formatting characters listed by the command editor. The alias is a short identifier used by other commands to locate time series for processing, as an alternative to the time series identifier (TSID).	None – must be specified.
ProbabilityUnits	Units to use for calculated probability statistics.	Fraction(0- 1).

The following table provides additional information about how some statistics are computed.

Statistic Computation Details

Statistic	Computation Details
Exceedance Probability	 Rank the values in the sample from highest to lowest. Duplicate values are retained in the sample Search the list of ranked values, starting from the largest: a. If the value exactly matches a value in the sample:
	b. If the value is outside any values in the sample (e.g., for Future and

Statistic	Computation Details	
	Previous sample methods), then the exceedance value is not	
	calculated and warnings are generated. In this case a different sample	
	method should be used.	
	c. If the value does not exactly match a value in the sample (e.g., for	
	Future and Previous sample methods):	
	i. Find the ranked values that bound the value.	
	ii. The exceedance probability for each bounding value is	
	calculated as $i/(n + 1)$, where i is the list position (1 for the	
	largest value) and <i>n</i> is the sample size.	
	iii. The exceedance probability for the specific value is interpolated	
	from the bounding values. Note that the exceedance probability	
	is not recomputed by adding the value to the sample. If this is	
	desired, use the FutureInclusive or	
	PreviousInclusive sample methods.	
	Duplicate values are handled by using the first value found in the sequence of	
	duplicates.	

A sample command file to convert State of Colorado HydroBase diversion time series to running averages is as follows:

```
# SetInputPeriod(InputStart="1993-01",InputEnd="2000-12")
# 0100501 - EMPIRE DITCH
0100501.DWR.DivTotal.Month~HydroBase
RunningStatisticTimeSeries(TSList=AllMatchingTSID,
    TSID="0100501.DWR.DivTotal.Month",Statistic=Mean,SampleMethod=Centered,
    Bracket=3,Alias="Centered")
RunningStatisticTimeSeries(TSList=AllMatchingTSID,
    TSID="0100501.DWR.DivTotal.Month",Statistic=Mean,SampleMethod=NYear,
    Bracket=5,Alias="NYear")
ProcessTSProduct(TSProductFile="Test_RunningStatisticTimeSeries_Example.tsp")
```

The resulting graph is as follows:



Results from RunningStatisticTimeSeries() Commands

Command Reference: RunProgram()

Run an external program

The RunProgram() command runs an external program, given the full command line or individual command line parts, and waits until the program is finished before processing additional commands. The TSTool command will indicate a failure if the exit status from the program being run is non-zero. It is therefore possible to call an external program that reads and/or writes recognized time series formats to perform processing that TSTool cannot. One use of this command is to create a calibration environment where a model is run and then the results are read and displayed using TSTool. It is also useful to use TSTool's testing features to implement quality control checks for other software tools.

TSTool internally maintains a working directory that is used to convert relative paths to absolute paths to locate files. The working directory is by default the location of the last command file that was opened. The external program may assume that the working directory is the location from which TSTool software was started (or the installation location if started from a menu). Therefore, it may be necessary to run TSTool in batch mode from the directory where the external software's data files exist, use absolute paths to files, or use the $\{WorkingDir\}$ property in the command line. Use " in the command line or arguments to surround whitespace. Some operating systems may have limitations on command line length. The following dialog is used to edit the command and illustrates the command syntax.

😹 Edit RunProgram() commar	hd	X	
This command runs another program, and TSTool waits for it to complete before continuing. Commands must use a full path to files, TSTool must be started from the directory where files exist to use relative paths, or use \${WorkingDir} in the command line to specify files relative to the working directory. Use \" to indicate double quotes if needed to surround program name or program command-line parameters - this may be needed if there are spaces in paths. Specify the exit status indicator if program output messages must be used to determine the program exit status (e.g., "Status:"). Specify the program to run using the command line OR separate arguments - the latter makes it simpler to know how to treat whitespace in command line arguments. The program by default will be run with a command shell (e.g., cmd.exe on Windows) - specify as False if it is known that the program is an executable (not a shell command or script). Command Line 2. (More Rel Line 2.) (More Rel Line 2.) (More Rel Line 2.) (Parater Dure Rel Line 2.) (More Rel Line 2			
Command to run (with arguments):	_		
Program to run:		Required - if full command line is not specified above.	
Program argument 1:		Optional - as needed if Program is specified.	
Program argument 2:		Optional - as needed if Program is specified.	
Program argument 3:		Optional - as needed if Program is specified.	
Program argument 4:		Optional - as needed if Program is specified.	
Program argument 5:		Optional - as needed if Program is specified.	
Program argument 6:		Optional - as needed if Program is specified.	
Program argument 7:		Optional - as needed if Program is specified.	
Program argument 8:		Optional - as needed if Program is specified.	
Use command shell:	•	Optional - use command shell (default=True).	
Timeout (seconds):		Optional - default is no timeout.	
Exit status indicator:		Optional - output string to indicate status (default=use process exit status).	
Command:	RunProgram(CommandLine="echo Hello > \${WorkingDir}/Results/Test_RunProgram_CommandLine_echo_out.txt")		
	Cancel OK		
		RunProgram	

RunProgram() Command Editor when Specifying Command Line

```
RunProgram(Parameter=Value...)
```

Command Parameters

Parameter	Description	Default
CommandLine	The full program command line, with arguments.	Must be specified if the
	If the program executable is found in the PATH	Program parameter is
	environment variable, then only the program name	not specified.
	needs to be specified. Otherwise, specify an	•
	absolute path to the program or run TSTool from a	The Program
	command shell the same directory.	parameter will be used
		if both are specified.
	The \${WorkingDir} property can be used in	-
	the command line to indicate the working	
	directory (command file location) when	
	specifying file names.	
	For Windows, it may be necessary to place a $\$ "	
	at the start and end of the command line, if a full	
	command line is specified.	
Program	The name of the program to run. Program	Must be specified if the
	arguments are specified using the ProgramArg#	CommandLine
	parameter(s). See the CommandLine parameter	parameter is not
	for more information about parameter formatting	specified.
	and locating the executable.	
ProgramArg1,	Command like arguments used with Program. If	No arguments will be
ProgramArg2,	necessary, use \${WorkingDir} to specify the	used with Program.
etc.	working directory to locate files.	
UseCommandShell	If specified as False, the program will be run	True, using cmd.exe
	without using a command shell. A command shell	/C on Windows and
	is needed if the program is a script (batch file), a	/bin/sh -con
	shell command, or uses $>$, $ $, etc.	UNIX/Linux.
Timeout	The timeout in seconds – if the program has not	No timeout.
	yet returned, the process will be ended. Zero	
	indicates no timeout. This behavior varies and	
	is being enhanced.	
ExitStatus	By default, the program exit status is determined	Determine the exit
Indicator	from the process that is run. Normally 0 means	status from the process
	success and non-zero indicates an error.	exit value.
	However, the program may not exit with a non-	
	zero exit status when an error occurs. If the	
	program instead uses an output string like STOP	
	3 to indicate the status, use this parameter to	
	indicate the leading string, which is followed by	
	the exit status (e.g., STOP).	

The following figure illustrates how a command would be entered using the program name and parts, and use the command shell to run. Note that the output redirection character ">" is entered as a program argument. The *echo* program on Windows is actually internal to the *cmd.exe* command shell and therefore must be run using the command shell (the default behavior).

Edit RunProgram() command This command runs another program, and TSTool waits for it to complete before continuing. Commands must use a full path to files, TSTool must be started from the directory where files exist to use relative paths, or use \${WorkingDir} in the command line to specify files relative to the working directory. Use \{' to indicate double quotes if needed to surround program name or program command-line parameters - this may be needed if there are spaces in paths. Specify the exit status indicator if program output messages must be used to determine the program exit status (e.g., "Status:"). Specify the program to run using the command line OR separate arguments - the latter makes it simpler to know how to treat whitespace in command or script). The program by default will be run with a command shell (e.g., cmd.exe on Windows) - specify as False if it is known that the program is an executable (not a shell command or script).		
Command to run (with arguments):		
Program to run:	echo	Required - if full command line is not specified above.
Program argument 1:	Hello	Optional - as needed if Program is specified.
Program argument 2:	>	Optional - as needed if Program is specified.
Program argument 3:	\${WorkingDir}/Results/Test_RunProgram_Program_echo_out.txt	Optional - as needed if Program is specified.
Program argument 4:		Optional - as needed if Program is specified.
Program argument 5:		Optional - as needed if Program is specified.
Program argument 6:		Optional - as needed if Program is specified.
Program argument 7:		Optional - as needed if Program is specified.
Program argument 8:		Optional - as needed if Program is specified.
Use command shell:		Optional - use command shell (default=True).
Timeout (seconds):		Optional - default is no timeout.
Exit status indicator:		Optional - output string to indicate status (default=use process exit status).
Command:	RunProgram(Program=echo,ProgramArg1=Helld ts/Test_RunProgram_Program_echo_out.txt)	o, Program&rg2=>, Program&rg3=\${WorkingDir}/Resul
	Cancel OK	
		RunProgram_Program

RunProgram() Command Editor when Specifying Program and Arguments

The following figure illustrates how a command can be run without a command shell and using the program output to determine the exit status. The *testecho.exe* program is a compiled executable and can therefore be run without a command shell. Because the standard output is being evaluated for the exit value, the output cannot be redirected to a file with > (this would result in no output being available to TSTool to evaluate), and > is only recognized if running with a command shell in any case.

The following approach is suitable, for example, when running a compiled model or data analysis tool. However, if the tool is run using a script or batch file, then a command shell must be used.

Edit RunProgram() command This command runs another program, and TSTool waits for it to complete before continuing. Commands must use a full path to files, TSTool must be started from the directory where files exist to use relative paths, or use \${WorkingDir} in the command line to specify files relative to the working directory. Use \{' to indicate double quotes if needed to surround program name or program command-line parameters - this may be needed if there are spaces in paths. Specify the exit status indicator if program output messages must be used to determine the program exit status (e.g., "Status:"). Specify the program to run using the command line OR separate arguments - the latter makes it simpler to know how to treat whitespace in command line arguments. The program by default will be run with a command shell (e.g., cmd.exe on Windows) - specify as False if it is known that the program is an executable (not a shell command or script). Command to run (with arguments):		
Program to run:	\${WorkingDir}/testecho.exe	Required - if full command line is not specified above.
Program argument 1:	STOP 2	Optional - as needed if Program is specified.
Program argument 2:		Optional - as needed if Program is specified.
Program argument 3:		Optional - as needed if Program is specified.
Program argument 4:		Optional - as needed if Program is specified.
Program argument 5:		Optional - as needed if Program is specified.
Program argument 6:		Optional - as needed if Program is specified.
Program argument 7:		Optional - as needed if Program is specified.
Program argument 8:		Optional - as needed if Program is specified.
Use command shell:	T	Optional - use command shell (default=True).
Timeout (seconds):		Optional - default is no timeout.
Exit status indicator:	STOP	Optional - output string to indicate status (default=use process exit status).
Command:	RunProgram(Program="\${WorkingDir}/testech 2",ExitStatusIndicator="STOP")	ho.exe",ProgramArg1="STOP
	Cancel OK	
		RunProgram Program ExitStatusIndicator

RunProgram() Command Editor when Specifying Program, Arguments, and Exit Status Indicator

Command Reference: RunPython()

Run a Python script

The RunPython() command runs a Python script, waiting until execution is finished before processing additional commands. Python is a powerful scripting language that is widely used (see http://www.python.org). This command allows Python scripts to be run using a variety of Python interpreters, as shown in the following table. It is assumed that Python is installed in the standard directory for the distribution. New versions of Python will reside in similar locations to those shown below.

Interpreter	Language, Program Name	
(Website)	(Example Install Home)	Comments
IronPython	.NET, ipy	Useful for integrating with .NET
(ironpython.net)	(C:\Program Files\IronPython 2.6)	applications, in particular to manipulate
		Microsoft Office software data files. Can
		use .NET assembly code (but this code in
		a Python script is only recognized by
		IronPython). Integration can occur within
		a running .NET application (essentially
		extending the functionality of the .NET
		application). Version 2.6 requires .NET
		2.0. Version 2.6.1 requires .NET 4.0.
Jython	Java, jython	Useful for integrating with Java
(www.jython.org)	$(C:\jython 2.5.1)$	applications, such as TSTool. Can use
		Java code (but this code in a Python script
		is only recognized by Jython).
Jython embedded	Java	Useful for integrating with Java
(www.jython.org)	(C: $jython 2.5.1$, but must use the	applications, such as TSTool. Can use
	installer option to create a JAR file in	Java code (but this code in a Python script
	order to embed – this is the file that is	is only recognized by Jython). Integration
	distributed with TSTool).	can occur within a running Java
		application (essentially extending the
		functionality of the Java application).
Python	C, python	The original Python interpreter, which
(www.python.org)	$(C: \forall Python 25)$	defines the Python language specification.

RunPython() Supported Python Interpreters

Python implementations have similar file organization, with the main executable (or batch file) residing in the main install folder. Core functionality is typically completely handled within the interpreter code and/or Python code included in the *Lib* folder under the main installation folder. Extended capabilities such as third-party add-ons are made available as module libraries that are installed in the *Lib*/*site-packages* folder. These folders are typically automatically included in the Python path and will be found when import statements are used in Python scripts. The folder for the main Python script that is run to start an execution is also typically included in the Python path by the interpreter at runtime. If any additional Python modules needed to be found, they can be added to the Python path at runtime (see the PythonPath command parameter below).

If the embedded Jython is used, then there may be no reliance on any other software if the core Python capabilities can be used. However, if third-party packages are used, it may be best to install them with the Jython distribution (e.g., in *Lib\site-packages*) so that the packages can be used for independent testing prior to use in the embedded interpreter. For example, perform a typical Jython install (e.g., into *C:\Jython2.5.1*), install the third-party packages into this location (using the installer for the package or directly copying into the *Lib\site-packages folder*), and then specify the PythonPath=C:\Jython2.5.1\Lib\site-packages) command parameter.

If a non-embedded approach is used, then IronPython, Jython, or Python must be installed on the computer for the appropriate Interpreter command parameter value. The interpreter program will be found if the installation folder is defined in the PATH environment variable, or use the Program command parameter to specify the full path to the interpreter program to run. The script is then run by running the following (see full parameter descriptions below):

Program InputFile Arguments

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

👌 Edit RunPythe	on() Command		
Run a Python script, ł	by calling a stand-alone interpreter or embedded Jython interpreter.		
Python scripts are use	eful for manipulating data outside of TSTool's capabilities.		
IronPython is the .NE	T implementation of Python and Jython is the Java implementation,	offering integration with packages available for each language.	
Specify a full or relativ	ve path to the script file (relative to working directory).		
Strings with special m	eaning can be specified for any parameter and include:		
\" - literal quote, ne	eded to surround arguments that include spaces.		
\${InstallDir} - the so	oftware installation directory.		
\${WorkingDir} - the	working directory (location of command file).		
\${Property} - other	global properties.		
The working directory	is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\comman	ds\general\RunPython	
Interpreter:	Python 💌	Required - interpreter to run.	
Program:		Optional - program to run (default=for interpreter, find using PATH).	
Python path:		Optional - add to Python path, use : or ; to separate.	
Python script to run:	Data/readwritefile.py	Browse	
	<pre>\${WorkingDir}/Data/readwritefile.txt</pre>		
A	\${WorkingDir}/Results/Test RunPython Int	erpreter=Python out.txt	
Arguments:		-	
	DupDuther (Interpreter="Duther", InputFile	-Whote/readwritefile www.browmenterWi(Ne	
	RunPython(Interpreter="Python", InputFile="Data/readwriteFile.py", Arguments="\${wo		
Command:	nmand:		
	(workingbil)/Results/lest_RunPython_interpreterPython_out.txt*)		
Add Working Directory Cancel OK			
		RunPythor	

RunPython() Command Editor

```
RunPython(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
Interpreter	The Python interpreter to run, one of:	None – must be
	• IronPython	specified.
	• Jython	
	• JythonEmbedded	
	• Python	
	Global properties can be used with the \${Property}	
	syntax.	
Program	The Python interpreter program to run. Specify as a	Determined based on
	full path to the installed program, or only the program	the Interpreter
	name (in which case the path to the program must be	parameter:
	included in the PATH environment variable). Global	• IronPython: ipy
	properties can be used with the \${Property}	• Jython: jython
	syntax.	• Python: python
PythonPath	Additional locations for modules, to be added to the	None – the core Python
	Python path. Specify paths separated by ; or :. For	capabilities are
	embedded Jython, the sys.path is updated prior to	available.
	running the script. For non-embedded interpreters, the	
	JYTHONPATH environment variable is updated for the	
	interpreter, which results in sys.path being updated.	
	Global properties can be used with the \${Property}	
T	syntax.	NT (1
InputFile	The Python script to run, specified as an absolute path	None – must be
	or relative to the command file. See the Arguments	specified.
	parameter for information about using properties to	
	with the c [Drop control support of the second seco	
Arguments	Arguments to pass to the script, such as the names of	None grouments are
Arguments	files to process. Use the S WorkingDir property	optional
	to specify the location of the command file. Use	optionui.
	$s{\text{InstallDir}}$ for the TSTool install folder. Use	
	\" to surround arguments that include spaces	
	Separate arguments by a space Global properties can	
	be used with the \${Property} syntax.	

The following command example illustrates how to run a Python script.

```
RunPython(InputFile="Data/readwritefile.py",
Interpreter="JythonEmbedded",Arguments="${WorkingDir}/Data/readwritefile.txt
${WorkingDir}/Results/Test_RunPython_Interpreter=JythonEmbedded_out.txt")
```

The corresponding Python script is as follows:

```
#
# Test command for running Python script from TSTool
#
import sys
import os
print "start of script"
print 'os.getcwd()="' + os.getcwd() + '"'
infile = None
outfile = None
if (len(sys.argv) < 3):
    print "Error. Expecting input file name as first command line argument,
output file name as second."
    sys.exit(1)
else:
    infile = sys.argv[1]
    outfile = sys.argv[2]
   print 'Input file to process is "' + infile + '"'
   print 'Output file to create is "' + outfile + '"'
inf=open(infile,'r')
outf=open(outfile,'w')
for line in inf:
    outf.write("out: " + line)
inf.close()
outf.close()
print "end of script"
```

The data file is as follows:

Line 1 (first line) Line 2 Line 3 Line 4 Line 5 (last line)

The output file is as follows:

```
out: Line 1 (first line)
out: Line 2
out: Line 3
out: Line 4
out: Line 5 (last line)
```

The following example illustrates the use of double quotes to surround Python script command-line arguments, to ensure that spaces and equal sign characters are properly handled:

```
# Retrieve the MEI (ENSO) index
WebGet(URI="http://www.esrl.noaa.gov/psd/data/correlation/mei.data",LocalFile="mei.data")
# Convert the MEI data file to a CSV file that can be read by TSTool
RunPython(Interpreter="Python",InputFile="mei2csv.py",Arguments="\"InputFile=${WorkingDir}/mei.data\"
\"OutputFile=${WorkingDir}/mei.csv\" \"LogFile=${WorkingDir}/mei2csv.log\"")
```

Command Reference: Scale()

Scale time series data values by a constant value

Version 10.12.00, 2012-08-13

The Scale() command scales each non-missing value in the specified time series. The value to use for scaling can be specified as a constant, monthly values, or special values that indicate to scale by the number of days in the month.

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

Sedit Scale() Command		
Scale time series data values (multiply	non-missing by a constant).	
Specify dates with precision approprial	te for the data, use blank for all a	vailable data, OutputStart, or OutputEnd.
TS list:	AllMatchingTSID 🛛 👻	Optional - indicates the time series to process (default=AIITS).
TSID (for TSList=AllMatchingTSID):	*	×
EnsembleID (for TSList=EnsembleID):		×
Scale value:	5	Required (if no monthly values) - constant scale value, DaysInMonth, or DaysInMonthInverse.
Monthly values:		Required (if no single value) - monthly scale values, separated by commas.
Analysis start:		Optional - analysis start date/time (default=full time series period).
Analysis end:		Optional - analysis end date/time (default=full time series period).
New units:		Optional - new data units.
	Scale(TSList=AllMato	hingTSID,TSID="*",ScaleValue=.5)
Command:		
		Cancel OK
		Scale

Scale() Command Editor

The command syntax is as follows:

```
Scale(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be	Allts
	processed, one of:	
	• AllMatchingTSID – all time series	
	that match the TSID (single TSID or	
	TSID with wildcards) will be modified.	
	• AllTS – all time series before the	
	command.	
	• EnsembleID – all time series in the	
	ensemble will be modified.	
	• FirstMatchingTSID – the first time	
	series that matches the TSID (single	
	TSID or TSID with wildcards) will be	

Parameter	Description	Default
	 modified. LastMatchingTSID - the last time series that matches the TSID (single TSID or TSID with wildcards) will be modified. SelectedTS - the time series are those selected with the SelectTimeSeries() command. 	
TSID	The time series identifier or alias for the time series to be modified, using the * wildcard character to match multiple time series.	Required if TSList=*TSID
EnsembleID	The ensemble to be modified, if processing an ensemble.	Required if TSList=EnsembleID
ScaleValue	 One of the following: The numerical value to scale to the time series. DaysInMonth to indicate a scale of the number of days in the month. DaysInMonthInverse to indicate a scale of the inverse of the number of days in the month. 	None – must be specified.
MonthValues	Monthly scale values, the fist being for January.	Use ScaleValue.
AnalysisStart	The date/time to start analyzing data.	Full period is analyzed.
AnalysisEnd	The date/time to end analyzing data.	Full period is analyzed.
NewUnits	New data units for the resulting time series.	Do not change the units.

The following example scales a precipitation time series from the State of Colorado's HydroBase by a factor of 3.5:

```
# 1458 - CENTER 4 SSW
1458.NOAA.Precip.Month~HydroBase
Scale(TSList=AllMatchingTSID,TSID="1458.NOAA.Precip.Month",ScaleValue=3.5)
```

The following example scales a monthly streamflow time series with units of ACFT (volume per month) in order to convert the data to average CFS flow values (note that two scale commands are required because the DaysInMonthInverse value cannot currently be combined with a numerical value in one command). See also the ConvertDataUnits() command for simple units conversions.

```
# 06754000 - SOUTH PLATTE RIVER NEAR KERSEY
06754000.DWR.Streamflow.Month~HydroBase
Scale(TSList=AllMatchingTSID,TSID="06754000.DWR.Streamflow.Month",
ScaleValue=.5042)
Scale(TSList=AllMatchingTSID,TSID="06754000.DWR.Streamflow.Month",
ScaleValue=DaysInMonthInverse,NewUnits="CFS")
06754000.DWR.Streamflow.Month~HydroBase
```

Command Reference: SelectTimeSeries()

Select time series for additional processing

Version 10.12.00, 2012-10-25

The SelectTimeSeries() command selects output time series, as if done interactively, to indicate which time series should be operated on by following commands. The command minimizes the need for the Free() command, when used in conjunction with other commands that use a time series list based on selected time series (TSList=SelectedTS). See also the DeselectTimeSeries() command.

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

This command selects time series, similar to how time series are interactively selected. Selected time series	may then be used by other commands.		
For example, commands may allow selected time series to be processed, rather than default to all time ser	ies.		
When matching a time series identifier (TSID) pattern:			
The dot-delimited time series identifier parts are Location.Databource.DataType.Interval.bcenario			
Lise * to match all time series.			
Use A* to match all time series with alias or location starting with A.			
Use *.*.XXXXX.*.* to match all time series with a data type XXXXX.			
When selecting time series by specifying time series positions (not recommended for production wor	k because positions may change):		
The first time series created is position 1.			
Separate numbers by a comma. Specify a range, for example, as 1-3. A valid combination is: 1,5-10,1	3		
When selecting time series by matching a property:			
Currently only string properties are supported.			
Comparisons are case-independent.			
TS list: AllMatchingTSLD V Optional - Indicates the	time series to process (default=All15).		
TSID (for TSList=AllMatchingTSID): 40*	×		
EnsembleID (for TSList=EnsembleID):	~		
Time series position(s) (for TSList=TSPosition): For example, 1,2,7-8 (p	oositions are 1+).		
Deselect all first?: True 💙 Optional - eliminates ne	ed for separate deselect (default=False).		
Property name: Optional - use to match	user-defined properties.		
Property criterion: 🛛 😽 Required if matching us	er-defined property.		
Property value: Required if matching us	er-defined property.		
SelectTimeSeries(TSList=AllMatching	TSID,TSID="40*",Deselec		
Command, tAllFirst=True)	, tAllFirst=True)		
Command:	b		
Cancel OK			

SelectTimeSeries() Command Editor

The command syntax is as follows:

SelectTimeSeries(Parameter=Value,...)

Parameter	Description	Default
Parameter TSList	 Description Indicates the list of time series to be processed, one of: AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will be modified. AllTS – all time series before the command. EnsembleID – all time series in the ensemble will be modified (see the EnsembleID parameter). LastMatchingTSID – the last time series that matches the TSID (single TSID or TSID with wildcards) will be modified. TSPosition – time series specified by position in the results 	AllTS
	list (see TSPosition parameter below).	
TSID	The time series identifier or alias for the time series to be modified, using the * wildcard character to match multiple time series.	Required if TSList=*TSID
EnsembleID	The ensemble to be modified, if processing an ensemble.	Required if TSList=EnsembleID
TSPosition	A list of time series positions (1+) in output, separated by commas. Ranges can be specified as Start-End.	Required if TSList=TSPosition
DeselectAllFirst	Indicates whether all time series should be deselected before selecting the specified time series: True or False.	False
PropertyName	Name of user-defined property to check.	
PropertyCriterion	Criterion to evaluate to determine which properties match.	Required if PropertyName is specified.
PropertyValue	Value to check against the property value, using criterion.	Required if PropertyName is specified.

Command Parameters

A sample command file is as follows:

```
NewPatternTimeSeries(Alias="401234",NewTSID="401234..Precip.Day",
Description="Example data",SetStart="2000-01-01",SetEnd="2000-12-31",
Units="IN",PatternValues="0,1,3,0,0,0")
SelectTimeSeries(TSList=AllMatchingTSID,TSID="40*",DeselectAllFirst=True)
```

Command Reference: SetAutoExtendPeriod()

Set whether time series periods should automatically be extended to the output

period Version 08.16.03, 2008-07-29

By default, the time series period is extended to include the output period, if specified, when a time series is read. See also the SetOutputPeriod() command. If the extended period subsequently contains missing data, it can be filled with other commands. The SetAutoExtendPeriod() command can be used to change this setting if it is not desirable (e.g., for performance reasons).

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

Edit SetAutoExtendPeriod() Command
This command controls whether the the time series period automatically is extended to match the output period when reading data.
Setting the option to True results in the time series period automatically being extended
to include the output period, if SetOutputPeriod() is used prior to this command.
The default if SetAutoExtendPeriod() is not used is True, allowing for data filling and manipulation for the output period.
Set the option to False to improve performance, when data filling and manipulation are not needed on the extended period.
Automatically extend period?: False
SetAutoExtendPeriod(AutoExtendPeriod=False)
Command:
Cancel OK

SetAutoExtendPeriod() Command Editor

The command syntax is as follows:

```
SetAutoExtendPeriod(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
AutoExtendPeriod	Indicate whether the period of time series should automatically be extended to the output period when time series are read,	None – must be specified. The default is True if this command is not used.
	True or False.	

A sample command file is as follows:

SetAutoExtendPeriod(AutoExtendPeriod=False)

This page is intentionally blank.

Command Reference: SetAveragePeriod()

Set the period used to compute historical averages

Version 08.16.03, 2008-07-29

The SetAveragePeriod() command sets the period that is used to compute historic averages used with the FillHistMonthAverage() and FillHistYearAverage() commands. If the averaging period is not specified, the available period is used. Use a SetAveragePeriod() command if a subset of the data should be used to compute averages.

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

📝 Edit SetAveragePeriod() Command			
Use SetAveragePeriod() to limit the period used to compute historical averages immediately after data are read.			
Calculating historical av	verages for data filling is only supported for monthly and yearly time series.		
Averages are by defau	It computed for the available period.		
Enter dates as MM/YYY	YY or YYYY-MM (or YYYY for yearly). Enter * to use all available data.		
Average period start:	1950-01		
Average period end:	2002-12		
	SetAveragePeriod(AverageStart="1950-01",AverageEnd="2002-12")		
Command:			
1			
	Cancel OK		
	S	etAveragePeric	

SetAveragePeriod() Command Editor

SetAveragePeriod(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
AverageStart	The date for the start of the averaging period. The precision of the date should agree with that of time series to be processed, and is limited to monthly and	None – must be specified.
AverageEnd	The date for the end of the averaging period. The precision of the date should agree with that of time series to be processed, and is limited to monthly and yearly precision.	None – must be specified.

A sample command file is as follows:

SetAveragePeriod(1950-01,2002-12)

Command Reference: SetConstant()

Set time series data to a single or monthly constant values

Version 08.15.00, 2008-05-11

The SetConstant() command sets the values of a time series to a single or monthly constant values.

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

💊 Edit SetConstant() Command 🛛 🛛 🔀					
Set time series data values to a single or monthly (Jan - Dec) constant values.					
If the time series data interval is month	n or smaller, constant values for e	ach month can be specified.			
In this case, each date/time that match	nes a month will have its correspo	nding value set.			
TS list:	LastMatchingTSID 🔽	Indicates the time series to process (default=AIITS).			
TSID (for TSList=AllMatchingTSID):	08235700.DVVR.Streamflow.Mont	h			
EnsembleID (for TSList=EnsembleID):		V			
Constant value:	0	Use for all intervals			
Monthly values:		Monthly values, separated by commas.			
Set start:		Set start (optional). Default is all.			
Set End:	1950-01	Set end (optional). Default is all.			
Command:	SetConstant(TSList=Last amflow.Month",ConstantV	MatchingTSID,TSID="08235700.DWR.Stre alue=0,SetEnd="1950-01")			
Cancel OK					

SetConstant() Command Editor

```
SetConstant(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be	Allts
	processed, one of:	
	• AllMatchingTSID – all time	
	series that match the TSID (single	
	TSID or TSID with wildcards) will	
	be modified.	
	• AllTS – all time series before the	
	Command.	
	• Ensemble will be modified	
	Legt Met ab in a TOL the last	
	time series that matches the TSTD	
	(single TSID or TSID with	
	wildcards) will be modified	
	 SelectedTS – the time series are 	
	those selected with the	
	SelectTimeSeries()	
	command.	
TSID	The time series identifier or alias for the	Required if TSList=*TSID.
	time series to be modified, using the *	
	wildcard character to match multiple	
	time series.	
EnsembleID	The ensemble to be modified, if	Required if
	processing an ensemble.	TSList=EnsembleID.
ConstantValue	The constant value to use as the data	None – must be specified, or
	value.	specify monthly values.
MonthValues	Monthly values to use as the data values.	None – must be specified, or
	Twelve values can be specified,	specify a constant value.
	separated by commas. If the time series	
	data interval is less than monthly, each	
SetStart	The starting data/time for the data set	Sat data for the full pariod
SetEnd	The starting date/time for the data set	Set data for the full period

A sample command file to process a time series from the State of Colorado's HydroBase is as follows (only the early period is set to zero):

```
# 08235700 - ALAMOSA RIVER BELOW CASTLEMAN GULCH NEAR JASPER
08235700.DWR.Streamflow.Month~HydroBase
SetConstant(TSList=AllMatchingTSID,TSID="08235700.DWR.Streamflow.Month",
ConstantValue=0,SetEnd="1950-01")
```

Command Reference: SetDataValue()

Set a data value at a single date/time

Version 08.16.04, 2008-09-12

The SetDataValue() command sets a single data value in a time series. Consequently, it can be used to condition a value for subsequent filling (e.g., with FillRepeat()) or to "hard-code" data that may not be available in files or databases. It is good practice to insert comments when editing data to explain the edits. See also the SetConstant() command.

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

🛃 Edit SetData¥alue() Comman	d 🔀
Set a data value for a specific date/tir	ne.
Specify the date/time to an appropriat	e precision using standard formats like the following:
YYYY for year interval	
MM/YYYY or YYYY-MM for month i	nterval
MM/DD/YYYY or YYYY-MM-DD for	day interval
MM/DD/YYYY HH or YYYY-MM-DD	HH for hour interval
MM/DD/YYYY hh:mm or YYYY-MM-	DD hh:mm for minute interval
See also the SetConstant() command	·
TS list:	AllMatchingTSID Indicates the time series to process (default=AlITS).
TSID (for TSList=AllMatchingTSID):	08235700.DWR.Streamflow.Month
EnsembleID (for TSList=EnsembleID):	
Date/time to set value:	1950-01
New data value:	550
	SetDataValue(TSList=AllMatchingTSID,TSID="08235700.DWR.
Command:	Streamflow.Month",SetDateTime="1950-01",NewValue=550)
	· · · · · · · · · · · · · · · · · · ·
	Cancel OK
	SetData

SetDataValue() Command Editor

```
SetDataValue(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be	Allts
	processed, one of:	
	• AllMatchingTSID – all time	
	series that match the TSID (single	
	TSID or TSID with wildcards) will	
	be modified.	
	• AllTS – all time series before the	
	command.	
	• EnsembleID – all time series in	
	the ensemble will be modified.	
	• FirstMatchingTSID – the first	
	time series that matches the TSID	
	(single TSID or TSID with	
	wildcards) will be modified.	
	• LastMatchingTSID – the last	
	time series that matches the TSID	
	(single ISID or ISID with	
	wildcards) will be modified.	
	• Selected TS – the time series are	
	select TimeSeries()	
	command	
TSID	The time series identifier or alias for the	Required if TSList=*TSID.
	time series to be modified, using the *	
	wildcard character to match multiple	
	time series.	
EnsembleID	The ensemble to be modified, if	Required if
	processing an ensemble.	TSList=EnsembleID.
SetDateTime	The date/time at which the data value	None – must be specified.
	should be set. Specify the date/time	
	precision according to the time series that	
	is being manipulated.	
NewValue	The new data value.	None – must be specified.

A sample command file to process data from the State of Colorado's HydroBase database is as follows:

```
# 08235700 - ALAMOSA RIVER BELOW CASTLEMAN GULCH NEAR JASPER
08235700.DWR.Streamflow.Month~HydroBase
SetDataValue(TSList=AllMatchingTSID,TSID="08235700.DWR.Streamflow.Month",
SetDateTime="1950-01",NewValue=550)
```

Command Reference: SetDebugLevel()

Set level for debug messages

Version 08.16.00, 2008-07-08

The setDebugLevel() command sets the debug levels for screen and log file diagnostic messages. This command can be used multiple times with different debug level (e.g., to isolate a problem). Currently the debug level applies to all components. In the future logging control may be grouped by component. Levels are not completely consistent but the following guidelines can be followed:

0 = no messages

1 =important messages generated in applications

2 =important messages generated in commands

3+ = messages generated in commands that may explain other problems

10+ = messages in processing code that may still be useful to end users

30+ = low-level messages, for example generated while reading from files or databases

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

🛃 Edit SetDebug	Level() command
Set the level for scre	een and	/or log file debug messages.
Debug information is	useful	for troubleshooting. The default debug level is 0.
Setting the debug lev	/el to a l	higher number prints more information.
Debug levels can be	increas	sed before and decreased after specific commands to troublesheet the commands.
Screen debug level:	0	0=none, 100=all, blank=no change.
Log file debug level:	10	0=none, 100=all, blank=no change.
Command:	SetDe	bugLevel(ScreenLevel=0,LogFileLevel=10)
		Cancel OK
		Set

SetDebugLevel() Command Editor

SetDebugLevel(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
ScreenLevel	The debug level for the screen $(0+)$.	Keep previous setting.
LogFileLevel	The debug level for the log file $(0+)$.	Keep previous setting.

A sample command file is as follows:

Cot Dobugt or old (Carcordon or old) Log Eilot or old (0)	
PERDEDIGTEAT (PCTEETTEAT-A'TOALITETEAT-IA)	
Command Reference: SetFromTS()

Set time series data using another time series

Version 10.00.01, 2011-05-11

The SetFromTS() command sets data in a dependent time series by transferring values from an independent time series. A period can be specified to limit the period that is processed. See also the FillFromTS() command, which will transfer values only when the dependent time series has missing data. Only data values are transferred – time series header information (e.g., data type, alias) will not be modified. If multiple time series or an ensemble is being processed, the number of independent time series must be one or the same number as the time series being filled.

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

Sedit SetFromTS() Command			×	
Copy data values from the independent time series to replace val	Topy data values from the independent time series to replace values in the dependent time series.			
All data values (by default including missing data) in the set period	d will be copied.			
If one independent time series is specified, it will be used for all de	ependent time series.			
If multiple independent time series are specified (e.g., for ensemb	bles), the same number of depende	ent time series must be specified.		
Use a SetOutputPeriod() command in the dependent time series pro- Specify dates with precision appropriate for the date, black for all	erioa Will be extendea. Lavailable data: OutputStart: or O	utoutEod		
The set period is for the independent time series.	ravailable data, outputotart, or o	apatena.		
Dependent TS List:	AllMatchingTSID 🚽 🗸	Optional - indicates the time series to process (default=AllTS).		
TSID (for TSList=AllMatchingTSID):	08241000.DWR.Streamflow.Mont	h	~	
EnsembleID (for TSList=EnsembleID):			~	
Independent TS List:	AllMatchingTSID 🛛 👻	Optional - indicates the time series to process (default=AllTS).		
Independent TSID (for Independent TSList=AllMatchingTSID): 08240500.DWR.Streamflow.Month		h	~	
Independent EnsembleID (for Independent TSList=EnsembleID):			\sim	
Set start:		Optional - set start (default is full period).		
Set End:		Optional - set end (default is full period).		
Transfer data how:	ByDateTime 🔽	Required - how are data values transferred?		
Handle missing data how?:	~	Optional - missing in independent handled how? (default=SetMiss	;ing).	
Set data flags?:	~	Optional - should data flags be copied (default=True).		
Recalculate limits:	×	Optional - recalculate original data limits after set (default=False)).	
Command:	SetFromTS(TSList=AllMatchingTSID,TSID="08241000.DWR.Streamflo w.Month",IndependentTSList=AllMatchingTSID,IndependentTSID="0 8240500.DWR.Streamflow.Month",TransferHow=ByDateTime)			
	Cancel OK			

SetFromTS() Command Editor

```
SetFromTS(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed,	AllTS
	one of:	
	• AllMatchingTSID – all time series that	
	match the TSID (single TSID or TSID with	
	wildcards) will be modified.	
	• AllTS – all time series before the command.	
	• EnsembleID – all time series in the	
	ensemble will be modified.	
	 FirstMatchingTSID – the first time 	
	series that matches the TSID (single TSID or	
	TSID with wildcards) will be modified.	
	• LastMatchingTSID – the last time series	
	that matches the TSID (single TSID or TSID with wildcards) will be modified	
	 SelectedTS – the time series are those 	
	selected with the SelectTimeSeries()	
	command.	
TSID	The time series identifier or alias for the time	Required when
	series to be modified, using the * wildcard	TSList=*TSID
	character to match multiple time series.	
EnsembleID	The ensemble to be modified, if processing an	Required when
	ensemble.	TSList=EnsembleID.
Independent	Indicates how to determine the list of independent	Allts
TSLIST	time series (see the explanation of TSList).	
Independent	The time series identifier or alias for the	Required when a
1510	independent time series (see the explanation of	Independent TSList=
Indopondont	TSID). The encomple identifier for the independent time	De graine d'arch en
EnsembleID	arrise (see the explanation of Engemble JD)	Independent TSList-
	series (see the explanation of EffsetibleTD).	EnsembleID.
SetStart	The date/time to start setting data, if other than	Full period if * is
	the full time series period.	specified.
SetEnd	The date/time to end setting data, if other than the	Full period if * is
	full time series period.	specified.
TransferHow	Indicates how to transfer data:	None – must be specified.
	• ByDateTime – a date/time in one time	
	series will be lined up with the other time	
	series.	
	• Sequentially – data from the	
	avon if the data/time does not align (used	
	when transferring continuous data over Ech	
	28/29 without gaps)	

Parameter	Description	Default
HandleMissingHow	 Indicates how to handle missing data in the independent time series: IgnoreMissing – missing values in the 	SetMissing
	independent time series WILL NOT be transferred to the dependent time series.	
	• SetMissing – missing values in the independent time series WILL be transferred to the dependent time series.	
	• SetOnlyMissingValues – only the missing values in the independent time series will be transferred, useful when a separate	
	time series has been used to insert additional missing values.	
SetDataFlags	Indicates if data flags should also be transferred.	True
RecalcLimits	Available only for monthly time series. Indicate	False (only the values in
	whether the original data limits for the time series	the initial time series will
	should be recalculated after the setting the time	be used for historical
	series values. Setting to True is appropriate if	data).
	the independent time series provides observations	
	consistent with the original data.	

A sample command file to process data from the State of Colorado's HydroBase is as follows:

```
# 08241000 - TRINCHERA CREEK ABOVE MOUNTAIN HOME RESERVOIR
08241000.DWR.Streamflow.Month~HydroBase
# 08240500 - TRINCHERA CREEK ABOVE TURNER'S RANCH
08240500.DWR.Streamflow.Month~HydroBase
SetFromTS(TSList=AllMatchingTSID,TSID="08241000.DWR.Streamflow.Month",
IndependentTSList=AllMatchingTSID,
IndependentTSID="08240500.DWR.Streamflow.Month",
TransferHow=ByDateTime)
```

This page is intentionally blank.

Command Reference: SetIgnoreLEZero() Indicate whether time series data values <= zero should be ignored in historical averages Version 08.16.00, 2008-07-09

The SetIgnoreLEZero() command sets the global property that indicates whether the computation of historical averages for time series should ignore values less than or equal to zero. By default, all values (other than the missing data placeholder) are used to compute averages. This command is useful when it is appropriate to ignore zero and negative values in averages, for example in cases where zero is assigned as an observation but may influence averages inappropriately. Commands that are concerned with this issue also typically provide a parameter and therefore using this global command may not be appropriate.

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

	Edit setIgnoreLEZero() Command	
	Setting the option to True results in values <= 0 being treated as missing	
	when computing historical averages.	
The default if setIgnoreLEZero() is not used is False, in which case all non-missing data are averaged.		
This command should be specified before time series read commands because averages		
	are computed immediately after reading time series.	
	Ignore <= 0 computing averages?: True 🔽	
	Command: setIgnoreLEZero(True)	
	Cancel OK	

SetIgnoreLEZero() Command Editor

The command syntax is as follows:

```
SetIgnoreLEZero(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
IgnoreLEZero	Indicates whether the computation of historical averages should ignore values less than or equal to zero, True or False.	If this command is not used, the default is False.

A sample command file is as follows:

```
SetIgnoreLEZero(IgnoreLEZero=True)
```

This page is intentionally blank.

Command Reference: SetIncludeMissingTS()

Indicate whether missing time series should automatically be added as blank time

Series Version 08.16.00, 2008-07-16

The SetIncludeMissingTS() command sets the global property that indicates whether time series that cannot be found should automatically be added as a time series with missing data. By default, time series that cannot be found generate a warning. This command is useful when processing large amounts of data, to guarantee a placeholder time series even if time series are not found. For example, use the command in the early stages of work to evaluate command sequence logic without addressing every data issue, and then remove the command when focusing on data.

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

Edit SetIncludeMissingTS() Command			
This command sets a global property indicating whether to automatically add empty time series if a time series is not read (because it is not in a database or file).			
The time series will automatically be initialized with default values and missing data.			
The default is to print a warning when a time series cannot be found.			
This command is useful in early stages of development when there is a need to focus on the overall process and not data issues.			
Automatically include missing time series?: True			
SetIncludeMissingTS(IncludeMissingTS=True)			
Command:			
Cancel OK			
SetIncludeMissingTS			

SetIncludeMissingTS() Command Editor

The command syntax is as follows:

SetIncludeMissingTS(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
IncludeMissingTS	Indicates whether time series that are not	If this command is not used, the
	found with read commands should	default is False.
	automatically be added with missing	
	data.	

A sample command file is as follows:

SetIncludeMissingTS(IncludeMissingTS=True)

This page is intentionally blank.

Command Reference: SetInputPeriod() Set the period for reading time series from files and querying from databases

Version 08.15.00, 2008-05-11

The SetInputPeriod() command sets the period used for reading time series data from files and querying data from databases. The default is to read/query all available data so that all data are available for analysis and data filling. However, a shorter period may be desirable to increase performance (e.g., when processing real-time data) or to force matching a historical period. This command replaces the SetQueryPeriod() command. See also the SetOutputPeriod() command.

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

💮 Edit SetInputPeriod() Command 🛛 🛛 🔀
The input period constrains the period when reading data from files and databases. Use this command only if a limited data period is necessary (e.g., to improve performance). Using a SetInputPeriod() command may result in incomplete data being available for data filling. Enter date/times to a precision appropriate for time series being read. For example: Year data: YYYY Month data: MM/YYYY or YYYY-MM Day data: MM/DD/YYYY or YYYY-MM-DD Hour data: MM/DD/YYYY HH or YYYY-MM-DD HH Minute data: MM/DD/YYYY HH:mm or YYYY-MM-DD HH Minute data: MM/DD/YYYY HH:mm or YYYY-MM-DD HH:mm Special values are also recognized (for all precisions): CurrentToYear = the current date to year precision CurrentToMinute = the current date/time to minute precision CurrentToMinute - 7Day = current date/time minus 7 days CurrentToMinute + 7Day = current date/time plus 7 days Leave blank to read all available data (default if SetInputPeriod() command is not used). Input period start: 1950-01
Input period end: 2000-09
Command: SetInputPeriod(InputStart="1950-01",InputEnd="2000- 09")
Cancel OK

SetInputPeriod() Command Editor

SetInputPeriod

```
SetInputPeriod(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
InputStart	 The date/time to start reading/querying time series data, one of: A date/time string (see dialog above for examples). CurrentToYear, CurrentToMonth, CurrentToDay, CurrentToHour, CurrentToMinute, indicating the current date/time to the specified precision. A Current* value +- an interval, for example: CurrentToMinute - 7Day 	None – must be specified.
InputEnd	 The date/time to end reading/querying time series data, one of: A date/time string (see dialog above for examples). CurrentToYear, CurrentToMonth, CurrentToDay, CurrentToHour, CurrentToMinute, indicating the current date/time to the specified precision. A Current* value +- an interval, for example: CurrentToMinute - 7Day An expression involving InputStart, used similar to the Current* values. 	None – must be specified.

A sample commands file for historical data from the State of Colorado's HydroBase is as follows:

```
SetInputPeriod(InputStart="1950-01",InputEnd="2000-09")
# 06754000 - SOUTH PLATTE RIVER NEAR KERSEY
06754000.DWR.Streamflow.Month~HydroBase
```

A sample commands file for real-time data is as follows:

```
SetInputPeriod(InputStart="CurrentToMinute - 14Day",
    InputEnd="CurrentToMinute + 1Hour")
# 06754000 - SOUTH PLATTE RIVER NEAR KERSEY
06754000.DWR.Streamflow-DISCHRG.Irregular~HydroBase
```

Command Reference: SetOutputPeriod()

Set the output period for time series

Version 09.08.01, 2010-09-14

The SetOutputPeriod() command sets the output period for time series. See also the SetInputPeriod() command. The period for a time series when read or created will be set to the maximum of the following periods, in order to satisfy output and data filling requirements:

- available data,
- output period (if specified),
- input period (if specified).

Specifying the output period is necessary when creating model files or filling an extended period (time series will not automatically be extended by fill commands).

The following dialog is used to edit this command and illustrates the syntax of the command. Note that the output period should always use calendar month and year, even if other than calendar year are used for output (see SetOutputYearType()).

👌 Edit SetOutpu	utPeriod() Command		
Set the global (default) output period for time series and output products.			
The time series period	The time series period after reading typically will be extended to the output period by using the missing value.		
Use a SetOutputPerio	od() command to guarantee a longe	er period when filling/extending data.	
Specify the command	l at the top of commands when fillir	ng a specific period.	
Enter date/times to a	precision appropriate for time serie	es being processed. For example:	
Year data: YYYY			
Month data: MM/	YYYY or YYYY-MM		
Day data: MM/D	D/YYYY or YYYY-MM-DD		
Hour data: MM/D	D/YYYY HH or YYYY-MM-DD HH		
Minute data: MM/(DD/YYYY HH:mm or YYYY-MM-DD H	H:mm	
Special values are als	o recognized (for all precisions):		
CurrentToYear = t	he current date to year precision		
CurrentToMinute =	the current date/time to minute p	recision	
CurrentToMinute -	7Day = current date/time minus 7	days	
CurrentToMinute + 7Day = current date/time plus 7 days			
See also the SetInputPeriod() command, which will constrain the period that is read.			
Output period start:	1950-01	Required	
Output period end:	2002-12	Required	
		·	
	SetOutputPeriod(OutputStart="1950-01",OutputEnd="2002-12")		
Command:			
Command.			
	Cano	iel OK	
		SetOutputPe	

SetOutputPeriod() Command Editor

The command syntax is as follows:

SetOutputPeriod(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
OutputStart	The date/time to start output.	None – must be specified.
OutputEnd	The date/time to end output.	None – must be specified.

A sample commands file is as follows:

Command Reference: SetOutputYearType()

Set the output year type for time series

The SetOutputYearType() command sets the global output year type for output reports and files. The default for most operations is calendar year (January through December); however, alternate year definitions may be useful. The global output year type is recognized by some common tools and commands that create output. Many write commands also allow the year type to be specified for the command. Internally, all data are managed using calendar years and are converted to different year types during output or display. The ChangeInterval() command also allows time series to be converted to annual values where the value corresponds to a year type.

The year type for output and analysis theoretically can be defined in many ways. Internally, TSTool allows the start and end year to have offsets from the calendar year. This allows the output year type to have a starting year previous to the calendar year or the same as the calendar year. A convention that is being implemented over time is to prepend Year to the year types where the start of the output year agrees with the calendar year number, and append Year to the year types where the end of the output year agrees with the calendar year number. For example, YearMayToApr would indicate that the output year is May of the calendar year to Apr of the next calendar year.

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

👌 Edit SetOut	putYearType() Command	×	
This command sets	the global output year type, which is recognized by some output commands		
(e.g., for model an	d summary output and as the default period when creating new time series).		
The following globa	al output year types are available:		
Calendar year (c	default): January to December.		
Water year: October (year - 1) to September (year) (e.g., water year 1970 is Oct 1969 to Sep 1970).			
NovToOct: November (year - 1) to October(year) (e.g., year 1970 is Nov 1969 to Oct 1970).			
Additional output y	ear types may be enabled for specific commands.		
Output year type:	Calendar 💙 Required - global output ye	ear type.	
	SetOutputYearType(OutputYearType=Calendar)		
Command:			
	Cancel OK		
		Sot	

SetOutputYearType() Command Editor

```
SetOutputYearType(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
Parameter OutputYearType	 Description The output year type, one of: Calendar – January through December. NovToOct – November of the previous calendar year to October of the current calendar year. For example, year 1970 spans Nov 1969 to Oct 1970. Water – October of the previous 	Default If this command is not specified, Calendar is the default.
	 Water - October of the previous calendar year through September of the current calendar year (and water year). For example, water year 1970 spans Oct 1969 to Sep 1970. In the future, more generic types like NovToOct may be implemented. 	

A sample commands file is as follows:

SetOutputYearType(OutputYearType=Calendar)

Command Reference: SetPatternFile()

Set the pattern file to be used with FillPattern() commands

Version 08.16.04, 2008-09-19

This command has been replaced with ReadPatternFile() - TSTool will automatically convert the command.

The SetPatternFile() command specifies a pattern file to be used with FillPattern() commands (see the FillPattern() command for more information).

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

🛃 Edit setPatternFile() 🛙	Command	×
Set the pattern file used with	n fillPattern() commands.	
The working directory is: J:V	CDSS\develop\Apps\TSTool\test\Commands\setPatternFile	
Pattern File: fill.pat		Browse
Command: setPatternFile("	'fill.pat'')	
	Add Working Directory Cancel OK	
		SetP

SetPatternFile() Command Editor

The command syntax is as follows:

```
SetPatternFile(PatternFile)
```

Command Parameters

Parameter	Description	Default
PatternFile	The path to the pattern file, which can be	None – must be specified.
	absolute or relative to the working	
	directory. The Browse button can be	
	used to select the pattern file (if a relative	
	path is desired, remove the leading path	
	after the select).	

A sample commands file is as follows:

SetPatternFile("fill.pat")

This page is intentionally blank.

Command Reference: SetProperty()

Set a property for the time series processor

Version 10.01.00, 2011-11-15

The SetProperty() command sets the value of a property used by the time series processor. The property will be available to subsequent commands that support using \${Property} notation in parameters, for example to specify filenames more dynamically. This command should not be confused with the SetTimeSeriesProperty() command, which sets a property on specific time series.

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

👌 Edit SetPr	operty() Command	X
Set a property fo	or the processor. The property ca	n be referenced in parameters of some commands using \${Property} notation.
Specify date/time	es using standard notations to app	ropriate precision (e.g., YYYY-MM-DD hh:mm:ss).
Special values al:	so are recognized for date/times (f	or all precisions):
CurrentToYea	r = the current date to year precis	ion
CurrentToMin	ute = the current date/time to min	ute precision
CurrentToMin	ute - 7Day = current date/time min	us 7 days
CurrentToMin	ute + 7Day = current date/time plu	ıs 7 days
Property name:	CurrentTime	Required - do not use spaces \$, { or } in name.
Property type:	DateTime 🔽	Required - to ensure proper initialization and checks.
Property value:	CurrentToMinute	Required - property value, can use \${Property}.
Command:	Command: SetProperty(PropertyName="CurrentTime",PropertyType=DateTime,Prop ertyValue="CurrentToMinute")	
Cancel OK		
		SetProperty

SetProperty() Command Editor

```
SetProperty(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
PropertyName	The property name.	None – must be specified.
PropertyType	The property type, used for validation, one of:	None – must be specified.
	• DateTime – a date/time.	
	• Double – a floating point number	
	• Integer – an integer	
	• String – a string	
PropertyValue	The value of the property, adhering to property	None – must be specified.
	type constraints. Date/time properties should be	
	specified using standard formats such as	
	"YYYY-MM-DD hh:mm:ss", to an appropriate	
	precision. Special date/time syntax is	
	recognized, as shown in the above figure.	
	Global properties can be used with the	
	\${Property} syntax.	

A sample commands file is as follows:

SetProperty(PropertyName="Scenario",PropertyType=String,PropertyValue="Likely")

Command Reference: SetTimeSeriesPropertiesFromTable()

Set time series properties using values in a table

ersion 10.07.00, 2012-04-18

The SetTimeSeriesPropertiesFromTable() command sets user-defined time series properties using values in a table. This is useful, for example, when additional attributes are available for locations associated with time series. The time series can then be selected for processing by matching properties with the SelectTimeSeries() command.

The following dialog is used to edit the command and illustrates the command syntax (in this case the location part of the time series identifier is used to match the contents of the "loc" column in the table).

🜢 Edit SetTimeSeriesPropertiesFromTable() Command 🛛 🛛 🔀			
Set time series properties using match	ng input from a table.		
For example, set properties for a local	ion associated with the time series.		
The table value is determined from a r	w with a matching time series identifier (TSID) and by specifying the column from which to get a value.		
TS list:	Optional - indicates the time series to process (default=AlITS).		
TSID (for TSList=AllMatchingTSID);			
EnsembleID (for TSList=EnsembleID):			
Table ID:	Table1 Required - table to process.		
Table TSID column:	loc Required - column name for TSID.		
Format of TSID:	Select Specifier 🔽 => %L Optional - use %L for location, etc. (default=alias or TSID).		
Table input columns:	Scenario,Status Required - column names from which to set properties.		
Time series property names:	Optional - property names in time series (default=input columns)		
Command:	SetTimeSeriesPropertiesFromTable(TableID="Table1",TableTSIDColumn=" loc",TableTSIDFormat="%L",TableInputColumns="Scenario,Status")		
Cancel OK			

SetTimeSeriesPropertiesFromTable() Command Editor

The command syntax is as follows:

SetTimeSeriesPropertiesFromTable(Parameter=Value,...)

Parameter	Description	Default
TSList	Indicates the list of time series to be	AllTS
	processed, one of:	
	• AllMatchingTSID – all time	
	series that match the TSID (single	
	TSID or TSID with wildcards) will	
	be modified.	
	• AllTS – all time series before the	
	command.	
	• EnsembleID – all time series in	
	the ensemble will be modified (see	
	the EnsembleID parameter).	
	• FirstMatchingTSID – the first	
	time series that matches the TSID	
	(single TSID or TSID with	
	wildcards) will be modified.	
	• LastMatchingTSID – the last	
	time series that matches the TSID	
	(single TSID or TSID with	
	wildcards) will be modified.	
TSID	The time series identifier or alias for the	Required if TSList=*TSID
	time series to be modified, using the *	-
	wildcard character to match multiple	
	time series.	
EnsembleID	The ensemble to be modified, if	Required if
	processing an ensemble.	TSList=EnsembleID
TableID	The identifier for the table that contains	None – must be specified.
	properties.	
TableTSIDColumn	Table column name that is used to	None – must be specified.
	match the time series identifier for	
	processing.	
TableTSIDFormat	The specification to format the time	Time series alias if available, or
	series identifier to match the ISID	otherwise the time series identifier.
	column. Use the format choices and	
	identifier	
TableInputColumna	The name(s) of the column(s) to supply	Nona must be specified
TableInpuccolumns	properties for the matching time series	None – must be specified.
	Separate column names with commas	
TSPropertvNames	The names(s) of the time series	Same as Table Input Columns
	properties. Separate names by commas.	
	leave blank to use the corresponding	
	TableInputColumns name, or use	
	the special value TS: Description	
	to set the time series description.	

Command Parameters

Command Reference: SetTimeSeriesProperty()

Set time series properties

Version 09.09.00, 2010-09-23

The SetTimeSeriesProperty() command sets the value of time series properties. Properties that are used to uniquely identify the time series cannot be set because other commands need to utilize this information to reference the time series; therefore, properties that cannot be changed include the location identifier, data source, data type, interval, and scenario. See also the SetTimeSeriesPropertiesFromTable() and SelectTimeSeries() commands.

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

👌 Edit SetTimeSeriesProperty() Command 🛛 🛛 🔀			×	
Set time series properties (metadata). The identifier information cannot be changed because it is used to define workflow processing. Several specific properties are built-in, and user-defined properties also can be set (see Property below) .				
TS list:	×		Optional - indicates the time series to process (default=AlITS).	
TSID (for TSList=AllMatchingTSID):				~
EnsembleID (for TSList=EnsembleID):				Y
Description:	Insert: -	Select Specifier 🛛 💙	Optional - use %L for location, etc.	
Data units:	AF/M		Optional - data units (does not change data values).	
Missing value:			Optional - missing data value (does not change data values).	
Are data editable?:		*	Optional - for interactive edits (default=False).	
Property name:			Required if user-defined property is set.	
Property type:	~		Required if user-defined property is set - to ensure proper initialization	on.
Property value:			Required if user-defined property is set.	
	SetTimeSeriesPro	operty(Units="A)	7/M")	
Command:				
Cancel OK				
			SetTimeSeriesPro	pertv

SetTimeSeriesProperty() Command Editor

The command syntax is as follows:

```
SetTimeSeriesProperty(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
TSList	 Indicates the list of time series to be processed, one of: AllMatchingTSID – all time series that 	AllTS
	wildcards) will be modified.	
	 AllIS – all time series before the command. EnsembleID – all time series in the ensemble will be modified. 	
	• FirstMatchingTSID – the last time	

	series that matches the TSID (single TSID or TSID with wildcards) will be modified.	
	• LastMatchingTSID – the last time series	
	that matches the TSID (single TSID or TSID	
	with wildcards) will be modified.	
	• SelectedTS – the time series are those	
	selected with the SelectTimeSeries()	
	command.	
TSID	The time series identifier or alias for the time	Required if
	series to be modified, using the * wildcard	TSList=*TSID.
	character to match multiple time series.	
EnsembleID	The ensemble to be modified, if processing an	Required if
	ensemble.	TSList=EnsembleID.
Description	The description to assign to the time series. Use	None.
	the format choices and other characters to define	
	a unique alias.	
Units	The data units to assign to the time series. The	None.
	units should agree with the time series data	
	values.	
Editable	If set to True, then graphing the time series will	False
	enable interactive editing features, including the	
	ability to save the edited time series.	
PropertyName	Name of user-defined property.	
PropertyType	Property type, to ensure proper initialization and	Required if
	data check.	PropertyName is
		specified.
PropertyValue	Value for property, adhering to the property type	Required if
	requirements.	PropertyName is
	1	specified.

A sample command file to set a property for time series read from a StateMod file is as follows:

```
ReadStateMod(InputFile="Data\ym2004.ddh")
SetTimeSeriesProperty(Units="AF/M")
```

Command Reference: SetToMax()

Set data values to the maximum of values from one or more time series

Version 08.16.04, 2008-09-25

The SetToMax() command sets a time series to contain, for each time step, the maximum of its own values and those of one or more additional (independent) time series. This command replaces the SetMax() command. See also the SetToMin() command.

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

Set the time series data values to the maximum of itself and one or more (independent) time series. Time series to receive results: 08236000.DWR.Streamflow.Month Independent TS List: SpecifiedTSID Indicates the time series to process (default=AIITS Independent TSID (for Independent TSList=AIMatchingTSID): 08236500.DWR.Streamflow.Month	🛃 Edit SetToMax() Command	×
Time series to receive results: 08236000.DWR.Streamflow.Month Independent TS List: SpecifiedTSID Indicates the time series to process (default=AIITS Independent TSID (for Independent TSList=AIIMatchingTSID); 08236500.DWR.Streamflow.Month Independent	Set the time series data values to the maximum of itself and one or	more (independent) time series.
Independent TS List: SpecifiedTSID Indicates the time series to process (default=AIITS Independent TSID (for Independent TSList=AIIMatchingTSID): 08236500.DWR.Streamflow.Month	Time series to receive result:	8 08236000.DWR.Streamflow.Month
Independent TSID (for Independent TSList=AllMatchingTSID): 08236500.DWR.Streamflow.Month	Independent TS Lis	t: SpecifiedTSID Indicates the time series to process (default=AIITS).
Independent EncembleD: (for Independent TCI int-EncembleD);	Independent TSID (for Independent TSList=AllMatchingTSID): 08236500.DWR.Streamflow.Month
	Independent EnsembleID (for Independent TSList=EnsembleID	ji 🔽
Independent specified TSID (for IndependentTSList=SpecifiedTSID): 08236000.DWR.Streamflow.Month 08236500.DWR.Streamflow.Month	Independent specified TSID (for IndependentTSList=SpecifiedTSID): 08236000.DWR.Streamflow.Month 08236500.DWR.Streamflow.Month
Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Command: Com	Command	SetToMax(TSID="08236000.DWR.Streamflow.Month",Indep endentTSList=SpecifiedTSID,IndependentTSID="0823650 0.DWR.Streamflow.Month")
Cancel OK	Car	

SetToMax() Command Editor

```
SetToMax(Parameter=Value,...)
```

Command Parameters

Parameter	Description Default	
TSID	The time series identifier or alias for the time	None – must be specified.
	series to be modified.	
IndependentTSList	IndependentTSList Indicates how the list of time series is specified, AllTS (the	
	one of:	receiving the result will
	• AllTS – all time series before the command.	not be checked)
	• AllMatchingTSID – all time series that	
	match the IndependentTSID (single	
	TSID or TSID with wildcards).	
	• EnsembleID – the time series from the	
	specified ensemble will be processed.	
	• FirstMatchingTSID – the first time	
	series that matches the TSID (single TSID or	
	TSID with wildcards) will be processed.	
	• LastMatchingTSID – the last time series	
	that matches the TSID (single TSID or TSID	
	with wildcards) will be processed.	
	• SelectedTS – the time series are those	
	selected with the SelectTimeSeries()	
	command.	
	• SpecifiedTSID – the specified list of	
	time series given by the	
	IndependentTSID parameter.	D 110
IndependentISID	If the IndependentTSList=	Required if
	SpecifiedTSID, provide the list of time	TSList=*TSID.
	series identifiers (or alias) to process, separated	
	by commas. If the independent is list	
	FirstMatchingTSID, or	
	Last Matching TSID, on	
	a TSID with wildcards	
Independent	Ensemble identifier	Required if
EnsembleID	TSList=Ensemble	

A sample command file to process data from the State of Colorado's HydroBase database is as follows:

```
# 08236000 - ALAMOSA RIVER ABOVE TERRACE RESERVOIR
08236000.DWR.Streamflow.Month~HydroBase
# 08236500 - ALAMOSA RIVER BELOW TERRACE RESERVOIR
08236500.DWR.Streamflow.Month~HydroBase
SetToMax(TSID="08236000.DWR.Streamflow.Month",
IndependentTSList=SpecifiedTSID,
IndependentTSID="08236500.DWR.Streamflow.Month")
```

Command Reference: SetToMin()

Set data values to the minimum of values from one or more time series

Version 08.16.04, 2008-09-25

The SetToMin() command sets a time series to contain, for each time step, the minimum of its own values and those of one or more additional (independent) time series.

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

Set the time series data values to the minimum of itself and one or more (independent) time series. Time series to receive results 08236000 DWR.Streamflow.Month Independent TS List SpecifiedTSID Independent TSID (for Independent TSList=AllMatchingTSID) 08236500 DWR.Streamflow.Month Independent EnsembleID (for Independent TSList=EnsembleID): 08236000 DWR.Streamflow.Month Independent specified TSID (for IndependentTSList=SpecifiedTSID): 08236000 DWR.Streamflow.Month 082366000 DWR.Streamflow.Month 08236500 DWR.Streamflow.Month 08236600 DWR.Streamflow.Month 08236500 DWR.Streamflow.Month 0.DWR.Streamflow.Month 08236500 DWR.Streamflow.Month 0.DWR.Streamflow.Month'') 0.DWR.Streamflow.Month'')	🛃 Edit SetToMin() Command	×
Time series to receive results 08236000 DVWR.Streamflow.Month Independent TS List SpecifiedTSID Independent TSID (for Independent TSList=AIIMatchingTSID): 08236500 DWR.Streamflow.Month Independent EnsembleD (for Independent TSList=EnsembleD): 08236000 DWR.Streamflow.Month Independent specified TSID (for Independent TSList=EnsembleD): 08236500 DWR.Streamflow.Month Independent specified TSID (for IndependentTSList=SpecifiedTSID): 08236500 DWR.Streamflow.Month 08236500 DWR.Streamflow.Month 08236500 DWR.Streamflow.Month 08236500 DWR.Streamflow.Month 08236500 DWR.Streamflow.Month 08236500 DWR.Streamflow.Month 08236500 DWR.Streamflow.Month 08236500 DWR.Streamflow.Month 08236500 DWR.Streamflow.Month 0.0000.DWR.Streamflow.Month 08236500 DWR.Streamflow.Month	Set the time series data values to the minimum of itself and one or more (independent) time series.	
Independent TS List: SpecifiedTSID Indicates the time series to process (default=AIITS). Independent TSID (for Independent TSList=AIIMatchingTSID) D8236500.DWR.Streamflow.Month Independent EnsembleID (for Independent TSList=EnsembleID) 08236500.DWR.Streamflow.Month Independent specified TSID (for IndependentTSList=SpecifiedTSID) 08236500.DWR.Streamflow.Month 08236500.DWR.Streamflow.Month 08236500.DWR.Streamflow.Month 08236500.DWR.Streamflow.Month 08236500.DWR.Streamflow.Month 08236500.DWR.Streamflow.Month 08236500.DWR.Streamflow.Month 0.0WR.Streamflow.Month 08236500.DWR.Streamflow.Month 0.DWR.Streamflow.Month 08236500.DWR.Streamflow.Month	Time series to receive results:	08236000.DVVR.Streamflow.Month
Independent TSID (for Independent TSList=AllMatchingTSID): D8236500.DWR.Streamflow.Month Independent EnsembleID (for Independent TSList=EnsembleID): Independent specified TSID (for IndependentTSList=SpecifiedTSID): D82366000.DWR.Streamflow.Month 08236500.DWR.Streamflow.Month 0.DWR.Streamflow.Month 0.DWR 0	Independent TS List:	SpecifiedTSID Indicates the time series to process (default=AIITS).
Independent EnsembleD (for Independent TSList=EnsembleD): Independent specified TSID (for IndependentTSList=SpecifiedTSID): Independent Specified TSID (for IndependentTSList=SpecifiedTSID, IndependentTSID="08236500"): Independent Specified TSID (for Independent Specified TSID): Independent Specified TSID (for Independent Specified Sp	Independent TSID (for Independent TSList=AllMatchingTSID):	08236500.DWR.Streamflow.Month
Independent specified TSID (for IndependentTSList=SpecifiedTSID): 08236500.DWR.Streamflow.Month 08236500.DWR.Streamflow.Month 08236500.DWR.Streamflow.Month 08236500.DWR.Streamflow.Month 08236500.DWR.Streamflow.Month 08236500.DWR.Streamflow.Month 08236500.DWR.Streamflow.Month	Independent EnsembleID (for Independent TSList=EnsembleID):	V
SetToMin(TSID="08236000.DWR.Streamflow.Month",Indep Command: endentTSList=SpecifiedTSID,IndependentTSID="0823650 0.DWR.Streamflow.Month") Cancel OK	Independent specified TSID (for IndependentTSList=SpecifiedTSID):	08236000.DWR.Streamflow.Month 08236500.DWR.Streamflow.Month
Cancel OK	Command:	SetToMin(TSID="08236000.DWR.Streamflow.Month",Indep endentTSList=SpecifiedTSID,IndependentTSID="0823650 0.DWR.Streamflow.Month")
	Cance	

SetToMin() Command Editor

```
SetToMin(Parameter=Value,...)
```

Command Parameters

Parameter	Description Default	
TSID	The time series identifier or alias for the time	None – must be specified.
	series to be modified.	
IndependentTSList	IndependentTSList Indicates how the list of time series is specified, AllTS (the	
	one of:	receiving the result will
	• AllTS – all time series before the command.	not be checked)
	• AllMatchingTSID – all time series that	
	match the IndependentTSID (single	
	TSID or TSID with wildcards).	
	• EnsembleID – the time series from the	
	specified ensemble will be processed.	
	• FirstMatchingTSID – the first time	
	series that matches the TSID (single TSID or	
	TSID with wildcards) will be processed.	
	• LastMatchingTSID – the last time series	
	that matches the TSID (single TSID or TSID	
	with wildcards) will be processed.	
	• SelectedTS – the time series are those	
	selected with the SelectTimeSeries()	
	command.	
	• SpecifiedTSID – the specified list of	
	time series given by the	
IndopondontTSID	Independent ISID parameter.	De mine dif
Independencisib	If the independentTSList=	
	series identifiers (or alias) to process separated	ISLISC=^ISID.
	by commas. If the Independent TSList	
	parameter is AllMatchingTSID	
	FirstMatchingTSID or	
	LastMatchingTSID, specify a single TSID or	
	a TSID with wildcards.	
Independent	Ensemble identifier.	Required if
EnsembleID	TSList=EnsembleID	

A sample command file to process data from the State of Colorado's HydroBase database is as follows:

```
# 08236000 - ALAMOSA RIVER ABOVE TERRACE RESERVOIR
08236000.DWR.Streamflow.Month~HydroBase
# 08236500 - ALAMOSA RIVER BELOW TERRACE RESERVOIR
08236500.DWR.Streamflow.Month~HydroBase
SetToMin(TSID="08236000.DWR.Streamflow.Month",
IndependentTSList=SpecifiedTSID,
IndependentTSID="08236500.DWR.Streamflow.Month")
```

Command Reference: SetWarningLevel()

Set level for warning messages

Version 08.16.00, 2008-08-24

The SetWarningLevel() command sets the warning levels for the screen and log file. Higher warning levels are useful for troubleshooting commands. The higher the level, the more messages will be generated. This command can be used multiple times, for example to isolate a problem. Currently the warning level applies to all components. In the future logging control may be grouped by component. Levels are not completely consistent but the following guidelines can be followed:

0 = no messages

1 = important messages generated in applications

2 =important messages generated in commands

3+ = messages generated in commands that may explain other problems

10+ = messages in processing code that may still be useful to end users

30+ = low-level messages, for example generated while reading from files or databases

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

🛃 Edit SetWarningLe	evel() command	×
Set the warning level for	screen and/or log file warning messages.	
Setting the warning level	to a higher number prints more warning information.	
Warning information is us	sed for troubleshooting.	
Warning levels can be in	creased before and decreased after specific commands to troublesheet th	e commands.
Screen warning level:	0=none, 100=all, blan	k=no change.
Log file warning level: 10	0 0=none, 100=all, blan	k=no change.
Command:	etWarningLevel(LogFileLevel="10")	
	Cancel OK	
		SetWa

SetWarningLevel() Command Editor

SetWarningLevel(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
ScreenLevel	The warning level for the screen $(0+)$.	Keep previous setting.
LogFileLevel	The warning level for the log file $(0+)$.	Keep previous setting

A sample commands file is as follows:

SetWarningLevel(ScreenLevel=1,LogFileLevel=10)	

Command Reference: SetWorkingDir()

Set working directory Version 10.00.02, 2011-05-23

The SetWorkingDir() command sets the working directory for following commands. The working directory is normally set in one of the following ways, with the current setting being defined by the most recent action that has occurred:

- 1. The startup directory for the TSTool program,
- 2. The directory containing the most recently opened or saved command file.
- 3. The directory specified by a SetWorkingDir() command,
- 4. The directory specified by *File...Set Working Directory*.

In most cases, a SetWorkingDir() command is not needed and should be avoided because it may complicate commands and troubleshooting. However, for complicated command files that process data in multiple directories, it may be useful to change the working directory during processing. Setting the working directory to an absolute path causes all relative paths for input and output files to be appended to the working directory. Relative paths that use "../" can be specified to move up and down a directory tree. The current working directory during processing is reset to the initial working directory (the location of the command file) each time that the commands are run.

In any case, it is recommended that paths used in command parameters be specified using relative paths (relative to the command file) so that command files and associated data files can be easily moved from one computer to another.

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

👌 Edit SetWorkingDi	ir() Command		
Use of this command is Set the working directory, If browsing for files while e The command file folder is I The working directory is cu	i discouraged - comm which will be prepended t diting other commands, p the initial working director rrently: C:\Develop\TSTo	ands may generate warnings after editing, but will run correctly. to relative paths in commands. baths may need to be converted to be relative to the working directory. ry. bol_SourceBuild\TSTool\test\regression\commands\general\SetWorkingDir	
Working directory:	C:\TMP		Browse
Run mode:	~	Optional - does command run in GUI and/or batch? (default=GUIAndBatch)	
Run on operating system:	~	Optional - operating system to run on (default=All)	
	SetWorkingDir(WorkingDir="C:\TMP")		
Command:			
		Cancel OK	
			SetWorkingDi

SetWorkingDir() Command Editor

SetWorkingDir(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
WorkingDir	The working directory that should be	None – must be specified.
	used. Specify a relative path (e.g., "")	
	to adjust the current working directory.	
RunMode	Indicate the run mode in which the	GUIAndBatch
	command should be applied, one of:	
	• GUIOnly – the command applies	
	only to interactive runs	
	• GUIAndBatch – the command	
	applies to interactive and batch runs	
	• BatchOnly – the command applies	
	to batch runs only	

A sample command file is as follows:

SetWorkingDir(WorkingDir="C:\temp")

Command Reference: ShiftTimeByInterval()

Shift time series data by one or more time intervals

Version 08.15.00, 2008-05-11

The ShiftTimeByInterval() command shifts a time series in time. This command can be used to perform a simple shift (e.g., to shift hourly data because the Disaggregate() command did not result in data being set at the desired hours) and to perform simple routing.

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

Edit ShiftTimeByInterval() Command
Shift a time series by factoring time step values (e.g., to lag a streamflow time series).
The shift data consists of interval offset and weight pairs. For example:
-2,1.0
shifts the data from interval i-2 to interval i (no weighting).
The example:
0,.5,1,.5
shifts the data by setting the value at i to .5 its previous value + .5 the i+1 value.
Specify as many pairs as needed. The period is not automatically extended.
The resulting value is set to missing if one or more input values are missing.
TS list: AllMatchingTSID 🔽 Indicates the time series to process (default=AlITS).
TSID (for TSList=AllMatchingTSID): 08213500.DWR.Streamflow.Day
EnsembleID (for TSList=EnsembleID):
Shift offset, weight pairs: -1,1
ShiftTimeByInterval(TSList=AllMatchingTSID,TSID="08
Command: 213500.DWR.Streamflow.Day",ShiftData="-1,1")
Cancel OK

ShiftTimeByInterval() Command Editor

The command syntax is as follows:

ShiftTimeByInterval(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of:	Allts
	• AllMatchingTSID – all time series that match the TSID	
	(single TSID or TSID with wildcards) will be modified.	
	• AllTS – all time series before the command.	
	• EnsembleID – all time series in the ensemble will be	

Parameter	Description	Default
	 modified. LastMatchingTSID – the last time series that matches the TSID (single TSID or TSID with wildcards) will be modified. SelectedTS – the time series are those selected with the SelectTimeSeries() command. 	
TSID	The time series identifier or alias for the time series to be modified, using the * wildcard character to match multiple time series.	TSID or EnsembleID must be specified if identifiers are being matched.
EnsembleID	The ensemble to be modified, if processing an ensemble.	TSID or EnsembleID must be specified if identifiers are being matched.
ShiftData	Interval, multiplier tuples to apply to the data to perform the shift. All values should be separated by commas. An interval of -1 indicates that the previous time step should be shifted to the current time step. If the interval is -1 and the multiplier is 1, the previous time step is shifted to the current and multiplied by 1, effectively shifting the time series by one interval.	None – at least 1 value,multiplier tuple must be specified.

A sample command file to shift data from the State of Colorado's HydroBase is as follows:

```
# 08213500 - RIO GRANDE RIVER AT THIRTY MILE BRIDGE NEAR CREEDE
08213500.DWR.Streamflow.Day~HydroBase
ShiftTimeByInterval(TSList=AllMatchingTSID,TSID="08213500.DWR.Streamflow.Day",
ShiftData="-1,1")
08213500.DWR.Streamflow.Day~HydroBase
```



Results from ShiftTimeByInterval() Command

Command Reference: SortTimeSeries()

Sort time series by their identifiers

Version 08.15.00, 2008-05-11

The SortTimeSeries() command sorts the time series alphabetically using the time series identifier. This command is useful for ordering time series before writing output, for example to facilitate comparison with another version of the output or to be consistent with other data files.

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

Edit SortTimeSeries() command		
This command sorts time series. Currently the sort is alphabetical by the full identifier.		
Command:		
Cancel OK		

SortTimeSeries() Command Editor

The command syntax is as follows:

```
SortTimeSeries(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
	Currently no parameters are available for this command.	

A sample command file using data from the State of Colorado's HydroBase is as follows:

06759100 - BIJOU CREEK NEAR FT. MORGAN, CO. 06759100.USGS.Streamflow.Month~HydroBase # 06759000 - BIJOU CREEK NEAR WIGGINS, CO. 06759000.USGS.Streamflow.Month~HydroBase # BOXHUDCO - BOX ELDER CREEK NEAR HUDSON, CO BOXHUDCO.DWR.Streamflow.Month~HydroBase # 06756500 - CROW CREEK NEAR BARNSVILLE, CO. 06756500.USGS.Streamflow.Month~HydroBase # 06758300 - KIOWA CREEK AT BENNETT, CO. 06758300.USGS.Streamflow.Month~HydroBase # 06758000 - KIOWA CREEK AT ELBERT, CO. 06758000.USGS.Streamflow.Month~HydroBase # 06757600 - KIOWA CREEK AT K-79 RES, NEAR EASTONVILLE, CO. 06757600.DWR.Streamflow.Month~HydroBase # 06758200 - KIOWA CREEK AT KIOWA, CO. 06758200.USGS.Streamflow.Month~HydroBase # 06753400 - LONETREE CREEK AT CARR, CO. 06753400.USGS.Streamflow.Month~HydroBase # 06753990 - LONETREE CREEK NEAR GREELEY, CO. 06753990.USGS.Streamflow.Month~HydroBase # 06753500 - LONETREE CREEK NEAR NUNN, CO. 06753500.USGS.Streamflow.Month~HydroBase # 06759910 - SOUTH PLATTE RIVER AT COOPER BRIDGE NEAR BALZAC 06759910.DWR.Streamflow.Month~HydroBase # 06759500 - SOUTH PLATTE RIVER AT FORT MORGAN 06759500.USGS.Streamflow.Month~HydroBase # 06756995 - SOUTH PLATTE RIVER AT MASTERS, CO. 06756995.USGS.Streamflow.Month~HydroBase # 06757000 - SOUTH PLATTE RIVER AT SUBLETTE, CO. 06757000.USGS.Streamflow.Month~HydroBase # 06754000 - SOUTH PLATTE RIVER NEAR KERSEY 06754000.DWR.Streamflow.Month~HydroBase # 06758500 - SOUTH PLATTE RIVER NEAR WELDONA 06758500.DWR.Streamflow.Month~HydroBase # 06758100 - WEST KIOWA CREEK AT ELBERT, CO. 06758100.USGS.Streamflow.Month~HydroBase SortTimeSeries()

Command Reference: StartLog() (Re)start the log file Version 09.08.01, 2010-09-14

The StartLog() command (re)starts the log file. It is useful to insert this command as the first command in a command file, in order to persistently record the results of processing. A useful standard is to name the log file the same as the command file, with an additional .log extension, and this convention is enforced by default. A date or date/time can optionally be added to the log file name.

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

👌 Edit S	tartLog() command					
(Re)start th	(Re)start the log file. This is useful when it is desirable to have a log file saved for a commands file.					
A blank log file name will restart the current file.						
The log file can be specified using a full or relative path (relative to the working directory).						
The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\StartLog						
The Browse button can be used to select an existing file to overwrite.						
Specifying a suffix for the file will insert the suffix before the "log" file extension.						
Log file:	Example_StartLog.log	Browse				
Suffix:		Optional - suffix for log file (blank=none).				
	StartLog(LogFile="Example_StartLog.log")					
Command:						
Add Working Directory Cancel OK						
StartLoc						

StartLog() Command Editor

StartLog(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
LogFile	The name of the log file to write surrounded by double quotes. The extension of <i>.log</i> will automatically be added, if not specified.	If not specified, the existing file will be restarted.
Suffix	 Indicates that a suffix will be added before the .log extension, one of: Date - add a date suffix of the form YYYYMMDD. DateTime - add a date/time suffix of the form YYYYMMDD_HHMMSS. This is useful for automatically archiving logs corresponding to commands files, to allow checking the output at a later time. However, generating date/time stamped log files can increase the amount of disk space that is used. 	Do not add the suffix.

A sample command file to process State of Colorado HydroBase data is as follows (the Add() command will generate an error because the units of the time series are incompatible):

StartLog(LogFile="Example_StartLog.log")
06753400 - LONETREE CREEK AT CARR, CO.
06753400.USGS.Streamflow.Month~HydroBase
1179 - BYERS 5 ENE
1179.NOAA.Precip.Month~HydroBase
Add(TSID="06753400.USGS.Streamflow.Month",AddTSList=AllTS,HandleMissingHow="IgnoreMissing")
Command Reference: StartRegressionTestResultsReport()

Start a report file to contain regression test results

Version 08.15.00, 2008-05-11

The StartRegressionTestResultsReport() command starts a report file to be written to as regression tests are run. The CreateRegressionTestCommandFile() automatically inserts this command. The CompareFiles() and CompareTimeSeries() commands will write to this file if it is available.

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

🌑 Edit StartRegr	essionTestResultsReport() command 🛛 🔀
(Re)start the regres	sion test results report file, written to by the CompareFiles() command.
The report file can b	e specified using a full or relative path (relative to the working directory).
The working director	$ry is: C: \verb"Develop" (TSTool_SourceBuild" (TSTool" test" regression") User Manual Examples (Test Cases) Command Reference (Start Regression Test Results Report test (Start Regression)) and the start Regression (Start Regression) and the start Regression (Start Regression)) and the start Regression (Start Regression) and the start Regression (Start Regression)) and the start Regression (Start Regression) and the start Regression (Start Regression)) and the start Regression (Start Regression) and the start Regression (Start Regression)) and the start Regression (Start Regression) and the$
The Browse button	can be used to select an existing file to overwrite.
Report (output) file:	RunRegressionTest_commands_general.TSTool.out.txt Browse
	StartRegressionTestResultsReport(OutputFile="RunRegressionTest_commands_general.TSTool.out.txt")
Command:	
	Remove Working Directory Cancel OK
	StartRegressionTestResultsRepor

StartRegressionTestResultsReport() Command Editor

The command syntax is as follows:

StartRegressionTestResultsReport(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
OutputFile	The name of the report file, enclosed in double quotes if	None – must be
	the file contains spaces or other special characters. A	specified.
	path relative to the command file can be specified.	

See the RunCommands () documentation for how to set up a regression test. The following command file illustrates how to start the results report:

```
StartRegressionTestResultsReport(
    OutputFile="RunRegressionTest_commands_general.TSTool.out.txt")
...
RunCommands(InputFile="..\..\commands\general\ReadStateMod\Test_ReadStateMod_1.TSTool")
...
```

Each of the above command files should produce expected time series results, without warnings. If any command file unexpectedly produces a warning, a warning will also be visible in TSTool. The issue can then be evaluated to determine whether a software or configuration change is necessary. An example of the output file is:

SUCCESS	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\add\Test_Add_1.TSTool
SUCCESS	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\addConstant\Test_AddConstant_1.TSTool
SUCCESS	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\adjustExtremes\Test_AdjustExtremes_1.TSTool
SUCCESS	
C:\Develop\	,TSTool_SourceBuild\TSTool\test\regression\commands\general\analyzePattern\Test_AnalyzePattern_FromMonthDataValues.TSTool
SUCCESS	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ARMA\Test_ARMA_1.TSTool
SUCCESS	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\blend\Test_Blend_1.TSTool
SUCCESS	
C:\Develop\	TSTool_SourceBuild\TSTool\test\regression\commands\general\ChangeInterval\Test_ChangeInterval_DayMean_To_MonthMean.TSTool
SUCCESS	$\verb C:\Develop TSTool_SourceBuild TSTool test regression commands general ChangePeriod Test_ChangePeriod_1.TSTool test regression commands general ChangePeriod_Test_ChangePeriod_1.TSTool test regression test regression test regression test $
SUCCESS	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\compareTimeSeries\Test_AllDifferent.TSTool
SUCCESS	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\compareTimeSeries\Test_AllSame.TSTool
SUCCESS	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\convertDataUnits\Test_ConvertDataUnits_1.TSTool
SUCCESS	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\Copy\Test_Copy_1.TSTool
SUCCESS	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\CreateEnsemble\Test_CreateEnsemble_1.TSTool
FAILURE	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\CreateFromList\Test_CreateFromList_1.TSTool
WARNING	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\CreateTraces_Alias\Test_CreateTraces_1.TSTool
SUCCESS	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\cumulate\Test_Cumulate_1.TSTool
SUCCESS	
C:\Develop\	$\tt TSTool_SourceBuild\TSTool\test\regression\commands\general\DeselectTimeSeries\Test_DeselectTimeSeries_1.TSTool$
SUCCESS	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\Disaggregate_Alias\Test_Disaggregate_1.TSTool
SUCCESS	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\divide\Test_Divide_1.TSTool
WARNING	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\fillCarryForward\Test_FillCarryForward_1.TSTool
SUCCESS	C:\Develop\TSTool SourceBuild\TSTool\test\regression\commands\general\fillConstant\Test FillConstant Day.TSTool

Command Reference: StateModMax()

Compute the maximum of time series in two StateMod files

Version 08.16.04, 2008-09-23

A StateModMax() command performs the following actions:

- 1. Read all time series from one StateMod time series file,
- 2. Read all time series from a second StateMod time series file,
- 3. Generate a list of time series that contains the maximum values comparing matching time series (using the location identifier). The first list is updated and the second list is discarded.

This command is useful, for example, when creating a demand time series file that is to be the maximum of historical diversions and irrigation water requirement divided by an average efficiency. It is assumed that the specified time series have matching identifiers (the first file is used as the master list) and have consistent units and data intervals. After the time series have been processed, they can be viewed or written out as a new StateMod file (see the WriteStateMod() command).

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

Edit StateModMax() Command	×
Create a list of time series where each time series contains the maximum values for same-identifier time series from two S	StateMod files.
Typically all results are then written with other commands.	
This command is useful when computing StateMod demands as the maximum of historical diversions and (irrigation water	requirement)/efficiency.
Specify a full or relative path (relative to working directory).	
$\label{eq:constraint} The \ working \ directory \ is: \ C: \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$	ndReference\StateModMax
First StateMod file to read: rgTVV.ddh	Browse
Second StateMod file to read: rgTVVC_prelim.ddm	Browse
StateModMax(InputFilel="rgTW.ddh",InputFile2="rgTWC_prelim.ddm"	')
Add Working Directory (File 1) Add Working Directory (File 2) Cancel OK	

StateModMax() Command Editor

The command syntax is as follows:

StateModMax(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
InputFile1	The name of the first StateMod time	None – must be specified.
	series file to read, surrounded by double	
	quotes. The path to the file can be	
	absolute or relative to the working	
	directory.	
InputFile2	The name of the second StateMod time	None – must be specified.
	series file to read, which must have the	
	same data interval and units as the first	
	file.	

A sample command file is as follows:

```
StateModMax("rgTW.ddh","rgTWC_prelim.ddm")
WriteStateMod("rgTW.ddm",*)
```

Command Reference: Subtract()

Subtract one or more time series from another time series

Version 08.15.00, 2008-05-12

The Subtract() command subtracts time series of the same interval. The receiving time series will have data values set to its original values minus the data values in the indicated time series. If an ensemble is being processed, another ensemble can be subtracted, a single time series can be subtracted from all time series in the ensemble, or a list of time series can be subtracted from the ensemble (the number in the list must match the number of time series in the ensemble).

This command will generate an error if the time series do not have compatible units. If the units are compatible but are not the same (e.g., IN and FT), then the units of the part will be converted to the units of the result before subtraction. Missing data in the parts can be ignored (do not set the result to missing) or can set missing values in the result. The user should consider the implications of ignoring missing data. Time series being subtracted must have the same data interval.

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

Edit Subtract() Command	×
Subtract one or more time series from a time series (or enser	nble of time series). The receiving time series (or ensemble) is modified.
The time series to be subtracted are selected using the TS lis	t parameter:
AllMatchingTSID - subtract all previous time series with mate	ching identifiers.
AIITS - subtract all previous time series.	
SelectedTS - subtract time series selected with selectTimeS	Series() commands
SpecifiedTSID - subtract time series selected from the list be	elow
Time series to receive results:	0100501.DVVR.DivTotal.Month
Ensemble to receive results:	
Time series to subtract (SubtractTSList):	SpecifiedTSID Indicates the time series to process (default=AIITS).
Subtract TSID (for TSList=AllMatchingTSID):	0100503.DWR.DivTotal.Month
Add EnsembleID (for SubtractTSList=EnsembleID):	V
Subtract specified TSID (for SubtractTSList=SpecifiedTSID):	0100501.DVVR.DivTotal.Month
	0100503.DVVR.DivTotal.Month
Handle missing data how?:	IgnoreMissing
	Subtract(TSID="0100501.DWR.DivTotal.Month",SubtractTSLis
Command:	t=SpecifiedTSID,SubtractTSID="0100503.DWR.DivTotal.Month
	",HandleMissingHow="IgnoreMissing")
	Cancel OK
-	

Subtract() Command Editor

The command syntax is as follows:

Subtract(Parameter=Value,...)

Parameter	Description	Default
TSID	The time series identifier or alias for the time series to	TSID or EnsembleID
	receive the result.	must be specified.
EnsembleID	The ensemble to receive the result, if processing an	TSID or EnsembleID
	ensemble.	must be specified.
Subtract	Indicates how the list of time series is specified, one of:	AllTS (the time series
TSList	• AllTS – all time series before the command.	receiving the sum will
	• AllMatchingTSID – all time series that match the	not be subtracted from
	AddTSID (single TSID or TSID with wildcards) will	itself)
	be subtracted.	
	• EnsembleID – the time series from ensemble will be	
	subtracted.	
	• LastMatchingTSID – the last time series that	
	matches the TSID (single TSID or TSID with	
	wildcards) will be subtracted.	
	• Selected TS – the time series are those selected with	
	the selectTimeSeries() command.	
	• Specified ISID – the specified list of time series	
	version 8 02 00 or earlier use Space field TS	
SubtractTSID	If the Subtract TSL ist parameter is	Must be specified if
Saberacerbib	Specified TSID provide the list of time series	TSList=
	identifiers (or alias) to subtract, separated by commas. If	SpecifiedTSID,
	the SubtractTSList parameter is	ignored otherwise.
	AllMatchingTSID, specify a single TSID or a TSID	
	with wildcards.	
Subtract	If the EnsembleID parameter is specified, providing an	Use if an ensemble is
EnsembleID	ensemble ID will subtract the ensembles.	being subtracted from
		another ensemble.
Handle	Indicates how to handle missing data in a time series, one	IgnoreMissing
MissingHow	of:	
	• IgnoreMissing – create a result even if missing	
	data are encountered in one or more time series – this	
	option is not as rigorous as the others	
	• SetMissingIfOtherMissing – set the result	
	missing if any of the other time series values is missing	
	• SetMissingIfAnyMissing – set the result	
	missing if any time series value involved is missing	

Command Parameters

A sample command file to subtract data from the State of Colorado's HydroBase is as follows:

```
# 0100501 - EMPIRE DITCH
0100501.DWR.DivTotal.Month~HydroBase
# 0100503 - RIVERSIDE CANAL
0100503.DWR.DivTotal.Month~HydroBase
Subtract(TSID="0100501.DWR.DivTotal.Month",SubtractTSList=SpecifiedTSID,
SubtractTSID="0100503.DWR.DivTotal.Month",
HandleMissingHow="IgnoreMissing")
```

Command Reference: TableMath()

Perform simple math operation on columns in a table

Version 09.08.01, 2010-09-14

The TableMath() command performs a simple math operation on columns in a table. Although the design of the command could support more advanced cell range addressing schemes, it currently processes complete columns of data. For example, a table that is populated by the CalculateTimeSeriesStatistic() command could be manipulated to produce a new column of data. This command and related table commands are not an attempt to replace full-feature spreadsheet programs but are intended to help automate common data processing tasks.

The input is specified by a table column name (Input1) and either a second input column name or a constant value (Input2), with the result being placed in the output column (Output). Output that cannot be computed is set to the NonValue value.

The following dialog is used to edit the command and illustrates the syntax of the command (in this case illustrating how values in a column named tsl are multiplied by the number 2.

🌌 Edit Table	eMath() Command	\mathbf{X}		
Perform simple n	nath operation on columns of	data in a table, using one of the following approaches:		
- process input from two columns to populate the output column				
- process input from a column and a constant to populate the output column				
Future enhance	ments may provide more cell r	ange addressing - currently full columns are processed.		
Table ID:	Table1 🗸 🗸	Required - table to process.		
Input 1:	tsi 🗸	Required - first input column name.		
Math operator:	*	Required - math calculation to perform on input.		
Input 2:	2	Required - second input column name, or constant.		
Output column:	result 💌	Required - output column name.		
Non-value:	×	Optional - non-value for missing, unable to compute (default=Null).		
	TableMath(TableID	="Table1", Input1=ts1, Operator="*", Input2		
Command:	=2,Output=result)			
		Cancel OK		
		Т		

TableMath() Command Editor

The command syntax is as follows:

```
TableMath(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
TableID	The identifier for the table to process.	None – must be
		specified.
Input1	First input column name.	None – must be
		specified.
Operator	The operator to be applied as follows:	None – must be
	Input1 Operator Input2 = Output	specified.
	For example:	
	Input1 * Input2 = Output	
Input2	Second input column name, or a constant value to use as	None – must be
	input.	specified.
Output	Output column name. If the column is not found it will be	None – must be
	added to the table and will contain the results of processing.	specified.
NonValue	The value to use in cases where an output result could not	Null
	be computed (missing input, division by zero). Null will	
	result in blanks in output whereas NaN may be shown in	
	some output products, depending on the specifications for	
	the format.	

Command Reference: TableTimeSeriesMath()

Perform simple math operation on time series using table input

The TableTimeSeriesMath() command performs a simple math operation on time series using values from a table. For example, a table that is populated by the CalculateTimeSeriesStatistic() command or ReadTableFromDelimitedFile() could be used to modify time series data. See also the TableMath() command, which performs math on a table.

The table value is determined by matching the time series identifier (formatted according to the TableTSIDFormat parameter) with the TSID value in the table column specified by the TableTSIDColumn parameter. If necessary, use the ManipulateTableString() command to generate an identifier column in the table that allows that match. Missing values in the time series generally will not be updated, although the assignment (=) operator will do so.

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

🜢 Edit TableTimeSeriesMath() Command 🛛 🛛 🔀							
Perform a simple math operation on time series using matching input from a table.							
For example, multiply values in a time:	For example, multiply values in a time series by a value from a table.						
The table value is determined from a r	ow with a matching time series identifier (TS	5ID) and by specifying the column from which to get a value.					
Missing values in the time series gener	ally cannot be modified, other than by the	assignment (=) operator.					
TS list:	AllMatchingTSID 💙	Optional - indicates the time series to process (default=AlITS).					
TSID (for TSList=AllMatchingTSID):	tsi	×					
EnsembleID (for TSList=EnsembleID):		×					
Math operator:	=	Required - math calculation to perform on input.					
Table ID:	Table1	💌 Required - table to process.					
Table TSID column:	TSID	Required - column name for TSID.					
Format of TSID:	%L Insert: Select Specific	r 💌 Optional - use %L for location, etc. (default=alias or TSID).					
Table input column:	DataValue	Required - column name for table input.					
If table input is blank:	✓	Optional - action if table input value is blank (default=Warn).					
If time series list is empty:	✓	Optional - action if time series list is empty (default=Fail).					
Command:	TableTimeSeriesMath(TSLis e1",TableTSIDColumn="TSID ataValue")	t=AllMatchingTSID,TSID="ts1",TableID="Tabl ",TableTSIDFormat="%L",TableInputColumn="D					
	Cancel	OK TablaTimeSariadMat					

TableTimeSeriesMath() Command Editor

The command syntax is as follows:

```
TableTimeSeriesMath(Parameter=Value,...)
```

Parameter	Description	Default
TSList	Indicates the list of time series to be	Allts
	processed, one of:	
	• AllMatchingTSID – all time series	
	that match the TSID (single TSID or	
	TSID with wildcards).	
	• AllTS – all time series before the	
	command.	
	• EnsembleID – all time series in the	
	ensemble.	
	 FirstMatchingTSID – the first time 	
	series that matches the TSID (single	
	TSID or TSID with wildcards).	
	 LastMatchingTSID – the last time 	
	series that matches the TSID (single	
	TSID or TSID with wildcards).	
	• SelectedTS – the time series selected	
	with the SelectTimeSeries()	
	command.	
TSID	The time series identifier or alias for the time	Required if
	series to be processed, using the * wildcard	TSList=*TSID.
	character to match multiple time series.	
EnsembleID	The ensemble to be processed, if processing	Required if
	an ensemble.	TSList=EnsembleID.
Operator	The operator to be applied to the time series	None – must be specified.
	and table input.	
TableID	Identifier for table that provides input.	None – must be specified.
TableTSIDColumn	Table column name that is used to match the	None – must be specified.
	time series identifier for processing.	
TableTSIDFormat	The specification to format the time series	Time series alias if
	identifier to match the ISID column. Use the	available, or otherwise
	unique identifier	the time series identifier.
TableInput Column	Table column name to retrieve the table value	None – must be specified
If Table Input IsBlank	Action if time table input is blank during	Warn
	processing (no value to operate on)	MALTI
IfTSListIsEmpty	Action if time series list is empty.	Fail

Command Parameters

The delimited file corresponding to that used in the above dialog example is shown below. In this example, the time series identifiers have location parts with values tsl and ts2.

Simple test data
"TSID","DataValue"
ts1,2
ts2,3

Command Reference: TableToTimeSeries()

Create time series from a table

Version 10.21.00, 2013-06-27

Note: This command may be split into two separate commands (one for single column data values and one for multiple column data values) if editing the command parameters becomes confusing.

The TableToTimeSeries() command creates time series from a table. This command can be used when a command to read time series from a specific file format or datastore has not been implemented. The table typically is read using one of the following commands:

- ReadTableFromDataStore() for example, define an ODBC DSN connection to a database and query time series using an SQL statement.
- ReadTableFromDelimitedFile() for example, read time series from a commaseparated-value (CSV) file.
- ReadTableFromExcel() for example, read time series from a comma-separated-value (CSV) file
- ReadTableFromHTML() envisioned for the future.

TSTool internally represents tables as a collection of columns, where a column contains values of a consistent data type (e.g., integer, string, double). A time series table requires at a minimum a date/time column (or separate date and time columns), at least one data value column, and optionally one or more columns for data flags. Data represented in one of two table designs are handled by this command:

- Data for multiple locations/series stored in a single column (common in a database or stream of data from a data logger) specify the LocationColumn command parameter.
- Data for multiple locations/series stored in multiple columns (common in spreadsheets and CSV files) do not specify the LocationColumn command parameter but instead specify the ValueColumn and optionally LocationID parameters.

The command provides flexibility to specify time series metadata (e.g., data source, units) as command parameters, or read from the file. However, this flexibility is limited by practical considerations in supporting likely data formats. One current limitation of the command is that TSTool does not determine table column names during discovery mode (discover mode is a partial command run that allows data such as time series and table identifiers to be provided to later commands for editing). Consequently, although this command will create time series when run, it does not produce time series information in discovery mode and the time series will not be listed in later command editors. This limitation will be addressed in future TSTool updates.

An example of a table with single data value column with flags is shown in the following figure (note that a column is used for the location identifier and that the location is different for the topmost and bottommost records).

Date	Location	Value	Flag	
2000-12-26	06754000	515.00	-	*
2000-12-27	06754000	529.00		
2000-12-28	06754000	552.00	-	
2000-12-29	06754000	582.00		
2000-12-30	06754000	595.00		
2000-12-31	06754000	578.00		1
2000-01-01	06758500	899.00		-
2000-01-02	06758500	988.00		
2000-01-03	06758500	1000.00		
2000-01-04	06758500	981.00		
2000-01-05	06758500	994.00		
2000-01-06	06758500	999.00		
2000-01-07	06758500	998.00		
Displaying 732	rows, 4 columns.	1 1441.44	1	Ready

TableToTimeSeries_Single_Data

Simple Table with Data Values in a Single Column

In the above example, the list of unique time series is determined by examining the location column contents. Other time series metadata such as data source and units can be assigned using the DataSource, Units, and similar parameters. The following dialog is used to edit the command and illustrates the command syntax when processing single-column data from the above example. Note that time series metadata are specified with command parameters.

Sedit TableToTimeSeries Command			
Create 1+ time series from a table. The table can contain one column per time series, or a singl The column name(s), date/time column, value column(s), and Location ID(s) columns can use the For example, "Date, TC[2:]" defines the first column as "Date" and column names 2+ will be take If used, specify input start and end to a precision appropriate for the data.	e column for all time series. e notation TC[start:stop] to use column names. n from the table.		
Table ID: SingleColumnData	Prequired - table to process.		
Date/time column: Date	Date Required - if date and time are in the same column (can use "TC[N]").		
Date/time format:	Optional - date/time format MM/DD/YYYY, etc. (default=auto-detect).		
Date column:	Required - if date and time are in separate columns (can use "TC[N:N]").		
Time column:	Required - if date and time are in separate columns (can use "TC[N:N]").		
Indicate how to assign location identifier			
Multiple Data Value (Number) Columns Single Data Value (Number) Column			
Location type column:	Optional - column name for location type		
Location column: Location	Required - column name for location identifier.		
Data source column:	Optional - column name for data source if not provided with DataSource.		
Data type column:	Optional - column name for data type if not provided with DataType.		
Scenario column:	Optional - column name for scenario if not provided as Scenario.		
Data units column:	Optional - column name for units, if not provided as Units.		
Value column(s): Value	Required - specify column names for time series values, separated by commas (can use "TC[N:N]").		
Flag column(s):	Optional - specify column names for time series flags, separated by commas (can use "TC[N:N]").		
Data source: USGS	USGS Optional - data source (provider) for the data (default=blank).		
Data type(s): Streamflow	Streamflow Optional - data type for each value column, separated by commas (default=value column name(s))		
Data interval: Day	Day Required - data interval for time series.		
Scenario:	Optional - scenario for the time series (comma-separated, default=blank).		
Units of data: cfs	cfs Optional - separate by commas (default=blank).		
Missing value(s):	Optional - missing value indicator(s) for table data (default=blank values).		
Alias to assign: Select Specifier 💙 => %L-%T	Optional - use %L for location, etc. (default=no alias).		
Input start:	Optional - overrides the global input start.		
Input end:	Optional - overrides the global input end.		
TableToTimeSeries(TableID="SingleColumnData",I "Value",DataSource="USGS",DataType="Streamflow Command:)ateTimeColumn="Date",LocationColumn="Location",ValueColumn= ",Interval=Day,Units="cfs",&lias="%L-%T")		
Cance	И ОК		
	TableToTimeSeries Single		

TableToTimeSeries() Command Editor for Table with Data in a Single Column

The following example is also treated as single-column because a single column of data values is present. However, metadata are taken from other columns. This data format is consistent with a database query where several tables have been joined together. Although not efficient because time series metadata is repeated for every row, the format is convenient for data translation. Use the DataSourceColumn, UnitsColumn and similar parameters to specify metadata. The unique list of time series will be determined from the combinations of location identifier and other metadata..

🔷 TSTool - Table "SingleColumnData"								
Date	Location	Value	LocationType	DataSou	DataType	Scenario	Units	
2000-12-25	06754000	520.00	StreamGage	USGS	Streamflow	Irrig	cfs	
2000-12-26	06754000	515.00	StreamGage	USGS	Streamflow	Irrig	cfs	1 -
2000-12-27	06754000	529.00	StreamGage	USGS	Streamflow	Irrig	cfs	1
2000-12-28	06754000	552.00	StreamGage	USGS	Streamflow	Irrig	cfs	1
2000-12-29	06754000	582.00	StreamGage	USGS	Streamflow	Irrig	cfs	1
2000-12-30	06754000	595.00	StreamGage	USGS	Streamflow	Irrig	cfs	1
2000-12-31	06754000	578.00	StreamGage	USGS	Streamflow	Irrig	cfs	1 🖃
2000-01-01	06758500	899.00	StreamGage	USGS	Streamflow	Irrig	cfs	1
2000-01-02	06758500	988.00	StreamGage	USGS	Streamflow	Irrig	cfs	1
2000-01-03	06758500	1000.00	StreamGage	USGS	Streamflow	Irrig	cfs	1
2000-01-04	06758500	981.00	StreamGage	USGS	Streamflow	Irrig	cfs	1
2000-01-05	06758500	994.00	StreamGage	USGS	Streamflow	Irrig	cfs	
2000-01-06	06758500	999.00	StreamGade	USGS	Streamflow	Irria	cfs	
Displaying 732 rov	Displaying 732 rows, 8 columns. Ready							
TableToTimeSeries_SingleMeta_Data								

Table with Data Values in a Single Column and Metadata Provided in Other Columns

The following dialog is used to edit the command and illustrates the command syntax when processing single-column data from the above example. Note that time series metadata are specified with command parameters.

S Edit TableToTimeSeries Command						
Create 1+ time series from a table. The table can contain one column per time series, or a sing	e column for all time series.					
The column name(s), date/time column, value column(s), and Location ID(s) columns can use the	e column name(s), date/time column, value column(s), and Location ID(s) columns can use the notation TC[start:stop] to use column names.					
For example, "Date, TC[2:]" defines the first column as "Date" and column names 2+ will be take If used, specify input start and end to a precision appropriate for the data.	"Date, TC[2:]" defines the first column as "Date" and column names 2+ will be taken from the table.					
Table ID: SingleColumnData	Required - table to process.					
Date/time column: Date	Required - if date and time are in the same column (can use "TC[N]").					
Date/time format:	Optional - date/time format MM/DD/YYYY, etc. (default=auto-detect).					
Date column:	Required - if date and time are in separate columns (can use "TC[N:N]").					
Time column:	Required - if date and time are in separate columns (can use "TC[N:N]").					
Indicate how to assign location identifier						
Multiple Data Value (Number) Columns Single Data Value (Number) Column						
Location type column:	Optional - column name for location type					
Location column: Location	Required - column name for location identifier.					
Data source column: DataSource	Optional - column name for data source if not provided with DataSource.					
Data type column: DataType	Optional - column name for data type if not provided with DataType.					
Scenario column: Scenario	Optional - column name for scenario if not provided as Scenario.					
Data units column: Units	Optional - column name for units, if not provided as Units.					
Value column(s): Value	Required - specify column names for time series values, separated by commas (can use "TC[N:N]").					
Flag column(s):	Optional - specify column names for time series flags, separated by commas (can use "TC[N:N]").					
Data source:	Optional - data source (provider) for the data (default=blank).					
Data type(s):	Optional - data type for each value column, separated by commas (default=value column name(s)).					
Data interval: Day	Day Required - data interval for time series.					
Scenario:	Optional - scenario for the time series (comma-separated, default=blank).					
Units of data:	Optional - separate by commas (default=blank).					
Missing value(s):	Optional - missing value indicator(s) for table data (default=blank values).					
Alias to assign: Select Specifier 💉 => %L-%T	Optional - use %L for location, etc. (default=no alias).					
Input start:	Optional - overrides the global input start.					
Input end:	Optional - overrides the global input end.					
TableToTimeSeries(TableID="SingleColumnData", J	DateTimeColumn="Date",LocationColumn="Location",DataSourceCo					
lumn="DataSource",DataTypeColumn="DataType",So	cenarioColumn="Scenario",UnitsColumn="Units",ValueColumn="Va					
lue", Interval=Day, Alias="%L-%T")	lue", Interval=Day, Alias="%L-%T")					
Command:						
Cance	я ок					

TableToTimeSeries() Command Editor for Table with Single Data Column and Metadata Columns

An example of multi-column data with flags is shown in the following figure, where each time series has its own data and flag columns:

🕪 TSTool - Table "MultiColumnData"					
Date	06754000	06754000-flag	06758500	06758500-flag	
2000-03-28	868.00	d	755.00	1	2
2000-03-29	655.00	d	705.00		
2000-03-30	599.00		561.00	d	_
2000-03-31	541.00		522.00	1.2	
2000-04-01	947.00	D	481.00		
2000-04-02	1220.00	D	740.00	D	
2000-04-03	1110.00	d	1060.00	D	
2000-04-04	1230.00	D	1020.00	1	
2000-04-05	943.00	d	1110.00		M
Displaying 366 r	ows, 5 columns.				Ready



TableToTimeSeries_Multiple_Data

The following dialog is used to edit the command and illustrates the syntax for the command when processing multi-column data from the above table.

Create 1+ time series from a table. The table can contain one column per time series, or a single column for all time series. The column name(s), date/time column, value column(s), and Location ID(s) columns can use the notation TC[start:stop] to use column names.	
For example, "Date,TC[2:]" defines the first column as "Date" and column names 2+ will be taken from the table. If used, specify input start and end to a precision appropriate for the data.	
Table ID: MultiColumnData	
Date/time column: Date	
Date/time format:	
Date column: Required - if date and time are in separate columns (can use "TC[N:N]").	
Time column: Required - if date and time are in separate columns (can use "TC[N:N]").	
Indicate how to assign location identifier	
Multiple Data Value (Number) Columns Single Data Value (Number) Column	
	ans
	17
Value column(s): 06754000,06758500 Required - specify column names for time series values, separated by commas (can use "TC[:N]").
Flag column(s): 06754000-flag,06758500-flag Optional - specify column names for time series flags, separated by commas (can use "TC[N:]").
Data source: USG5 Optional - data source (provider) for the data (default=blank).	
Data type(s): Streamflow Optional - data type for each value column, separated by commas (default=value column na	.e(s)).
Data interval: Day 💌 Required - data interval for time series.	
Scenario: Optional - scenario for the time series (comma-separated, default=blank).	
Units of data: cfs Optional - separate by commas (default=blank).	
Missing value(s): Optional - missing value indicator(s) for table data (default=blank values).	
Alias to assign: Select Specifier 💙 => %L-%T Optional - use %L for location, etc. (default=no alias).	
Input start: Optional - overrides the global input start.	
Input end: Optional - overrides the global input end.	
TableToTimeSeries(TableID="MultiColumnData", DateTimeColumn="Date", LocationID="06754000,06758500", Value(51
umn="06754000,06758500",FlagColumn="06754000-flag,06758500-flag",DataSource="USGS",DataType="Streamflow	",
Interval=Day, Units="cfs", Alias="%L-%T")	
Contrario:	
Cancel OK	
	- ا منا ا

TableToTimeSeries() Command Editor For Table with Data in a Single Column

The command syntax is as follows:

TableToTimeSeries(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TableID	The identifier for the table to read.	None – must be
		specified.
DateTime	The column for date/time, when date and time are in one	Required if
Column	column. If the table was read in a way that the column	DateColumn is
	type is "date/time", then the values are used directly. If	not specified.
	the table was read in a way that the column type is	•
	"string", then the string is parsed using default logic or the	

Parameter	Description	Default
	DateTimeFormat parameter if specified.	
DateTime	The format for date/time strings in the date/time column, if	Will automatically
Format	strings are being parsed. If blank, common formats such	be determined by
	as YYYY-MM-DD hh:mm and MM/DD/YYYY will	examining date/time
	automatically be detected. However, it may be necessary	strings.
	to specify the format to ensure proper parsing. This	
	format will be used to parse date/times from the	
	DateTimeColumn or the merged string from the	
	DateColumn and TimeColumn (if specified). The	
	format string will depend on the formatter type. Currently,	
	only the "C" formatter is available, which uses C	
	programming language specifiers. The resulting format	
	includes the formatter and specifiers (e.g., C: %m%d%y).	D
DateColumn	The name of column that includes the date, used when	Required if
	date and time are in separate columns.	DateTimeColumn
	The many of externa that is dealer the time and entern	1s not specified.
TimeColumn	I he name of column that includes the time, used when data and time are in concrete columns. If both	Required if
	Dete de lume and Time de lume are specified their	DateColumn 1s
	contents are merged with a joining colon character and are	specified and the
	then treated as if Dat of imore lump had been specified	time
LocationID	Used with multiple data column table. The location	None must be
Docacioniid	identifier(s) to assign to time series separated by columns	specified for
	if more than one column is read from the table. Column	multiple column
	names can be specified as literal strings or as	data tables
	TC[start:stop] to match table column names, where	
	start is $1 + $ and stop is blank to read all columns or a	
	negative number to indicate the offset from the end	
	column.	
LocationType	Used with single data column table. The name of the	Do not assign a
Column	column containing the location type.	location type.
LocationColumn	Used with single data column table. The name of the	None – must be
	column containing the location identifier.	specified for single
		column data tables.
DataSource	Used with single data column table. The name of the	Use the
Column	column containing the data source.	DataSource
		parameter, which
		can be blank.
DataType	Used with single data column table. The name of the	Use the DataType
Column	column containing the data type.	parameter, which
		can be blank.
ScenarioColumn	Used with single data column table. The name of the	Use the Scenario
	corumn containing the scenario.	parameter, which
UnitaColumn	Used with single data solutor table. The same of the	Lies the II.
UIIIUSCOIUMII	Used with single data column table. The name of the	Use the Units
		parameter, which
ValueColumn	The name(s) of column(s) containing data values	None must be
varaccorani	The name(s) of containing uata values.	I NORC - HUSE UC

Parameter	Description	Default
	Separate column names with commas. The TC[start:stop] notation discussed for LocationID can be used. Only one column should be specified for single data column table	specified.
FlagColumn	The name(s) of column(s) containing the data flag. Separate column names with commas. The TC[start:stop] notation discussed for LocationID can be used. If specified, the number of columns must match the ValueColumn parameter, although specifying blank column names is allowed to indicate that a value column does not have a corresponding flag column.	Flags are not read.
DataSource	The data source (provider) identifier to assign to time series for each of the value columns (or specify one value to apply to all columns).	No provider will be assigned.
DataType	The data type to assign to time series for each of the value columns (or specify one value to apply to all columns).	Use the value column names for the data types.
Interval	The interval for the time series. Only one interval is recognized for all the time series in the table. Interval choices are provided when editing the command. If it is possible that the date/times are not evenly spaced, then use the Irregular interval (this is difficult to do for multiple data column tables).	None – must be specified.
Scenario	The scenario to assign to time series for each of the value columns (or specify one value to apply to all columns).	No scenario will be assigned.
Units	The data units to assign to time series for each of the value columns (or specify one value to apply to all columns).	No units will be assigned.
Missing	Strings that indicate missing data in the table (e.g., "m"), separated by commas.	Interpret empty column values as missing data.
Alias	The alias to assign to time series, as a literal string or using the special formatting characters listed by the command editor. The alias is a short identifier used by other commands to locate time series for processing.	No alias will be assigned.
InputStart	The date/time to start reading data.	All data or global input start.
InputEnd	The date/time to end reading data.	All data or global input end.

Command Reference: TimeSeriesToTable()

Copy one or more time series into a table

Version 10.21.00, 2013-06-26

The TimeSeriesToTable() command copies one or more time series into a table. This command is useful when performing table analysis processing and outputting table formats (e.g., with the WriteTableToDelimitedFile() or WriteTableToHTML() commands). The command can be configured to output one of two table forms:

- Each time series in a separate column, with shared date/time column:
 - The time series must be regular interval (no irregular interval time series) and the intervals must match in order to allow alignment of the date/times.
 - o Do not specify the TableTSIDColumn or TableTSIDFormat parameters.
- All time series values in a single column (useful for converting time series to a stream of data for loading into a database)
 - Any interval is allowed although mixing time series of varying precision is discouraged.
 - o Specify the TableTSIDColumn and optionally TableTSIDFormat parameters.

Currently the command can only be used to create a new table but in the future the command is envisioned to write into an existing table. The following dialog is used to edit the command and illustrates the syntax of the command when writing a multi-column data table while also outputting data flags. Note that the value columns can be specified using time series properties.

👌 Edit TimeSeriesToTable() C	Command			
Copy time series date/time and value p	pairs to column(s) in a new table.If the table TS	51D column is specified, output will be to a single col	umn.	
The time series must have the same da	ata interval if a multi-column table is created.			
If the output window is specified, use	a date/time precision consistent with data.			
TS list:	×		Optional - indicates the time :	series to process (default=AllTS).
TSID (for TSList=AllMatchingTSID):				~
EnsembleID (for TSList=EnsembleID):				~
Table ID:	TestTable 💌		Required - table identifier.	
Date/time column in table:	Year		Required - column name for c	late/times.
Single-column output parameters				
Table TSID column:		Optional - column name for TSID (if values in	single column).	
Format of TSID: Select Sp	pecifier 💌 =>	Optional - can use if TableTSIDColumn is spe	cified (default=alias or TSID).	
Include missing values?:		Optional - include missing values (default=Tr	ue).	
Data value column(s) in table:	Select Specifier 🔽 => %L-%T		Required - value column nam	e(s) for 1+ time series.
Flag column(s) in table:	Select Specifier 💌 => %L-%T-flag		Optional - flag column name(:	;) for 1+ time series.
First row for data:	1		Required - row number (1+)	for first data value.
Output start date/time:			Optional (default=copy all).	
Output end date/time:			Optional (default=copy all).	
	Start	T LEU		
Output window:	Month: Day: Hour: Minute:	Month: Day: Hour: Minute:	Optional - output window wit	hin each year (default=full year).
Action if table not found:	Create 💌		Required.	
	TimeSeriesToTable(TableID="T	estTable",DateTimeColumn="Year"	,ValueColumn="%L-%	T",FlagColumn="%L-
Command:	%T-flag",DataRow=1,IfTableNo	tFound="Create")		
Lonnardi				
	L			
		Cancel OK		
				TimeCaricaTaTabl

TimeSeriesToTable() Command Editor to Create Multi-Column Data Table

The command syntax is as follows:

TimeSeriesToTable(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed,	Allts
	one of:	
	• AllMatchingTSID – all time series that	
	match the TSID (single TSID or TSID with	
	wildcards).	
	• AllTS – all time series before the	
	command.	
	• EnsembleID – all time series in the	
	ensemble.	
	• FirstMatchingTSID - the first time	
	series that matches the TSID (single TSID	
	or TSID with wildcards)	
	• LastMatchingTSID - the last time	
	series that matches the TGID (single TSID	
	or TSID with wildcards)	
	Coloct od TC the time series are those	
	• Selected 15 - the time series are those selected with the Collect Time Coming ()	
	selected with the SelectlimeSelles()	
TOTO	The time series identifier or alies for the time	Paguirad whan
1510	series to be modified, using the * wildcard	TSLigt - *TSTD
	character to match multiple time series	191186- 1910
FncembleID	The ensemble to be modified, if processing an	Pequired when
EIISERDIEID	ensemble	TSList=EnsembleID
TableID	The identifier for the table to copy data into (or	None – must be specified
1001010	the identifier for the new table to create if	i tone indist de speemed.
	If TableNotFound=Create)	
DateTimeColumn	The table column name to receive date/time	None – must be specified
	information.	i tone must be specifical
TableTSIDColumn	For single-column output, the name of the	Optional – if specified will
	column in the table for time series identifier	indicate single-column
	information. The format of the identifier can be	output.
	specified using the TableTSIDFormat	
	parameter.	
TableTSIDFormat	For single-column output, indicates how to	Optional – if not specified
	format the time series identifier that is inserted	the alias or full TSID will
	in the column specified by the	be used.
	TableTSIDColumn parameter.	
Include	For single-column output, indicates whether	True
MissingValues	missing values should be transferred to the table.	
	This is useful to screen out missing values from	
	sparse time series.	
ValueColumn	The data value column name(s) to receive time	None – must be specified.

Parameter	Description	Default
	series data, specified as follows:	
	• Multiple names separated by a comma.	
	• Time series property format specifiers,	
	available in a list of choices. These	
	specifiers are consistent with other	
	commands and the legend formatter in the	
	graphing tool.	
	• If a literal string is specified with multi-	
	column output, names for columns 2+ will	
	be generated by adding a sequential number	
	to ValueColumn.	
FlagColumn	The data flag column name(s) to receive time	Do not output flags to the
	series flags, specified using the same syntax as	table.
	ValueColumn. A blank in the list will result	
	in no transfer of flags for the specific time	
	series.	
DataRow1	First table row for data $(1+)$, where the row	None – must be specified.
	number is data only (column names are not	
	considered a data row).	
OutputStart	The starting date/time for the copy.	Available period.
OutputEnd	The ending date/time for the copy.	Available period.
OutputWindowStart	The calendar date/time for the output start	Output the full year.
	within each year. Specify using the format MM,	
	MM-DD, MM-DD hh, or MM-DD hh:mm,	
	consistent with the time series interval precision.	
	A year of 2000 will be used internally to parse	
	the date/time. Use this parameter to limit data	
	processing within the year, for example to	
	output only a single month or a season.	
OutputWindowEnd	Specify date/time for the output end within each	Output the full year.
	year. See OutputWindowStart for details.	
IfTableNotFound	Indicate action if the table identifier is not	Warn
	matched, one of:	
	• Create – create a new table	
	• Warn – warn that the table was not matched	

A sample command file is as follows (this command file is used to verify the command during testing):

```
# Test copying annual time series to a table, and also create the table
StartLog(LogFile="Results/Test_TimeSeriesToTable_Year_Create.TSTool.log")
RemoveFile(InputFile="Results/Test_TimeSeriesToTable_Year_Create_out.csv",
IfNotFound=Ignore)
NewPatternTimeSeries(Alias="ts1",NewTSID="ts1..Flow.Year",SetStart="1960",
SetEnd="2000",Units="ACFT",PatternValues="1,2,5,8,,20")
NewPatternTimeSeries(Alias="ts2",NewTSID="ts2..Flow.Year",SetStart="1950",
SetEnd="2005",Units="ACFT",PatternValues="2,4,10,16,,40")
TimeSeriesToTable(TableID=TestTable,DateTimeColumn=Year,ValueColumn=%L-%T,
FlagColumn="%L-%T-flag",DataRow=1,IfTableNotFound="Create")
# Generate the results.
WriteTableToDelimitedFile(TableID="TestTable",
```

TimeSeriesToTable2

The resulting table will be listed in the **Tables** area of the TSTool interface and clicking on the TestTable identifier will display the table similar to the following:

UTSTool	- Table "TestTable"				
Year	ts1-Flow	ts1-Flow-flag	ts2-Flow	ts2-Flow-flag	
1950		and the state	2.00	-	-
1951			4.00		
1952			10.00	1	
1953			16.00	ĺ	
1954				ĺ -	
1955			40.00	1	
1956			2.00	ĺ.	
1957			4.00	1	
1958			10.00	1	
1959		1	16.00	1	
1960	1.00	l .		ĺ -	
1961	.2,00	1	40.00	Ì	
1962	5.00	Ì	2.00	1	
1963	8.00	1	4.00	1	
1964		1	10.00	1	
1965	20.00	1	16.00	1	
1966	1.00	1		[· · · · · · · · · · · · · · · · · · ·	
1967	2.00	1	40.00	İ.	
1968	5.00	1	2.00	1	
1969	8.00	11-	4.00		11
1070			10.00	i Ener	×
Displaying 56	rows, 5 columns.			Rea	dy

Multi-Column Data Table

The following example illustrates how to create a single data column table. Because a single column is being used for data, the data value and corresponding data flag column names are specified literally (not as time series properties). The column and format for the TSID also must be specified.

👌 Edit TimeSeriesToTable()	Command	\mathbf{X}		
Copy time series date/time and value pairs to column(s) in a new table. If the table TSID column is specified, output will be to a single column.				
The time series must have the same data interval if a multi-column table is created.				
If the output window is specified, use	a date/time precision consistent with data.			
TS list:		Optional - indicates the time series to process (default=AIITS).		
TSID (for TSList=AllMatchingTSID):		×		
EnsembleID (for TSList=EnsembleID):		×		
Table ID:	TestTable	Required - table identifier.		
Date/time column in table:	: Year	Required - column name for date/times.		
Single-column output parameters—				
Table TSID column: TSID		Optional - column name for TSID (if values in single column).		
Format of TSID: Select S	ipecifier 💉 => %L-%T	Optional - can use if TableTSIDColumn is specified (default=alias or TSID).		
Include missing values?:		Optional - include missing values (default=True).		
Data value column(s) in table:	: Select Specifier 🕑 => Value	Required - value column name(s) for 1+ time series.		
Flag column(s) in table:	; Select Specifier 💉 => Flag	Optional - flag column name(s) for 1+ time series.		
First row for data:	1	Required - row number (1+) for first data value.		
Output start date/time:		Optional (default=copy all).		
Output end date/time:		Optional (default=copy all).		
V Output window:	Start- Month: Day: 5 • 1 • 1 • 7 • 31 •	Optional - output window within each year (default=full year).		
Action if table not found:	Create 💌	Required.		
Command:	TimeSeriesToTable(TableID="TestT "TSID",TableTSIDFormat="%L-%T",V =1,OutputWindowStart="05-01",Out ")	able",DateTimeColumn="Year",TableTSIDColumn= alueColumn="Value",FlagColumn="Flag",DataRow putWindowEnd="07-31",IfTableNotFound="Create		
Cancel OK				

TimeSeriesToTable() Command Editor to Create Single Data Column Table

🖉 TSTool - T	able "TestTable"			
Year	TSID	Value	Flag	3
2000-05-01	ts1-Flow	-2.00	b	
2000-05-02	ts1-Flow	3.00		
2000-05-03	ts1-Flow	-4.00	d	
2000-05-04	ts1-Flow	5.00	e	
2000-05-05	ts1-Flow			
2000-05-06	ts1-Flow	7.00		
2000-05-07	ts1-Flow	-8.00	h	
2000-05-08	ts1-Flow	9.00	Ľ –	
2000-05-09	ts1-Flow	-10.00	j –	
2000-05-10	ts1-Flow	11.00	L .	
2000-05-11	ts1-Flow	-12.00	m	
2000-05-12	ts1-Flow	1.00	a	
2000-05-13	ts1-Flow	-2.00	b	
2000-05-14	ts1-Flow	3.00		-
2000-05-15	ts1-Flow	-4.00	d	
2000-05-16	ts1-Flow	5.00	e	
2000-05-17	ts1-Flow		1	
2000-05-18	ts1-Flow	7.00		
2000-05-19	ts1-Flow	-8.00	h	
2000-05-20	ts1-Flow	9.00	ľ.	
2000-05-21	ts1-Flow	-10.00	i -	
	le altres	1 44.88	ĥ.	1 1
Displaying 664 ro	ws, 4 columns.			Ready

The resulting table is as shown in the following figure:

Single Data Column Table

TimeSeriesToTable_Single2

Command Reference: VariableLagK() Lag and attenuate (route) a time series with parameters that vary by rate

The VariableLagK() command can be used to lag and attenuate an input time series, resulting in a new time series. The command is commonly used to route an instantaneous flow time series through a stretch of river (reach). Lag and K routing is a common routing method that combines the concepts of:

- 1. Lagging the inflow to simulate travel time in a reach and,
- 2. Attenuating the wave to simulate the storage-outflow relationship for the reach (see **Figure 1**).



Figure 1: Lag and K Routing

At its fundamental level, the method solves the continuity equation using an approach similar to Muskingum routing (assuming that the Muskingum parameter representing wave storage is negligible). The governing equation for this routing method is given as:

$$Q_{in} - Q_{out} = \frac{\Delta S}{\Delta t}$$

where:

 Q_{in} = instantaneous inflow [rate] lagged appropriately, Q_{out} = instantaneous outflow [rate] lagged appropriately, ΔS = change in storage in the reach [volume], Δt = time difference. The relationship assumes an outflow-storage relationship of the form:

$$S = k \cdot Q_{out}$$

where:

k = attenuation for the outflow [time].

To ensure accurate results, k should be larger or equal to $\Delta t/2$. For discrete time steps these relationships translate into:

$$O_2 = \frac{I_1 + I_2 + \frac{2S_1}{\Delta t} - O_1}{\frac{2k}{\Delta t} + 1}, \qquad k \ge \frac{\Delta t}{2}$$

where: I_1 and I_2 are the lagged inflows into the reach at the previous and current time step, respectively,

 O_1 and O_2 are the outflows out of the reach at the previous and current time step, respectively, S_1 is the storage within the reach at the previous time step, defined as $S_1 = k \cdot O_1$, and Δt is the time difference between the two time steps.

Values for Lag and K can usually be established by comparing routed flows to downstream observations. Alternatively, the Lag can be estimated using the reach length and wave speed in the reach. Without any other information, K can be set to Lag/2.

The above discussion applies where the Lag and K parameters are single values (as implemented in the LagK() command). However, there are cases where the values vary by flow, which is handled by this command. The approach that is implemented is an adaptation of that described in National Weather Service River Forecast System LAG/K documentation:

http://www.nws.noaa.gov/oh/hrl/nwsrfs/users_manual/part2/_pdf/24lagk.pdf.

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

💧 Edit VariableLagK	gK() Command		
Lag and attenuate a time s	e series, creating a new time series using variable Lag and K technique.		
The time series to be route	uted cannot contain missing values.		
See the documentation for	for a complete description of the algorithm.		
The input and output s	state parameters are currently ignored - states default to zero.		
Time series to lag (TSID):): Inflow	*	
New time series ID:	P: TestLocSQIN.3Hour.routed Specify to avoid confusion with TSID from original TS	5.	
	Edit		
Flow units:	s: CMS Required - units of Lag and K flow values, compatibl	e with time series.	
Lag interval:	al: Hour 🗸 Required - Lag and K interval units.		
Lag:	g: 200, 24.0; 600, 12.0; 1500, 9.0; 3000, 42.0 Optional - flow,Lag;flow,Lag pairs.		
к:	K: 200,24.0;600,12.0;1500,9.0;3000,42.0 Optional - flow,K;flow,K pairs.		
Input time series states:	s: Optional - separate values by commas (default=0 fo	vrall).	
Output time series states:	s: Optional - separate values by commas (default=0 fo	or all).	
Alias to assign:	n: 3Hr Insert: Select Specifier 🔽 Optional - use %L for location, etc.		
	VariableLagK(TSID="Inflow" NewTSID="TestLog _SOIN 3Hour routed" FlowUnit	SECMS Lag	
Command:	Interval=Hour, Lag="200,24.0;600,12.0;1500,9.0;3000,42.0", K="200,24.0;600,12.0;150 0,9.0;3000,42.0", Alias="3Hr")		
Cancel OK			
		ViriableLad	

VariableLagK() Command Editor

The command syntax is as follows:

```
VariableLagK(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
TSID	Identifier or alias for the time series to be routed. It is assumed that this series describes an instantaneous flow. Due to the lagging, the first data values required for the computation of O_2 are not available within this time series and are therefore set to values set in the InflowStates parameter.	None – must be specified.
NewTSID	Identifier for the new (routed) time series. This is required to ensure that the internal identifier for the time series is unique and accurate for the data. The interval of the identifier must be the same as for the time series specified by TSID.	None – must be specified.

Parameter	Description	Default
FlowUnits	The units of the flow data specified in the Lag and K tables. These units must be compatible with the time series units. The table values will be converted to the time series units if the units are not the same.	None – must be specified.
LagInterval	The base interval for the time data specified in the Lag and K tables. The interval must be compatible with the time series base interval. The table values will be converted to the time series time interval if the intervals are not the same. For example, table data specified in Hour base interval will be converted to Minute if the time series being routed contains NMinute data.	None – must be specified.
Lag	Flow value and lag time pairs to control routing. The units of the data values are as specified by the FlowUnits parameter (see above). The units of the lag are time as specified by the LagInterval parameter. The Lag value is not required to be evenly divisible by the time step interval; values in the time series between time steps will be linearly interpolated. Use commas and semi-colons to separate values, for example: 100.0,10;200.0,20	None – must be specified.
K	Flow value and K time pairs to control routing. The attenuation factor K is applied to the wave. The units of K are time as specified by the LagInterval parameter. Use commas and semi-colons to separate values, for example: 100.0,5;200.0,10	None – must be specified.
InflowStates	Comma-delimited list of default inflow values prior to the start of the time series. The order of the values is earliest to latest. The array must specify (Lag/multiplier) + 1 values; i.e., a 10 minute interval with a LAG of 30 must be provided with $30/10$ + 1 = 4 inflow carryover values. Note: Specifying values that are not consistent with the Lag and K parameters will result in oscillation!	0 for each value CURRENTLY ALWAYS DEFAULT
OutflowStates	Comma-delimited list of default outflow values prior to the start of the time series. See InflowStates for details.	0 for each value CURRENTLY ALWAYS DEFAULT
Alias	The alias that will be assigned to the new time series.	No alias will be assigned.

A sample command file is as follows (commands to read time series are omitted):

```
# Test routing at 3 hour interval
StartLog(LogFile="Results/Test_VariableLagK_3hr.TSTool.log")
# Read NWSCard input file
ReadNwsCard(InputFile="Data\3HR_INPUT.SQIN",Alias="Inflow")
#
# Route using the same routing parameters used in the mcp3 input deck
# (metric units: Lag(hrs) K(hrs) Q(cms)
# Lag
# K
    24.0
           200.0
                   12.0 600.00
                                     9.0 1500.0
                                                    42.0 3000.0
#
           200.0 12.0 600.00
                                     9.0 1500.0
                                                  42.0 3000.0
    24.0
#
#
VariableLagK(TSID="Inflow",NewTSID="TestLoc..SQIN.3Hour.routed",DataUnits=CMS,
   LagInterval=Hour,Lag="200,24.0;600,12.0;1500,9.0;3000,42.0",
   K="200,24.0;600,12.0;1500,9.0;3000,42.0",Alias="3Hr")
```

This page is intentionally blank.

Command Reference: WebGet()

Retrieve a file from a website

Version 10.01.00, 2011-11-15

The WebGet() command retrieves content from a website and writes the content to a local file. The transfer occurs using binary characters and the local copy is the same as that shown by *View...Source* (or *View...Page Source*) in the web browser. This command is useful for mining time series data and other content from data provider web sites. The local file can then be processed with additional commands such as ReadFromDelimitedFile().

Extraneous content (such as HTML markup around text) and inconsistencies in newline characters (r n for windows and n on other systems) may lead to some issues in processing the content. Additional command capabilities may be implemented to help handle these issues.

The following dialog is used to edit the command and illustrates the syntax for the command. This example reads stream gage data from the State of Colorado's website.

\delta Edit WebGet() command 🛛 🛛 🔀				
This command retrieves content from the web using a Uniform Resource Identifier (URI) and saves the content to a local file. The URI and local file can be specified using \${Property} notation to utilize global properties. It is recommended that the local file name be relative to the working directory, which is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\WebGet				
URI:	http://www.dwr.state.co.us/SurfaceWater/data/export_tabular .aspx?ID=ADATUNCO&MTYPE=GAGE_HT,DISCHRG&INTERVAL=1&START=10 /1/06&END=10/6/06			
Local file:	Results\Test_WebGet_CoDwr_out.txt	Browse		
Command:	WebGet(URI="http://www.dwr.state.co.us/SurfaceWater/data/export_tab ular.aspx?IDADATUNCO&MTYPEGAGE_HT,DISCHRG&INTERVAL1&START10/1/06&EN D10/6/06",LocalFile="Results\Test_WebGet_CoDwr_out.txt")			
Add Working Directory to Local File Cancel OK				
		WebGe		

WebGet() Command Editor

The command syntax is as follows:

WebGet(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
URI	The Uniform Resource Identifier (URI)	None – must be specified.
	for the content to be retrieved. This is	
	often also referred to as the Uniform	
	Resource Locator (URL). Global	
	properties can be used with the	
	\${Property} syntax.	
LocalFile	The local file in which to save the	None – must be specified.
	content. Global properties can be used	
	with the \${Property} syntax.	

Command Reference: WeightTraces() Create a time series by weighting data from time series ensemble traces

Version 10.00.00, 2011-03-28

The WeightTraces() command creates a new time series as a weighted sum of time series ensemble traces, for example as produced by a CreateEnsemble() command. If any trace contains missing data for a point, the resulting time series value will also be missing. Note that this approach may not be appropriate for some analyses – the user should evaluate the implications of whether the weighted result appropriately reflects the (in)dependence of input data.

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

🗞 Edit WeightTraces() command 🛛 🔀					
Create a new time series t	Create a new time series by weighting traces from an ensemble. The result is identified by its alias.				
Enter trace years and wei	ghts (0.0 to 1.0), one val	ue per line.			
Any trace value that is mis	sing will cause the weight	ed result to be missing.			
Ensemble to process:	Ensemble_Jasper			*	
New time series ID parts:	08235350.USGS.S	treamflow.Day.weigh	Specify to a	avoid confusion with TSID from original TS.	
	ted		Edit	Clear	
Alias to assign:	WeightedTS	Insert: Select Specifier 🛛 👻	Required - (use %L for location, etc.	
Specify weights how?:	AbsoluteWeights 💌				
	1997 1998			.5 .4	
Trace years:	1999		Weights:	.1	
Command:	WeightTraces(Alias="WeightedTS",EnsembleID="Ensemble_Jasper",Speci fyWeightsHow="AbsoluteWeights",Weights="1997,.5,1998,.4,1999,.1")				
Cancel OK					

WeightTraces() Command Editor

WeightTraces

The command syntax is as follows:

WeightTraces(Parameter=Value,...)

The following older command syntax is updated to the above syntax when a command file is read:

TS Alias = WeightTraces(Parameter=Value,...)

Parameter Description Default EnsembleID The ensemble identifier indicating time None - must be specified. series to be processed (e.g., from a CreateEnemble() command). Time series matching the years specified by the Weights parameter will be processed. NewTSID The time series identifier for the new None – must be specified. time series that is created. This typically uses the same information as the original time series, with an added scenario. Alias The alias to assign to the time series, as a None – must be specified. literal string or using the special formatting characters listed by the command editor. The alias is a short identifier used by other commands to locate time series for processing, as an alternative to the time series identifier (TSID). SpecifyWeightsHow Weights are currently only applied as Must be AbsoluteWeights (in the future an AbsoluteWeights. option may be added to normalized weights to 1.0 accounting for missing data in the traces). Weights Specify pairs of trace year and weights None – must be specified. (0-1.0), used to create the new time series. Trace years must be manually entered because at the time that the command is edited, time series have not yet been queried. The weights do not need to add to 1. Example data are: 1995,.5,1998,.3,2005,.2

Command Parameters

A sample commands file is as follows (longer commands that wrap are shown indented):



UserManualExamples/TestCases/CommandReference/WeightTraces/WeightTraces.TSTool

The results from the commands are shown in the following graph:



Results of the WeightTraces() Command

This page is intentionally blank.
Command Reference: WriteCheckFile()

Write a check file containing a summary of data/processing problems

Version 09.03.04, 2009-04-23

The WriteCheckFile() command summarizes the results of command processing warning/failure messages in a "check file". This file is useful for reviewing results and for quality control. The check file is essentially a persistent record of any problems that occurred during processing, whereas a full log file contains a sequential list of processing.

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

Edit WriteCheckFile() Command
This command writes command warning/failure messages to a check file, as a summary of data/processing problems. Use Check*() commands prior to this command to perform checks on specific data object types. Specify an "html" extension for the output file to generate an HTML report, or "csv" to create a comma-separated value file The HTML file will contain navigable information whereas the CSV file will only contain a list of warning/failure messages.
Check (output) file: Results/Test_CheckTimeSeries_GreaterThan_out.html Browse
Command: WriteCheckFile(OutputFile="Results/Test_CheckTimeSeries_Great
Add Working Directory Cancel OK

WriteCheckFile() Command Editor

The command syntax is as follows:

WriteCheckFile(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
OutputFile	The name of the check 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.
	Specify a filename with <i>.html</i> extension to generate an HTML file or <i>.csv</i> to generate a comma-separated value file suitable for use with Excel. The HTML file will contain more information and include navigation links.	

This page is intentionally blank.

Command Reference: WriteDateValue()

Write time series to a DateValue format file

Version 10.06.00, 2012-04-05

The WriteDateValue() command writes time series to the specified DateValue format file. See the **DateValue Input Type Appendix** for more information about the file format. The time series being written must have the same data interval – use the TSList parameter to select appropriate time series.

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

\delta Edit WriteDateValue() Command 🛛 🛛 🔀		
Write time series to a DateValue forma The working directory is: C:\Develop\1	Write time series to a DateValue format file, which can be specified using a full or relative path (relative to the working directory). The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\\\\\/riteDateValue	
The output filename can be specified u	ising \${Property} notation to utilize	e global properties.
Enter date/times to a precision approp	riate for output time series.	
TS list:	×	Optional - indicates the time series to process (default=AllT5).
TSID (for TSList=AllMatchingTSID):		
EnsembleID (for TSList=EnsembleID):		
DateValue file to write:	Results/Diversions.dv	Browse
Delimiter:		Optional - delimiter between values (default=space, comma is only other allowed delimiter).
Output precision:		Optional - digits after decimal (default=4).
Missing value:		Optional - value to write for missing data (default=initial missing value).
Output start:		Optional - override the global output start (default=write all data).
Output end:		Optional - override the global output end (default=write all data).
Interval for irregular time series:	×	Required for irregular time series - used to process date/times.
	WriteDateValue(Outpu	tFile="Results/Diversions.dv")
Command:		
	Add Working D	Directory Cancel OK

WriteDateValue() Command Editor

The command syntax is as follows:

WriteDateValue(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one	AllTS
	of:	
	• AllMatchingTSID – all time series that	
	match the TSID (single TSID or TSID with	
	wildcards) will be processed.	
	• AllTS – all time series before the command.	
	• EnsembleID – all time series in the ensemble	
	will be processed.	
	• FirstMatchingTSID – the first time series	
	that matches the TSID (single TSID or TSID	

Parameter	Description	Default
	with wildcards) will be processed.	
	• LastMatchingTSID – the last time series that	
	matches the TSID (single TSID or TSID with	
	wildcards) will be processed.	
	• SelectedTS – the time series are those	
	selected with the SelectTimeSeries()	
	command.	
TSID	The time series identifier or alias for the time series	Required if
	to be processed, using the * wildcard character to	TSList=*TSID.
	match multiple time series.	
EnsembleID	The ensemble to be processed, if processing an	Required if TSList=
	ensemble.	EnsembleID.
OutputFile	The DateValue output file. The path to the file can	None – must be specified.
	be absolute or relative to the working directory	
	(command file location). Global properties can be	
	used to specify the filename, using the	
	\${Property} syntax.	~
Delimiter	The delimiter character to use between data values.	Space.
	Comma is the only other allowed value other than the	
	default space and is recommended for irregular time	
	don't align with other time series	
Precision	The number of digits after the decimal for numerical	4 (in the future may
11001010	output.	$\frac{1}{2}$ (in the future may default based on data type)
MissingValue	The value to write to the file to indicate a missing	As initialized when
moorngvarae	value in the time series.	reading the time series or
		creating a new time series,
		typically -999, NaN, or
		another value that is not
		expected in data.
OutputStart	The date/time for the start of the output.	Use the global output
		period.
OutputEnd	The date/time for the end of the output.	Use the global output
		period.
Irregular	The interval (e.g., Day) used when writing irregular	Determined from the
Interval	time series, to indicate the precision of date/times.	period start date/time of
	This may be necessary when it is not possible to	each time series, defaulting
	automatically determine the date/time precision. The	to Minute where the
	date/time precision to format output is assumed to be	date/time precision is set to
	Minute if unknown; nowever, specifying the	irregular (unknown).
	irregular interval will inform the data processing.	

A sample command file to process data from the State of Colorado's HydroBase database is as follows:

```
# 0100503 - RIVERSIDE CANAL
0100503.DWR.DivTotal.Month~HydroBase
WriteDateValue(OutputFile="Diversions.dv")
```

Command Reference: WriteHecDss()

Write time series to a HEC-DSS File

Version 09.03.00, 2009-04-10

The WriteHecDss() command writes time series to a HEC-DSS file. See the **HEC-DSS Input Type Appendix** for information about how time series properties are output to HEC-DSS files. Current limitations of the command are:

- Irregular time series are not supported the focus of initial development has been regular interval time series.
- 24-hour time series in TSTool cannot be written to HEC-DSS because HEC-DSS only supports 1DAY interval. Therefore, the time series must be converted to a daily time series before writing. An option to convert 24-hour values to 1DAY may be added to this command in the future.
- HEC-DSS uses times through 2400. TSTool will convert this to 0000 of the next day. Year, month, and day data are not impacted. The internal TSTool values will be converted to hour 2400 when writing. Therefore, reading from a HEC-DSS file and then writing should result in no change in data.
- Time series that are written overwrite existing time series, but only for the period that is written. Therefore, previously written values may remain, even if not appropriate. A future enhancement will allow the option of removing the old data before writing new data. The work-around is to write a period that is sufficiently long to guarantee that old data values do not remain in the file, or clear the file out with another tool such as DSSUTL before writing.
- Currently the connections to the HEC-DSS file will remain open after the write, in order to minimize performance degradation for multiple write commands. However, this will lock the HEC-DSS file so that other commands or programs cannot perform file manipulation, such as removing the file. The connections will automatically time out after several minutes. A future enhancement will ensure that the file connections can be closed.

The A-F parts of the HEC-DSS time series pathname by default are taken from the time series properties, as follows:

- The A and B parts are taken from the time series identifier location, where location should be defined as A:B.
- The C part is taken from the time series data type.
- The D part is taken from the time series period in memory or as defined by the output period.
- The E part is taken from the time series interval.
- The F part is taken from the time series identifier scenario.

These conventions can be overruled by specifying the parts explicitly with command parameters. The parameter values will apply to all time series being written.

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

Edit WriteHecDss() Command Write time series to a HEC-DSS format The working directory is: C:\Develop\T The Browse button can be used to sel Specifying a file that does not exist wi Enter date/times to a precision approp HEC-DSS parts will be taken from the t The defaults can be overridden with th	file, which can be specified using a 'STool_SourceBuild\TSTool\test\rea ect an existing file to update (or ec Il create a new file. riate for output time series. ime series identifier as follows: Ap ne optional part fields.	a full or relative path (relative to the working directory). gression\commands\general\WriteHecDss dit the file name after selection to specify a new file). art:Bpart.*.Cpart.Epart.Fpart	X
HEC-DSS file to write:	Results/Test_WriteHecDss_Day_	out.dss	Browse
Type:	PER-AVER	Required - HEC-DSS time series type.	
TS list:	-	Optional - indicates the time series to process (default=AlITS).	
TSID (for TSList=AllMatchingTSID):			~
EnsembleID (for TSList=EnsembleID):			-
Output start:		Optional - override the global output start (default=write all data).	
Output end:		Optional - override the global output end (default=write all data).	
Precision:		Optional - number of digits after decimal (default=HEC-DSS default).	
A:		Optional - A part to write (default=first : part of TSID location).	
В:		Optional - B part to write (default=second : part of TSID location).	
G		Optional - C part to write (default=TSID data type).	
E:		Optional - E part to write (default=TSID interval).	
F:		Optional - F part to write (default=TSID scenario).	
Replace entire time series?:	-	Optional - replace the entire time series (default=True). Under development.	
Close HEC-DSS files after?:	T	Optional - close HEC-DSS file after write (default=False).	
	WriteHecDss (OutputFi	le="Results/Test WriteHecDss Day out.dss",Type=PER	-AVER)
Command:			
command			
	Add Working	g Directory Cancel OK	
			WriteHecDss

WriteHecDss() Command Editor

The command syntax is as follows:

```
WriteHecDss(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
OutputFile	The name of the HEC-DSS file to write,	None – must be specified.
	surrounded by double quotes to protect	
	whitespace and special characters. If the file	
	does not exist it will be created.	
Туре	The HEC-DSS time series type, indicating	None – must be specified.
	whether the time series is instantaneous, mean,	
	or accumulated.	
TSList	Indicates the list of time series to be processed,	AllTS
	one of:	
	• AllMatchingTSID – all time series that	
	match the TSID (single TSID or TSID	
	with wildcards) will be processed.	

Parameter	Description	Default
	• AllTS – all time series before the	
	command will be processed.	
	• EnsembleID – all time series in the	
	ensemble will be processed.	
	• FirstMatchingTSID – the first time	
	series that matches the TSID (single TSID)	
	or TSID with wildcards) will be processed.	
	• LastMatchingTSID – the last time	
	series that matches the TSID (single TSID)	
	or TSID with wildcards) will be processed.	
	• SelectedTS – the time selected with the	
	SelectTimeSeries() command will	
	be processed.	
TSID	The time series identifier or alias for the time	Required if TSList=*TSID.
	series to be processed, using the * wildcard	-
	character to match multiple time series.	
EnsembleID	The ensemble to be processed, if processing an	Required if TSList=
	ensemble.	EnsembleID.
OutputStart	The date/time for the start of the output.	Use the global output period or
		write all available data.
OutputEnd	The date/time for the end of the output.	Use the global output period or
		write all available data.
Precision	The number of digits after the decimal for	HEC-DSS default.
	numerical output.	
А	The DSS path A-part to use for the time series	Time series identifier location
	as written to the HEC-DSS file.	part before the : (1f : 1s present)
		or the entire location if : is not
		present.
В	The DSS path B-part to use for the time series	Time series identifier location
	as written to the HEC-DSS file.	the block if the set present or
C	The DSS noth C part to use for the time series	Time series identifier data tune
C	as written to the HEC DSS file	Time series identifier data type.
म	The DSS path E-part to use for the time series	Time series identifier data
	as written to the HEC-DSS file	interval converted to HEC-
	as written to the file-bbs file.	DSS conventions
F	The DSS path F-part to use for the time series	Time series identifier scenario
	as written to the HEC-DSS file.	series racharier sechario.
Replace	Under development – whether to replace the	Only replace what is actually
	contents of the previous time series in the	written.
	HEC-DSS file.	
Close	Indicate whether to close connections to the	False – let the HEC-DSS
	HEC-DSS file and allow other processes to	internal software close the
	move/rename/delete the file. Specifying as	connection after timing out.
	True may slow the software as files are	
	repeatedly opened and closed.	

A sample command file is as follows:

```
WriteHecDss(OutputFile="sample.dss",TYPE=PER-AVER,
OutputStart="1992-01-01",OutputEnd="1992-12-31")
```

Command Reference: WritePropertiesToFile()

Write one or more time series processor properties to a file

Version 10.12.00, 2012-07-27

The WritePropertiesToFile() command writes the value of one or more time series processor properties to a file (this command replaces the older WriteProperty() command, which is being phased out). The ReadPropertiesFromFile() command can be used to read properties from a file. Processor properties include global defaults such as InputStart, InputEnd, OutputStart, OutputEnd, OutputYearType, WorkingDir, and also user-defined properties set with the SetProperty() command. Internally, properties have a name and a value, which is of a certain type (string, integer, date/time, etc.). Examples of using the command include:

- creating tests to verify that properties are being set
- passing information from TSTool to another program, such as a Python script
- storing persistent information for later use, such as the date/time that data were last downloaded from a web service

A number of property formats are supported as listed in the following table.

Format	Description
NameValue	Simple format, all properties handled as text:
	PropertyName=PropertyValue
	PropertyName="Property value, quoted if necessary"
NameTypeValue	Same as NameValue format, with non-primitive objects treated as simple
	constructors:
	PropertyName=PropertyValue
	DateTimeProperty=DateTime("2010-10-01 12:30")
NameTypeValue	Similar to the NameTypeValue format, however, objects are represented
Python	using "Pythonic" notation, to allow the file to be used directly by Python
	scripts:
	PropertyName="PropertyValue"
	DateTimeProperty=DateTime(2010,10,1,12,30)

Property File Formats

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

👌 Edit WritePro	pertiesToFile() Command	
Write one or more pro	perties to a file.	
It is recommended that	t the output file be relative to the	current working directory.
The working directory	is: C:\Develop\TSTool_SourceBuild	I\TSTool\test\regression\commands\general\WritePropertiesToFile
Property file to write:	LastRunPeriod	Browse
Property to write:	OutputStart,OutputEnd	Optional - properties to write, separated by commas (default=write all).
Write mode:	¥	Optional - how to write file (default=Overwrite).
File format:	×	Optional - property file format (default=NameTypeValue).
Command:	WritePropertiesToFil utputStart,OutputEnd	e(OutputFile="LastRunPeriod.txt",IncludeProperty="O ")
	Add Wor	rking Directory Cancel OK

WritePropertiesToFile() Command Editor

The command syntax is as follows:

WritePropertiesToFile(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
OutputFile	The property file to write, as an absolute path or relative to the command file.	None – must be specified.
IncludeProperty	The names of properties to write, separated by commas.	If not specified, all processor properties will be written.
WriteMode	 Indicates how the file should be written: Append – append the properties to the file without checking for matches (create the file if it does not exist) Overwrite – overwrite the properties file Update – update the properties in the file by first checking for matching property names (which will be updated) and then appending unmatched properties (not yet implemented) 	Overwrite
FileFormat	 Format of the properties file (see descriptions in the above Property File Formats table): NameValue NameTypeValue NameTypeValuePython 	NameValue

Command Reference: WriteProperty()

Write a time series processor property to a file

Version 10.12.00, 2012-07-30

This command has been replaced with the WritePropertiesToFile() command and will be removed from the software in a future release.

The WriteProperty() command writes the value of a time series processor property to a file. This is useful for testing that properties are being set. It could also be used to pass information from TSTool to another program. The format of the output is:

Property="Value"

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

Edit WriteProperty() Command
te a property to a file. This is useful for automated software testing.
recommended that the output file be relative to the current working directory.
working directory is: C:Develop\TSTool_SourceBuild\TSTooltestvegression\UserManualExamples\TestCases\CommandReference\WriteNwsrfsEspTraceEnsemble
perty file to write: Results/Example_WriteProperty.txt Browse
Property to write: WorkingDir Properties are maintained by the command processor.
Append to file?: Default=True
WriteProperty(OutputFile="Results/Example_WriteProperty.txt",PropertyName="WorkingDir")
Command
Command
Add Working Directory Cencel OK
WritePrope

WriteProperty() Command Editor

The command syntax is as follows:

WriteProperty(Parameter=Value,...)

Command Parameters

Parameter	Description	Default		
OutputFile	The property file to write, as an absolute path or relative to the command file.	None – must be specified.		
PropertyName	The property name to write	None – must be specified.		
Append	Indicates whether the property should be appended to the file (True) or overwrite the file (False).	True		

A sample command file is as follows:

```
WriteProperty(OutputFile="Results/Example_WriteProperty.txt",
PropertyName="WorkingDir")
```

Command Reference: WriteReclamationHDB()

Write a time series to a Reclamation HDB database

Version 10.20.00, 2013-04-21

The WriteReclamationHDB() command writes time series to a Reclamation HDB database:

- a single time series (which can be part of an ensemble), indicated by the individual time series identifier:
 - a "real" time series (observations)
 - a "model" time series (output from a model)
 - all time series in an ensemble, indicated by the ensemble identifier:
 - ensemble trace time series are stored as "model" time series (individual ensemble trace time series can then be read as single time series by specifying the appropriate "hydrologic indicator", which is set to the ensemble time series sequence number from TSTool time series)

See the ReadReclamationHDB() command documentation for information about reading the time series that are written by this command. See the **Reclamation HDB Data Store Appendix** for more information about the database features and limitations. Command functionality includes:

• Time series metadata/new time series:

- The command will not define a new time series (site and model data). It is expected that time series previously have been defined in the database. This ensures that TSTool can perform error handling and users do not accidentally load new time series.
- The exception is that new ensembles and corresponding trace time series can be defined by specifying ensemble name, trace number, and model run date.

• Date/time handling:

- TSTool uniformly uses the time at the end of the recorded interval for data values (instantaneous time or end of interval for mean and accumulated values), whereas HDB uses the time at the beginning of the recorded interval for hourly data. See the *ReclamationHDB Datastore* appendix for more information.
- Writing NNour data uses WRITE_TO_HDB procedure where the SAMPLE_END_DATETIME is set to the TSTool date/time and SAMPLE_DATE_TIME is set the TSTool date/time minus NHour.
- Writing other than NHour data uses the WRITE_TO_HDB procedure with SAMPLE_DATE_TIME passed as the same value as the TSTool date/time.

• Updating time series records:

- Time series data records for an existing time series will be updated if previously written.
- Missing data:
 - Missing data currently are not written. By convention missing values in HDB are simply not included in the database. Currently the command will not delete previous records if the new value at a date/time is missing.
- Data units:
 - Data units in the time series are not checked against data units in the database because the units in TSTool data may originally have come from various sources that do not use the same units abbreviations as HDB. It is the user's responsibility to ensure that time series that are being written have units that are compatible with HDB.
- Data flags:

- Data flags from the time series are not written to the database. The ValidationFlag, OverwriteFlag, and DataFlags parameters are provided to specify HDB flags. Additional capability may be added in the future.
- Time zone:
 - Time zone can be indicated in TSTool time series by including in the start and end date/time information; however, time zones are difficult to standardize when data comes from different sources. The default time zone for HDB is configured for the Reclamation office that uses the database. If the time series time zone is different from the default (displayed in the note for the TimeZone command parameter in the command editor), it can be specified as a command parameter. It is the user's responsibility to verify that the correct time zone is being used.

• HDB data table:

- The time series interval is used to determine the HDB time series table to write, with irregular data being written as instantaneous data with date/time precision to minute.
- Irregular data also can be written to a specific output table by using the IntervalOverride parameter, for example in cases where a time series was read as irregular but should be treated as hourly in HDB.
- TSTool treats year-interval data generically and does not manage water year (or other types of years) in special fashion, other than when processing data into year interval time series. Water year data can be saved in year interval data but currently there is no way to write to the water-year tables in HDB.

• HDB database procedure:

- The HDB WRITE_TO_HDB stored procedure is used to write individual time series data records:
 - The time series is written to a model time series table if model parameters are specified.
 - The model run date, for single time series and ensembles, is truncated to minutes in time series identifiers and for query purposes.
- When writing ensembles, the HDB procedure

ENSEMBLE.GET_TSTOOL_ENSEMBLE_MRI is used to determine the model run identifier corresponding to model time series and then the WRITE_TO_HDB procedure (above) is used to write data records:

- The ensemble name is determined from the EnsembleName parameter existing names can be selected or a new name can be specified
- The trace number is determined from the EnsembleTraceID command parameter, and will result in the trace being taken from specific time series properties.
- The model name is determined from the EnsembleModelName parameter.
 Model names consistent with non-ensemble model time series are used.
- The model run date is determined from the EnsembleModelRunDate parameter (if specified then the P_IS_RUNDATE_KEY procedure parameter is set to Y, if not specified N).

The following dialog is used to edit the command and illustrates the syntax of the command when writing "real" data, in which case model information is not specified.

& Edit WriteReclamationHDB	() Command						
Write a single "real" or model time series, or write an ensemble of time series to a Reclamation HDB database. The parameters are used to determine database internal numeric primary keys (site_datatype_id and optionally model_run_id for model data). The HDB time series table is determined from the data interval, with irregular data being written to the instantaneous data table. TSTool will only write time series records and will not write records for time series metadata (site, data type, and model data must have been previously defined).							
Specify output date/times to a precision appropriate for output time series.							
Datastore:	Datastore: ReclamationHDB-Dev 🔽 Required - datastore for HDB database.						
TS list:	TS list: AllMatchingTSID 🔽 Optional - indicates the time series to process (default=AllTS).						
TSID (for TSList=AllMatchingTSID):	AAA_DELETE		¥				
EnsembleID (for TSList=EnsembleID):			~				
$\Gamma^{\text{Specify how to match the HDB site_d}}$	latatype_id (required for all time :	series and ensembles)					
Site common name: AAA_DE	ELETE	Required - used with data type common name to determine site_datatype_id.					
Data type common name: storage	×	Required - used with site common name to determine site_datatype_id.					
Matching site_id: 100072 ((9 matches)	Information - useful when comparing to database contents.					
Matching site_datatype_id: 101351		Information - useful when comparing to database contents.					
Site data type ID:	*	Optional - alternative to selecting above choices.					
Specify how to match the HDB model	_run_id (leave blank if not writing	g single or ensemble model time series data)					
Single model time series Ensemble	of model time series						
Use I	these parameters to write a singl	e model time series to HDB.					
	Model name:	Required - used to determine the model_run_id.					
	Model run name: 🛛 💌	Required - used to determine the model_run_id.					
	Model run date: 🛛 💌	Required - YYYY-MM-DD hh:mm, used to determine the model_run_id.					
н	ydrologic indicator: 🛛 💌	Required - used to determine the model_run_id.					
	Selected model_id: No matches	Information - useful when comparing to database contents.					
Selec	ted model_run_id: No matches	Information - useful when comparing to database contents.					
	Model run ID:	Optional - alternative to selecting above choices.					
Agency:	CODWR - Colorado Division of V	/ater Resources 🔽 Optional - agency supplying data (default=	=no agency).				
Validation flag:	A - USGS Approved Data	Optional - validation flag (default=no flag)					
Overwrite flag:	0 - overwrite 🔽	Optional - overwrite flag (default=0).					
Data flags:		Optional - user-defined flag (default=no fl	ag).				
Time zone:	MST 🗸	Optional - time zone for instantaneous and	l daily data (default=MST).				
Output start:	Outout start: Optional - override the global outout start (default=write all data).						
Output end:		Optional - override the global output end (default=write all data).				
	WriteReclamationHDB		="AAA DELETE"				
	SiteCommonName="% a) DFLETF_DataTureCommonName="storage".lengus=COMDRW_Waiteries.						
Command: verwriteFlag="0", TimeZone="MST")							
		Cancel OK					
			vvriteReclamationHDB				

WriteReclamationHDB() Command Editor for "Real" Time Series

The following figure illustrates the syntax of the command when writing "model" data for a single time series, in which case the model parameters are specified via the *Single model time series* tab.

Sedit WriteReclamationHDB	() Command	×					
Write a single "real" or model time series, or write an ensemble of time series to a Reclamation HDB database. The parameters are used to determine database internal numeric primary keys (site_datatype_id and optionally model_run_id for model data). The HDB time series table is determined from the data interval, with irregular data being written to the instantaneous data table. TSTool will only write time series records and will not write records for time series metadata (site, data type, and model data must have been previously defined).							
Datastore:	Datastore: ReclamationHDB-Dev 🗸 Required - datastore for HDB database.						
TS list:	AllMatchingTSID	MatchingTSID Optional - indicates the time series to process (default=AllTS).					
TSID (for TSList=AllMatchingTSID):	AAA_DELETE						
EnsembleID (for TSList=EnsembleID);							
Specify how to match the HDB site_c	Jatatype_id (required for all tim	e series and ensembles)					
Site common name: AAA_DE	ELETE	Required - used with data type common name to determine site_datatype_id.					
Data type common name: storage	v	Required - used with site common name to determine site_datatype_id.					
Matching site_id: 100072 ((9 matches)	Information - useful when comparing to database contents.					
Matching site_datatype_id: 101351		Information - useful when comparing to database contents.					
Site data type ID:		Optional - alternative to selecting above choices.					
Specify how to match the HDB mode	_run_id (leave blank if not writi	ing single or ensemble model time series data)———————————————————————————————————					
Single model time series Ensemble	of model time series						
Model name: CBT AOP RiverWare Required - used to determine the model_run_id. Model name: CBT AOP Test Required - used to determine the model_run_id. Model run date: 2012-07-01 00:00 Required - YYYY-MM-DD hh:mm, used to determine the model_run_id. Hydrologic indicator: Image: Required - used to determine the model_run_id. Selected model_id: 10 Information - useful when comparing to database contents. Selected model_run_id: No matches Information - useful when comparing to database contents. Model run ID: Optional - alternative to selecting above choices.							
Agency:	CODWR - Colorado Division of	Water Resources V Optional - agency supplying data (default=no agency).					
Validation flag:	A - USGS Approved Data	 Optional - validation flag (default=no flag). 					
Overwrite flag:	0 - overwrite 🔽	Optional - overwrite flag (default=0).					
Data flags:		Optional - user-defined flag (default=no flag).					
Time zone:	MST 🔽	Optional - time zone for instantaneous and daily data (default=MST).					
Output start:		Optional - override the global output start (default=write all data).					
Output end:		Optional - override the global output end (default=write all data).					
Command:	Comman: WriteReclamationHDB(DataStore="ReclamationHDB-Dev",TSList=AllMatchingTSID,TSID="AAA_DELETE", SiteCommonName="AAA_DELETE",DataTypeCommonName="storage",ModelName="CBT AOP RiverWare",ModelRunName="CBT AOP Test",ModelRunDate="2012-07-01 00:00",Agency="CODWR",ValidationFlag="A",OverwriteFlag="0",TimeZone="MST")						
Cancel OK							

WriteReclamationHDB() Command Editor for Single Model Time Series

The following figure illustrates the syntax of the command when writing an ensemble of "model" time series, in which case ensemble and related model parameters are specified via the **Ensemble of model** *time series* tab. The TSTool ensemble is specified with the TSList=EnsembleID and EnsembleID parameters.

Sedit WriteReclamationHDB	() Command					
Write a single "real" or model time serie The parameters are used to determine The HDB time series table is determine TSTool will only write time series record	Write a single "real" or model time series, or write an ensemble of time series to a Reclamation HDB database. The parameters are used to determine database internal numeric primary keys (site_datatype_id and optionally model_run_id for model data). The HDB time series table is determined from the data interval, with inregular data being written to the instantaneous data table.					
Specify output date/times to a precision	on appropriate for output time series.					
Datastore:	ReclamationHDB-Dev 💙	Required - datastore for HDB database.				
TS list:	AllMatchingTSID	Optional - indicates the time series to proc	ess (default=AllTS).			
TSID (for TSList=AllMatchingTSID):	AAA_DELETE		~			
EnsembleID (for TSList=EnsembleID):			~			
FSpecify how to match the HDB site_o	datatype_id (required for all time series a	and ensembles)				
Site common name: AAA_DE	ELETE	Required - used with data type common name to determine site_datatype_id.				
Data type common name: current	air temp 💌	Required - used with site common name to determine site_datatype_id.				
Matching site_id: 100072 ((9 matches)	Information - useful when comparing to database contents.				
Matching site_datatype_id: 101355		Information - useful when comparing to database contents.				
Site data type ID:	*	Optional - alternative to selecting above choices.				
-Specify how to match the HDB model	l_run_id (leave blank if not writing single	e or ensemble model time series data)				
Single model time series Ensemble	of model time series					
Use these parameters to write an er	nsemble of model time series to HDB.					
Match an existing ensemble name an	id model run date, or specify new value:	s to create a new ensemble (model name must have been defined).				
If the run date is specified, the ense	emble time series will be uniquely identific	ed with the run date (to the minute).				
Ensemble name: H	lardcodedTest 🗸	Required - used to determine the ensemble model_run_id.				
Ensemble model name: C	IBT AOP RiverWare	Required - used to determine the ensemble model_run_id.				
Ensemble trace ID:	- Select Specifier 🛛 💙 => %z	Optional - use %z for sequence number, etc. or \${TS:property} (default=sequence	e number).			
Ensemble model run date:	×	Optional - YYYY-MM-DD hh:mm, used to determine the ensemble model_run_id (def	ault=run date not used).			
Selected ensemble_id: No	o matches	Information - useful when comparing to database contents.				
Selected ensemble model_id: 10) stermined when command is win	Information - useful when comparing to database contents.				
	scennined when command is run]			
Agency:	CODWR - Colorado Division of Water R	Resources Optional - agency supplying data (default=	=no agency).			
Validation flag:	A - USGS Approved Data	Optional - validation flag (default=no flag)				
Overwrite flag:	0 - overwrite 💙	Optional - overwrite flag (default=0).				
Data flags:		Optional - user-defined flag (default=no fl	ag).			
Time zone:	MST 💙	Optional - time zone for instantaneous and	d daily data (default=MST).			
Output start:		Optional - override the global output start	(default=write all data).			
Output end:		Optional - override the global output end (default=write all data).			
	WriteReclamationHDB(Dat	aStore="ReclamationHDB-Dev", TSList=AllMatchingTSID, TSID	="AAA_DELETE",			
Commande	SiteCommonName="AAA_DEL	ETE",DataTypeCommonName="current air				
commandi	temp",EnsembleTraceID="%z",EnsembleModelName="CBT_AOP					
	RiverWare", Agency="CODWR", ValidationFlag="A", OverwriteFlag="0", TimeZone="MST")					
		Cancel OK				
		WriteR	eclamationHD Ensemble			

WriteReclamationHDB() Command Editor for Ensemble of Model Time Series

The command syntax is as follows:

WriteReclamationHDB(Parameter=Value,...)

Parameter	Description	Default
DataStore	The identifier for the ReclamationHDB data store to use	None – must be
	for the database.	specified.
TSList	Indicates the list of time series to be processed, one of:	Allts
	• AllMatchingTSID – all time series that match the	
	TSID (single TSID or TSID with wildcards) will be	
	processed.	
	• AllTS – all time series before the command.	
	• EnsembleID – all time series in the ensemble will	
	be processed.	
	• FirstMatchingTSID – the first time series that	
	matches the TSID (single TSID or TSID with	
	wildcards) will be processed.	
	• LastMatchingTSID – the last time series that	
	matches the TSID (single TSID or TSID with	
	wildcards) will be processed.	
	• SelectedTS – the time series are those selected	
	with the SelectTimeSeries() command.	
TSID	The time series identifier or alias for the time series to be	Required if
	processed, using the * wildcard character to match	TSList=*TSID.
	multiple time series.	
EnsembleID	The identifier for the TSTool ensemble to be processed, if	Required if TSList=
	processing an ensemble, not to be confused with the	EnsembleID.
	Ensemble* parameters below that match HDB data.	
Site	The site common name for the time series location; used	None – must be
CommonName	with the data type common name to determine the	specified unless
	site_datatype_id in the database.	SiteDataTypeID is
		specified.
DataType	The data type common name for the time series; used	None – must be
CommonName	with the site common name to determine the	specified unless
	site_datatype_id in the database.	SiteDataTypeID is
		specified.
SiteDataTypeID	The site_datatype_id value to match the time series. If	
	specified, the value will be used instead of the	
	site_datatype_id determined from SiteCommonName	
	and DataTypeCommonName.	
	Use the following parameters when reading a single	
ModelName	model time series.	None must be
Modelname	The model name for the time series; used with the model	None – must be
	determine the model run id in the determine	Specified unless
	determine the model_full_id in the database.	specified
ModelRunName	The model run name for the time series: used with the	None – must be
	model name hydrologic indicator(s) and model run date	specified unless
	to determine the model run id in the database	Model Runth is
	to extermine the moder_ran_ia in the database.	specified.
ModelRunDate	The model run date (timestamp) to use for the time series;	None – must be

Command Parameters

Parameter	Description	Default		
	used with the model name, model run name, and	specified unless		
	hydrologic indicator(s) to determine the model_run_id in	ModelRunID is		
	the database. The run date should be specified using the	specified.		
	format YYYY-MM-DD hh:mm (zero-padded with hour 0-	•		
	23, minute 0-59, seconds and hundredths of seconds will			
	default to 0). Need to implement tests to make sure			
	this is properly handled, including formatting and			
	listing existing values.			
Hydrologic	The hydrologic indicator(s) to use for the time series;	None – must be		
Indicator	used with the model name, model run name, and model	specified unless		
	run date to determine the model_run_id in the database.	ModelRunID is		
		specified.		
ModelRunID	The model_run_id value to match the time series. If			
	specified, the value will be used instead of the			
	model_run_id determined from ModelName,			
	ModelRunName, ModelRunDate, and			
	HydrologicIndicator.			
	Use the following parameters when writing an ensemble			
	of model time series.			
EnsembleName	The name of the ensemble to write. The	Must be specified if		
	TSList=EnsembleID and EnsembleID parameters	writing an ensemble.		
	also should be specified.			
EnsembleModelName	The model name corresponding to the ensemble.	Must be specified if		
		writing an ensemble.		
EnsembleTraceID	Indicate how to identify time series trace identifiers:	The time series		
	• %X – use standard time series properties to format the	sequence number		
	ensemble trace ID (see command editor for format	(equivalent to the %z		
	characters)	formatting string)		
	• \${TS:property} – format the trace identifier			
	from time series properties (e.g., properties read from			
	original ensemble data)			
	TSTool and the HDB GET_TSTOOL_ENSEMBLE_MRI			
	procedure currently require the identifier to be an integer			
	– additional options for identifying traces may be added			
	in the future.			
EnsembleModel	When writing an ensemble, the model run date for the	If not specified, the		
RunDate	ensemble, specified using format:	ensemble identifier in		
	• YYYY-MM-DD hh:mm (zero-padded with hour 0-	HDB will not include		
	23)	the model run date.		
	• \${ts:property} – use a run date from a time			
	series property, truncated to minute			
	Need to implement tests to make sure this is properly			
	handled, including formatting and listing existing			
Aconcre	The removing parameters are always appropriate.	No oconce in its its 1		
Agency	I ne agency abbreviation (e.g., USBR) for data records	in detension		
Volidation	Written to the database.	III database.		
valluallOII	ו חטס vanuation hag. Unly uppercase characters are	INO Hag 18 Used.		

Parameter	Description	Default
Flag	supported.	
OverwriteFlag	HDB overwrite flag.	Overwrite (enforced by
		HDB stored
		procedure)
DataFlags	User-defined flags, up to 20 characters.	No flags are used.
TimeZone	Three-letter time zone abbreviation for the data records	Default HDB time
	written to the database.	zone is assumed.
OutputStart	The date/time for the start of the output.	Use the global output
		period.
OutputEnd	The date/time for the end of the output.	Use the global output
		period.

Command Reference: WriteRiversideDB()

Write time series to a RiversideDB database

Version 10.06.00, 2012-04-15

This command is under development - please provide feedback to developers. The

WriteRiversideDB() command writes time series to a RiversideDB database. See the **RiversideDB Data Store Appendix** for more information about the database features and limitations. The command will not define a new time series but will replace or insert the data records for an existing time series. The current functionality allows a single time series to be written by matching a specific existing time series in the database; however, in the future time series properties may be used to write multiple time series with one command (e.g., use %L to match the location).

The following dialog is used to edit the command and illustrates the syntax of the command when writing a single time series to the database. In this case the choices are used to select a matching time series and only one time series can be in the list to process (otherwise a warning will result and nothing is written).

Edit WriteRiversideDB() Com	nand	\mathbf{X}				
This command currently will write only one time series. Use parameters to match a single time series in the database. Use the parameters to match a specific time series; choices will be updated based on previous selections. TSTool will only write time series records. TSTool will not write records for time series metadata (the time series must have been previously defined).						
Data store:	RiversideDB_TSTool	Required - open data store for RiversideDB database.				
TS list:	✓	Optional - indicates the time series to process (default=AllTS).				
TSID (for TSList=AllMatchingTSID);						
EnsembleID (for TSList=EnsembleID):						
_Indicate how to match time series in	database					
Match Specified Single Time Series	Match Time Series Using Properties					
Data type: QIN - II	STANTANEOUS OBSERVED RIVER DI	SCHARGE 🛛 🗸 Required - matching (main) data type in the database.				
Data sub-type: 🔽		Required - matching (sub) data type in the database (may be blank).				
Data interval: 15MINL	ITE 🔽	Required - matching data interval in the database.				
Location (station/area) ID: 013501	01 🔽	Required - matching location ID in the database.				
Data source abbreviation: USGS	*	Required - matching data source in the database.				
Scenario: FILLED	✓	Required - matching scenario in the database (may be blank).				
Sequence number: 🔽		Required - for ensembles, the sequence number (trace starting year).				
Matching MeasLoc_num: 12		Information - useful when comparing to database contents.				
Matching MeasType_num: 744		Information - useful when comparing to database contents.				
Write data flags?:	✓	Optional - should data flags be written? (default=True).				
Output start:		Optional - override the global output start (default=write all data).				
Output end:		Optional - override the global output end (default=write all data).				
Write method:	×	Optional - how to write (default=DeleteInsert).				
Protected flags:		Optional - database flag(s) that indicate values protected from overwrite.				
Revision date/time:		Optional - date/time for revision (default=time that command is run).				
Revision user:		Optional - user that is writing revision (default=operating system user login).				
Revision comment:		Optional - comment for revision (default=no comment).				
Command:	WriteRiversideDB(DataS ource="USGS",DataType=	Store="RiversideDB_TSTool",LocationID="01350101",DataS ="QIN",Interval="15MINUTE",Scenario="FILLED")				
		Cancel OK				
		WriteRiversideDE				

WriteRiversideDB() Command Editor when Matching/Writing a Single Time Series

The following figure illustrates use of the *Match Time Series Using Properties* tab, which is envisioned as a future enhancement. In this case, the parameter values do not match a specific time series in the database but instead use the properties from the list of time series being written to indicate how to match a time series in the database. For example, DataType=%T would use the data type from the time series to match the data type in the database and Location=%L would use the location identifier from the time series to match the location identifier in the database. Writing multiple time series requires fewer commands but the command editor will not be able to confirm that the time series are matched in the database (this can only occur when running the commands and data are actually processed).

Indicate how to match time series in database	
Match Specified Single Time Series Match Time Series Using Properties	
In the future matching time series will be specified using time series properties (e.g., %L for location), to allow multipl	time series to be written with one command.
	WriteRiversideDB

WriteRiversideDB() Command Editor when Matching/Writing Multiple Time Series

The following technical issues apply when writing time series:

- Currently there is no authentication to prevent users from using this command. Options will be explored based on specific system requirements.
- Time series being written must have compatible units with the units of the time series in the database. The same units or conversion factor of 1.0 must be detected to allow writing.
- Missing values in time series are written as missing values (null) in the database. This allows missing values in the database to have flags.
 - Regular interval time series could be written in such a way that missing values are simply not written to the database. If this is required, then a new parameter
 WriteMissingValues could be added. However, tracking revisions might be difficult using this approach because older values would need more complex handling since no newer missing value record would be present.
 - Irregular interval time series in TSTool do not have missing value records unless they were specifically read from input.
- Irregular time series present challenges that may not be fully addressed with the current software features and may require additional enhancements. For example, WriteMethod=TrackChanges may not work properly if updates to irregular data have different date/times than the original values. In this case WriteMethod=DeleteInsert may be more appropriate; however, this does not check the ProtectedFlags parameter.

The command syntax is as follows:

```
WriteRiversideDB(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
DataStore	The identifier for the RiversideDB data store to use for the	None – must be
	database.	specified.
TSList	Indicates the list of time series to be processed, one of:	Allts
	• AllMatchingTSID – all time series that match the	
	TSID (single TSID or TSID with wildcards) will be	
	processed.	
	• AllTS – all time series before the command.	
	• EnsembleID – all time series in the ensemble will be	
	processed.	
	• FirstMatchingTSID – the first time series that	
	matches the TSID (single TSID or TSID with	
	wildcards) will be processed.	
	• LastMatchingTSID – the last time series that	
	matches the TSID (single TSID or TSID with	
	wildcards) will be processed.	
	• SelectedTS – the time series are those selected with	
	the SelectTimeSeries() command.	
TSID	The time series identifier or alias for the time series to be	Required if
	processed, using the * wildcard character to match multiple	TSList=*TSID.
TranspillerD	time series.	
EnsempterD	The ensemble to be processed, if processing an ensemble.	Required if TSList=
	This parameter is equipiered, if quiting multiple time equip	EnsembleID.
WriteMode	in is parameter is envisioned if writing multiple time series	writeSingle
	is enabled in the future (currently only writesingle is	
	of	
	• WriteSingle (the only mode that is enabled) in	
	which case the <i>Match Specified Single Time Series</i>	
	tab is used to enter command parameters.	
	• WriteMultiple (future enhancement), in which	
	case the Match Time Series Using Properties tab is	
	used to enter command parameters.	
	Ensemble time series may fit into either of the above	
	depending on how the sequence numbers are handled.	
DataType	The data type abbreviation in the database to match.	None – must be
		specified.
DataSubType	The data sub-type in the database to match (may be blank).	Not used if blank.
IIILEIVAL	The data interval in the database to match.	None – must be
LocationID	The location identifier in the database to match	None – must he
LOCACIONID	The location identifier in the database to match.	specified.
DataSource	The data source abbreviation in the database to match.	Not used if blank.
Scenario	The scenario in the database to match (may be blank).	Not used if blank.
SequenceNumber	Used for ensembles – the trace number, often the starting	Not used if blank.
	year of the trace (may be blank).	
WriteDataFlags	Indicate whether data flags should be written.	True

Parameter	Description	Default
OutputStart	The date/time for the start of the output.	Use the global output
		period.
OutputEnd	The date/time for the end of the output.	Use the global output
		period.
WriteMethod	The approach for writing time series, one of:	None – must be
	• Delete – delete the time series records but do not	specified.
	write the time series (useful for testing and database maintenance)	
	• DeleteInsert – delete time series records and then	
	insert. Revision data are not inserted into the database.	
	• TrackRevisions – track revisions as per the logic	
	described after this table.	
ProtectedFlags	Indicate the flag values for database data records that	None – no data will be
	should NOT be modified, when	protected.
	WriteMethod=TrackRevisions. Typically these	
	records indicate that data have been validated and/or	
	manually entered and automated processes should not	
	overwrite the values. Currently only one flag value can be	
	specified; however, multiple values or patterns may be	
	supported in the future.	
ComparePrecision	The number of digits after the decimal used when	4
	comparing previous database values with values in the time	
	series. This may be required where input/output operations	
	result in truncations or round-off.	
RevisionDateTime	The date/time for a new revision, if needed, one of:	Time from computer
	 CurrentToSecond syntax – see 	system clock at the
	SetOuputPeriod() command for more	time the command is
	information.	run.
	 \${Property} – property value string 	
	• YYYY-MM-DD hh:mm or similar date/time string	
RevisionUser	User identifier for revisions. Currently this is NOT taken	User's login from the
	from the RiversideDB user table. \${Property} notation	operating system.
	can be used.	
RevisionComment	Comment for revisions. \${Property} notation can be	No comment.
	used.	

Revision information will be utilized only if configured in the database. The time series data table must contain a Revision_num column and the Revision table must exist. The time series table layout information will indicate whether revision numbers are used. If WriteMethod=DeleteInsert and RevisionComment is provided, then all old data will be deleted (all revisions) and a new revision will be added corresponding to all inserted records. This ensures that the database size does not grow quickly. If RevisionComment is not specified, then a revision will not be added in the database (revision will be set to zero, which corresponds to the "original data" revision).

The following figure explains the logic when WriteMethod=TrackRevisions, which can be used when RiversideDB is configured with time series tables that use the Revision_num column. The following flow chart focuses on the case where a data record to write to the database (consisting of a date/time, a value, and a flag) already exists for the date/time. This indicates that this data record

previously has been written to the database. In this case, the existing data record in the database should be overwritten with the new record unless the existing record is flagged as protected (in this example, manually adjusted with ProtectedFlags=M). In this case, the manually adjusted value persists and can be overwritten only if the new data record also is flagged with M, indicating that an additional manual adjustment has occurred. The following figure illustrates the logic performed for each value. However, several steps are performed in bulk to improve performance.



Revisions to existing data records are stored in the Revision table in RiversideDB. Original data records in the time series tables have a revision number of '1', which refers to the revision number '1' in the Revision table. Any time a new revision is needed due to changes in a data value, a new entry is created in the Revision table, populated with the pertinent information (date/time of revision, user, and a comment) and the corresponding revision number is assigned to the revised data record in the time series table. Revision numbers are not incremented for each data value but are incremented for the bulk database operation (similar to a commit in content management systems). Care should be taken in using WriteRiversideDB() commands in order to minimize the number of revisions that are tracked. For example, rather than relying on the default value for the RevisionDateTime command parameter, a better approach may be to define a property at the top of the command file using a SetProperty() command and then refer to that property when specifying the RevisionDateTime property. This will ensure that multiple WriteRiversideDB() commands in a single command file utilize the same revision information.

If a data revision is detected based on the logic in the above figure, the corresponding revision will be searched for in the Revision table. If not found, then a new revision record will be inserted into the Revision table and that information will be used in the time series data table.

Time Series Table					Revision	Tabl	e		
MeasType_num	Date_Time	Val	Revision_num	Quality_flag]				
1	2011-04-11 14:35	585.98	1						
1	2011-04-11 14:50	586.02	1]				
1	2011-04-11 15:00	585.98	1]				
1	2011-04-11 15:10	585.98	1			Revision num	Date Time	llsor	Comment
1	2011-04-11 15:10	585.97	2	Μ	1	Revision_num	Date_mile	0361	comment
1	2011-04-11 15:15	585.99	1			1			NO REVISION
1	2011-04-11 15:15	585.98	3	М	í	2	2011-04-11 17:00	JP	Manual Change
1	2011-04-11 15:15	585.97	4	M		→ 3	2011-04-11 17:01	JP	Manual Change
1	2011-04-11 15:25	586	1			4	2011-04-11 18:01	JP	Manual Change after review

TSTool commands that read RiversideDB time series, such as the ReadRiversideDB() command, sort time series data records by the Date_Time and revision number (Revision_num) prior to transferring the data into data objects. Consequently, the entry with the largest revision number for records at the same sensor (MeasType_num) and time (Date_Time) is used as the valid data record because the record will be processed last. In the above example, this is the record with Revision_num 4 and a value of 585.97. This approach may result in performance degradation over time if many revisions are made to data values for the same date/time and consequently it may be appropriate to remove or archive very old revisions as part of database maintenance. The trade-off between performance and the ability to track revisions may vary between systems and in general there should be few revisions for the same data point because data loading will move forward through time without reloading the entire period.

Command Reference: WriteRiverWare()

Write time series to a RiverWare format file

Version 08.15.00, 2008-05-12

The WriteRiverWare() command writes one time series to the specified RiverWare format file. See the **RiverWare Input Type Appendix** for more information about the file format.

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

🗼 Edit WriteRiverWare() Command 🛛 🛛 🔀			
Write time series to a Rive	Write time series to a RiverWare format file, which can be specified using a full or relative path (relative to the working directory).		
The working directory is:	The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\WriteRiver\Vare		
It is recommended that the	It is recommended that the file name follow the convention ObjectName.SlotName.		
The time series to process are indicated using the TS list.			
If TS list is "AllMatchingTS	ID", pick a single time series, or enter a wildcard time series identifier pattern.		
Only the first time series v	will be written (limitation of file format).		
TS list:	AIITS How to get the time series to write (default=AIITS).		
Identifier (TSID) to match:	V		
RiverWare file to write:	08213500.Month.RiverWare Browse		
Units:	Default is time series units. Data are not converted.		
Scale:	The default is 1.		
Set_units:	The default is to not write.		
Set_scale:	The default is to not write.		
Precision:	2 Number of digits after decimal (default=4).		
	WriteRiverWare(TSList=AllTS,OutputFile="08213500.Month.RiverWare",Precision=2)		
Command:			
	Add Working Directory Cancel OK		
	WriteRiverWare		

WriteRiverWare() Command Editor

The command syntax is as follows:

```
WriteRiverWare(Parameter=Value, ...)
```

Command Parameters

Parameter	Description	Default
TSList	 Indicate how to determine the list of time series to process (only first one is written), one of: AllMatchingTSID – process time series that have identifiers matching the TSID parameter. AllTS – process all the time series. SelectedTS – process the time series that are selected (see SelectTimeSeries()). 	None – must be specified.
TSID	Used if TSList=AllMatchingTSID to indicate the time series identifier or alias for the time series to be filled. Specify * to match all time series or use a wildcard for one or more identifier parts.	Required if TSList=AllMatchingTSID.
OutputFile	The RiverWare file to write. The path to the file can be absolute or relative to the working directory. The Browse button can be used to select the file to write (if a relative path is desired, delete the leading path after the select).	None – must be specified.
Units	The data units to be output. Specify units that are recognized by RiverWare – the units are not actually converted but the new units string is used in the output file.	Use the units in the time series.
Scale	See the RiverWare Input Type Appendix.	1
Set_units	See the RiverWare Input Type Appendix.	Set_units are not output.
Set_scale	See the RiverWare Input Type Appendix.	Set_scale are not output.
Precision	The number of digits after the decimal to write.	4

A sample command file to write data from the State of Colorado's HydroBase is as follows:

```
# 08213500 - RIO GRANDE RIVER AT THIRTY MILE BRIDGE NEAR CREEDE
08213500.DWR.Streamflow.Month~HydroBase
WriteRiverWare(TSList=AllTS,OutputFile="08213500.Month.RiverWare",Precision=2)
```

Command Reference: WriteStateCU()

Write time series to a StateCU format file

Version 08.16.04, 2008-09-04

The WriteStateCU() command writes time series to the specified StateCU frost dates format file. **Currently only the frost dates file can be written with this command.** See the WriteStateMod() command to write StateMod format files (e.g., for the precipitation, temperature, and diversion time series files used with the StateCU model). See the **StateCU Input Type Appendix** for more information about the file format. All time series matching the data types FrostDateL28S, FrostDateL32S, FrostDateF32F, and FrostDateF28F will be written (all other time series will be ignored). Other StateCU files may be supported in the future. See also the StateDMI software, which processes other StateCU files.

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

🚯 Edit WriteStateCU() Command			
THIS COMMAND CURRENTLY ONLY WRITES StateCU FROST DATE FILES.			
Time series with data types FrostDateL28S, FrostDateL32S, FrostDateF32F, and FrostDateF28F are processed.			
Write frost date time series to a StateCU format file, which can be specified using a full or relative path (relative to the working directory).			
The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\WriteStateCU			
The Browse button can be used to select an existing file to overwrite (or edit the file name after selection).			
StateCU file to write: Results/test.fst Browse			
Output start: Overrides the global output start, specify as YYYY.			
Output end: Overrides the global output end, specify as YYYY.			
WriteStateCU(OutputFile="Results/test.fst")			
Command:			
Add Working Directory Cancel OK			

WriteStateCU() Command Editor

The command syntax is as follows:

WriteStateCU(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
OutputFile	The StateCU frost dates output file. The	None – must be specified.
	path to the file can be absolute or relative	
	to the working directory.	
OutputStart	The date/time for the start of the output.	Use the global output period.
OutputEnd	The date/time for the end of the output.	Use the global output period.

A sample command file is as follows, using data from the State of Colorado's HydroBase database:

0109 - AKRON 4 E 0109.NOAA.FrostDateL28S.Year~HydroBase 0109.NOAA.FrostDateL32S.Year~HydroBase 0109.NOAA.FrostDateF32F.Year~HydroBase 0109.NOAA.FrostDateF28F.Year~HydroBase WriteStateCU(OutputFile="test.stm")

Command Reference: WriteStateMod()

Write time series to a StateMod format file

Version 08.15.00, 2008-05-12

The WriteStateMod() command writes the time series in memory to the specified StateMod format file. See the **StateMod Input Type Appendix** for more information about the file format. It is expected that the time series have the same interval. The time series identifier location part is written as the identifier, even if an alias is assigned to a time series.

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

💊 Edit WriteStateMoo	d() Command		
Write time series to a StateMod format file.			
It is recommended that the	It is recommended that the file name be relative to the working directory.		
The working directory is:	The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\W/riteStateMod		
The Browse button can be used to select an existing file to overwrite (or edit the file name after selection).			
For the precision, a negat	For the precision, a negative integer allows auto-adjustment to prevent overflow.		
A precision of -2001 will o	default to 2 digits, adjusted for overflow, and also use no decimal (special precision option).		
The time series to proces:	s are indicated using the TS list.		
If TS list is "AllMatchingTS	ID", pick a single time series, or enter a wildcard time series identifier pattern.		
TS list:	AITS How to get the time series to write.		
Identifier (TSID) to match:	▼		
StateMod file to write:	RioGrande.rih Browse		
Output start:	Overrides the global output start.		
Output end:	Overrides the global output end.		
Missing value:	Value to write for missing data (default=-999).		
Output precision:	Digits after decimal (default=-2).		
	WriteStateMod(TSList=AllTS,OutputFile="RioGrande.rih")		
Command:			
	Add Working Directory Cancel OK		
	WriteStateMc		

WriteStateMod() Command Editor

The command syntax is as follows:

WriteStateMod(Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSList	Indicate how to determine the list of time	None – must be specified.
	series to process, one of:	
	 AllMatchingTSID – process time 	
	series that have identifiers matching the	
	ISID parameter.	
	• AllTS – process all the time series.	
	 Selected TS – process the time series that are selected (see 	
	that are selected (see	
תדפיד	SelectimeSeries()).	Required if
1510	indicate the time series identifier or alias for	Required in
	the time series to be filled. Specify * to	ISLISC-AIIMACCHINGISID.
	match all time series or use a wildcard for	
	one or more identifier parts	
OutputFile	The StateMod file to write. The path to the	None – must be specified.
-	file can be absolute or relative to the	
	working directory (command file location).	
OutputStart	The date/time for the start of the output.	Use the global output period.
OutputEnd	The date/time for the end of the output.	Use the global output period.
MissingValue	The value to write for missing data.	-999
Precision	The number of digits to use after the	The default output precision if
	decimal point, for data values. A negative	not specified is -2 , which is then
	number indicates that if the formatted	reset based on the data units (see
	number is larger than the allowed output	the system\DATAUNIT file).
	width, adjust the format accordingly by	
	truncating fractional digits. A special value	
	of -2001 is equivalent to -2 and	
	additionally NO decimal point will be	
	printed for large values.	

A sample command file to process data from the State of Colorado's HydroBase is as follows:

```
SetOutputPeriod(OutputStart="1950-01",OutputEnd="2002-12")
# 08213500 - RIO GRANDE RIVER AT THIRTY MILE BRIDGE NEAR CREEDE
08213500.DWR.Streamflow.Month~HydroBase
# 08217000 - RIO GRANDE AT WASON, BELOW CREEDE, CO.
08217000.USGS.Streamflow.Month~HydroBase
WriteStateMod(TSList=AllTS,OutputFile="RioGrande.rih")
```

Command Reference: WriteSummary()

Write time series to a summary format file

Version 09.07.00, 2010-08-09

The WriteSummary() command writes time series to a summary report file, as text or HMTL. The format of the file is a default for the data interval. The total/average column in reports (if output) is based on the units – a parameter may be added in the future to allow more flexibility.

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

🕥 Edit WriteSummary() Command 🛛 🛛 🔀				
Write time series to a summary format file, which can be specified using a full or relative path (relative to the working directory).				
The working directory is: C:\Develop\T Specify the file extension as "html" to	The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\WriteSummary			
	Die Gree de Chaser fleur hut		Durana	
Summary file to write:	Riograndestreamnow.txt		browse	
TS list:	Alits 💌	Optional - indicates the time series to process (default=AlITS).		
TSID (for TSList=AllMatchingTSID):			~	
EnsembleID (for TSList=EnsembleID):			~	
Output start:		Optional - output start (default=use global output period or write all data).		
Output end:		Optional - output end (default=use global output period or write all data).		
Output year type:	~	Optional - output year type (default is global output year type).		
	WriteSummary(TSList=	AllTS,OutputFile="RioGrandeStreamflow.txt")		
Command:				
Add Working Directory Cancel OK				
			WriteSummary	

WriteSummary() Command Editor

The command syntax is as follows:

```
WriteSummary(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
OutputFile	The summary file. The path to the file can be absolute or relative to the working directory (command file location). Specifying a filename with an "html"	None – must be specified.
	extension will result in HTML output, which is color- coded for missing values and has notes for flagged values.	
TSList	 Indicates the list of time series to be processed, one of: AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will be processed. AllTS – all time series before the command. EnsembleID – all time series in the ensemble will be processed. FirstMatchingTSID – the first time series that matches the TSID (single TSID or TSID with wildcards) will be processed. LastMatchingTSID – the last time series that matches the TSID (single TSID or TSID with wildcards) will be processed. 	AllTS
	 wildcards) will be processed. SelectedTS - the time series are those selected with the SelectTimeSeries() command. 	
TSID	The time series identifier or alias for the time series to be processed, using the * wildcard character to match multiple time series.	Required if TSList=*TSID.
EnsembleID	The ensemble to be processed, if processing an ensemble.	Required if TSList= EnsembleID.
OutputStart	The date/time for the start of the output.	Use the global output period.
OutputEnd	The date/time for the end of the output.	Use the global output period.
OutputYearType	The output year type, in particular for formatting monthly and daily time series.	Calendar

A sample command file to process data from the State of Colorado's HydroBase is as follows:

```
SetOutputPeriod(OutputStart="1950-01",OutputEnd="2002-12")
# 08213500 - RIO GRANDE RIVER AT THIRTY MILE BRIDGE NEAR CREEDE
08213500.DWR.Streamflow.Month~HydroBase
# 08217000 - RIO GRANDE AT WASON, BELOW CREEDE, CO.
08217000.USGS.Streamflow.Month~HydroBase
WriteSummary(TSList=AllTS,OutputFile="RioGrandeStreamflow.txt",TSList="AllTS")
```

Command Reference: WriteTableToDataStore()

Write a table to a datastore

Version 10.18.00, 2013-03-03

This command is under development and has the following limitations:

- Although some error handling has been implemented, it is not very detailed. Improvements will be made in response to exercising the command functionality.
- Write statements are created for each row of the table being written. This is inefficient and slow. Improvements will be made in future updates.
- Functionality has been tested mainly with SQL Server.
- Handling of date objects has not been tested.
- Better handling of blank rows needs to be implemented.

The WriteTableToDataStore() command processes each row in a table and executes an SQL statement to insert the row into a database datastore. If database datastore support is not specifically provided by TSTool, a generic datastore can be used (see the **Generic Database Datastore** appendix). This command cannot be used with web service datastores and use with Excel datastores has not been tested. This command is useful in particular for bulk data loading such as for database initialization and when tight integration with TSTool is not required or has not been implemented. In the future additional command parameters may be added to limit the rows that are being written and allow update functionality.

General constraints on the query are as follows:

- the table or views being written must be writeable by the user specified for the database connection (some databases restrict direct access to data and require using stored procedures)
- the table column names must match the database table column names (in the future a command parameter may be added to allow column names to be mapped)
- data types for table columns must closely match the database:
 - internally an SQL statement is created in which data values are formatted as per the data type (e.g., strings are quoted); consequently column types must be appropriate to generate correct formatting
 - the full precision of floating point numbers is passed to the database (formatting for display will not apply to values written to the database)
 - o null values in the table will transfer to null values in the database
 - date/time columns in the table will be represented as such in the database table; however, it may not be possible to limit the precision of the date/time (i.e., hours, minutes, and seconds may be shown with default zero values in output)
- the specified table columns are written (all are written by default)
 - primary keys in the database table do not need to be specified (their values will be assigned automatically)
 - table columns that correspond to related tables in the datastore table need to be mapped using the DataStoreRelatedColumnsMap command parameter

An example of column mapping to a related table is as follows, using the notation Table.Column to fully identify columns:

• the string TableID.DataType column is in the input data

• an integer database table TimeSeriesMeta.DataTypesID column is a foreign key to DataTypes.DataTypesID, and DataTypes.Abbreviation is the string data type – in other words, the datastore column being written does not match the string data type, but uses a relationship to match the integer data type in a separate table

To handle this relationship:

• Use the ColumnMap parameter to tell the command that the DataType column in input table maps to the DataTypesID column in the datastore table:

ColumnMap="DataType:DataTypesID"

• Use the DataStoreRelatedColumnsMap parameter to tell the command that the DataTypesID column should be looked up the Abbreviation column, which is a second level of column mapping:

DataStoreRelatedColumnsMap="DataTypesID:Abbreviation"

The following dialog is used to edit the command and illustrates the syntax for the command, in this case writing a table to a datastore that was defined as a GenericDatabaseDataStore.

\delta Edit WriteTableToDataStore() Command			
This command writes a table to a database datastore table or view. The table column names and types by default must match the database table columns but can be mapped with the ColumnMap parameter. The write mode can impact performance and should be consistent with data management processes.			
Table ID:	Excel_InsightDataIntervalTypes	Required - identifier of table to write.	
Table columns to write:		Optional - columns from TableID, separated by commas (default=all).	
Table columns to NOT write:		Optional - columns from TableID, separated by commas (default=all).	
Datastore:	INSIGHT-FABAnalysis-2012 💌	Required - database datastore to receive data.	
Datastore table/view:	InsightDataIntervalTypes	Required - database table/view to receive data.	
Table to datastore column map:		Optional - if column names differ (default=names are same).	
Datastore related columns map:		Optional - if table column matches related table column.	
Write mode:		Optional - how to write (default=InsertUpdate).	
Command:	WriteTableToDataStore(TableID="Excel_InsightDa alysis-2012",DataStoreTable="InsightDataInterv	taIntervalTypes",DataStore="INSIGHT-FABAn alTypes")	
Cancel OK			
		WriteTableToDataStore	

WriteTableToDataStore() Command

The command syntax is as follows:

WriteTableToDataStore(Parameter=Value, ...)
Parameter	Description	Default		
TableID	Identifier for table to write	None – must be specified		
IncludeColumns	The names of the table columns to write	All columns from		
	separated by commas	Table ID are written		
FygludeColumng	The names of table columns NOT to write	All columns from		
Excludecolulins	separated by common This will override			
	Include Columna	Tablelb are written.		
DataStoro	The name of a database datastore to receive	Nona must be specified		
Datastore	deta	None – must be specified.		
DataStoroTable	The name of the detenage table or view to	Nona must be specified		
Datastorerabie	receive deta	None – must be specified.		
ColumnMan	Indicate which columns in Upble ID have	DatagtoroTableName		
Сотантар	life and a second by the second secon	columns are assumed to		
	different names in DataStoreTable,	match the column names in		
	using the syntax:			
	ColumnName: Datastore l'ableName,	Tableib		
	ColumnName: DatastoreTableName,			
Detectementeleted	m To direct a detectory college of the days of the			
DataStoreRelated	Indicate datastore columns that need to			
Corumismap	match values in a related table in the	columns are assumed to		
	datastore. For example, TableID may	match the column names in		
	contain a column "Abbreviation" but the	Table1D, with no need to		
	corresponding column in	perform reference table		
	DataStoreTable may refer to a related	value matching.		
	table using a foreign key relationship			
	(matching integer column in both tables). It			
	is expected that the related table will have			
	only one primary key column, which will be			
	determined automatically. However, a			
	column mapping must be provided to tell the			
	command which DataStoreTable			
	column should be matched with the related			
	table. The syntax of the parameter is:			
	DataStoreTableColl:RelatedTableColl,			
	The above assumes that foreign keys have			
	been defined in the DataStoreTable			
	columns. If the database does not explicitly			
	define a foreign key relationship in the			
	database design, then specify the right side			
	of the map as:			
	RelatedTable1.RelatedCol1.			
WriteMode	The method used to write data, recognizing	InsertUpdate		
	the databases use insert and update SQL			
	statements, one of:			
	• DeleteInsert – delete the data first			
	and then insert (all values will need to be			
	matched to delete)			
	• Insert – insert the data with no			

Parameter	Description	Default
	attempt to update if the insert fails	
	• InsertUpdate – try inserting the data	
	first and if that fails try to update	
	• Update – update the data with no	
	attempt to insert if the update fails	
	• UpdateInsert – try updating the data	
	first and if that fails try to insert	

This page is intentionally blank.

Command Reference: WriteTableToDelimitedFile()

Write a table to a delimited file

Version 10.20.00, 2013-03-25

The WriteTableToDelimitedFile() command writes a table to a delimited file. Currently only the comma is supported as the delimiter. This command is the analog to the ReadTableFromDelimitedFile() command. It can be used to provide tabular data to other programs, such as spreadsheet programs and geographic information systems.

The default is to write a standard file header using comment lines that start with the # character. If available, column names will be written in double quotes as the first non-comment row. Formatting for cell values is limited and the default precision of floating point numbers may include too many digits – this will be addressed in future updates.

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

Edit WriteTableToDelimitedFile() Command					
Write a table to a delimited	ed format file, which can be specified using a full or relative path (relative to the working directory).				
The delimiter is a comma, h	header comment lines start with #, and column headings are the first non-comment line.				
The working directory is: C	$\verb C:\end{tabular} C:tab$				
Output file to write:	: Results\Test_WriteTableToDelimitedFile_AlwaysQuoteStrings_out.csv Br	owse			
Table to write:	: table1 💙 Required - table identifier.				
Write header comments?:	: Optional - should header comments be written? (default=True).				
Always quote strings?:	True 💌 Optional - always quote strings? (default=False, only quote if delimiter in string).				
Newline replacement:	Optional - replacement for newline character (use \t for tab or \s for space).				
Command: WriteTableToDelimitedFile(TableID="table1",OutputFile="Results\Tes t_WriteTableToDelimitedFile_AlwaysQuoteStrings_out.csv",AlwaysQuot eStrings=True)					
Add Working Directory Cancel OK					

WriteTableToDelimitedFile() Command Editor

WriteTableToDelimitedFile(Parameter=Value,...)

Parameter	Description	Default	
TableID	Identifier for the table to write.	None – must be specified.	
OutputFile	The name of the file to write, as an	None – must be specified.	
	absolute path or relative to the command		
	file location.		
WriteHeaderComments	Indicates whether to write the header	True	
	comments, True or False. Some		
	programs, such as Esri's ArcGIS do not		
	handle delimited files with comments.		
AlwaysQuoteStrings	Indicates whether values in string	False	
	columns should always be surrounded		
	by double quotes:		
	• False – only quote strings that		
	contain the delimiter		
	• True – always quote strings		
	An example of using		
	AlwaysQuoteStrings=True is to		
	quote identifiers that have a leading zero		
	(e.g., 01234567). Not quoting may		
	cause the values to be interpreted as		
	integers when read from the delimited		
	file.		
NewlineReplacement	The string to replace newlines in string	Do not replace newlines.	
	values, necessary to prevent unexpected		
	line breaks in output rows. In order to		
	handle newlines from various systems,		
	the following patterns are replaced in		
	sequence:		
	• \r\n		
	• \n		
	• \r		
	The following special parameter values		
	are recognized:		
	• $\t - replace$ newline with tab		
	• \sames – replace newline with space		

Command Reference: WriteTableToHTML()

Write a table to an HTML file Version 09.10.01, 2010-12-07

The WriteTableToHTML() command writes a table to an HTML file. It can be used to publish tables to the web.

Table column names are output as the HTML table column headers. Formatting for cell values is based on the precision of the original table data. Default styles are written at the top of the HTML. In the future the command may accept styles as input.

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

\delta Edit WriteTableToHTML() Command 🛛 🛛 🔀				
Write a table to an H	1TML file, which can be specified using a full or relative path (relative to the working di	irectory).		
The working director	y is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\Write	eTableToHTML		
Output file to write:	Results\Test_WriteTableToHTML_out.html Browse			
Table to write:	KNN_Aspinall Required - table identifier.			
Command:	WriteTableToHTML(TableID="KNN_Aspinall",OutputFile= ts\Test_WriteTableToHTML_out.html")	"Resul		
Add Working Directory Cancel OK				
		WriteTableTol		

WriteTableToHTML() Command Editor

WriteTableToHTML(Parameter=Value,...)

Parameter	Description	Default
TableID	Identifier for the table to write.	None – must be specified.
OutputFile	The name of the file to write, as an absolute path or relative to the command file location.	None – must be specified.

Command Reference: WriteTimeSeriesProperty()

Write a time series property to a file

This command is under development and is used primarily for software testing. In particular one limitation is that the time series identifier is not included in output and therefore properties for multiple time series are not uniquely identified.

The WriteTimeSeriesProperty() command writes the value of a time series property to a file. This command should not be confused with the WriteProperty() command, which writes processor properties. This is useful for testing whether properties are being set. It could also be used to pass information from TSTool to another program. The format of the output is:

Property="Value"

Multi-line properties will be contained within the quotes. The number of properties is limited at this time, as needed for testing software, but may be increased in the future. The format of output may also change in the future.

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

Edit WriteTimeSeriesPropert	r() Command	×		
This command is experimental.				
Write a time series property to a file.	This is useful for automated software testing.			
It is recommended that the output file I	pe relative to the current working directory.			
The working directory is: C:\Develop\	STool_SourceBuild\TSToolttest/regression\commands\generalW\/riteTimeSeriesPropert	У		
TS list:	Indicates the time series to process (default=AIITS).			
TSID (for TSList=AllMatchingTSID):		~		
EnsembleID (for TSList=EnsembleID):		~		
Property file to write:	Results/Test_WriteTimeSeriesProperty_PropertyName=DataLimitsOriginal_out.txt	Browse		
Property to write:	DataLimitsOriginal Properties are maintained by the command processor.			
Append to file?:	False Default=True			
	WriteTimeSeriesProperty(OutputFile="Results/Test_WriteTimeS	eriesProp		
Command:	erty_PropertyNameDataLimitsOriginal_out.txt",PropertyName="1	DataLimit		
Command.	sOriginal",Append=False)			
	Add Working Directory Cancel OK			
	WriteT	imeSeriesProperty		

WriteTimeSeriesProperty() Command Editor

```
WriteTimeSeriesProperty(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default		
TSList	Indicates the list of time series to be processed, one of:	AllTS		
	 AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will be modified. AllTS – all time series before the command. 			
	• EnsembleID – all time series in the ensemble will be modified.			
	• FirstMatchingTSID – the first time series that matches the TSID (single TSID or TSID with wildcards) will be modified.			
	 LastMatchingTSID – the last time series that matches the TSID (single TSID or TSID with wildcards) will be modified. 			
	• SelectedTS – the time series are those selected with the SelectTimeSeries() command.			
TSID	The time series identifier or alias for the time series to be modified, using the * wildcard character to match multiple time series.	Required when TSList=*TSID		
EnsembleID	The ensemble to be modified, if processing an ensemble.Required when TSList=Ensemble			
OutputFile	The property file to write, as an absolute path or relative to the command file.	None – must be specified.		
PropertyName	The property name to write.	None – must be specified.		
Append	Indicates whether the property should be appended to the file (True) or overwrite the file (False).	True		

A sample command file is as follows:

WriteTimeSeriesProperty(OutputFile="Results/Example_WriteTimeSeriesProperty.txt", PropertyName="DataLimitsOriginal")

Command Reference: WriteTimeSeriesToDataStore()

Write time series to a database datastore

Version 10.21.00, 2013-06-28

The WriteTimeSeriesToDataStore() command writes time series to the specified database datastore. This command can only write to databases that have a supported design structure. Currently this command is only available for generic datastores (see the **Generic Database Datastore** appendix for information about supported database designs and datastore configuration properties). This command cannot be used with web service datastores and use with Excel datastores has not been tested. This command is useful in particular for bulk data loading such as for database initialization and when tight integration with TSTool is not required or has not been implemented.

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

👌 Edit V	🕲 Edit WriteTimeSeriesToDataStore() Command 💦 💈 💈					
Write time :	Write time series to a database datastore, where time series to database table mapping is defined in the datastore configuration.					
Enter date; -Specify ti	(times to a preci me series inform	ision appropriate h nation	or output time se	eries.		
Spoce, y ca		TS list:		~	Optional - indicates the time series to process (default=AlITS).	
TSID (f	or TSList=AllMa	tchingTSID):			×	
Ensemble:	ID (for TSList=E	insembleID):			~	
	C	Output start:			Optional - override the global output start (default=write all data).	
		Output end:			Optional - override the global output end (default=write all data).	
-Specify d	atastore inform	ation-				
	Datastore:	INSIGHT-FABAna	lysis-2012 🛛 👻	Required - (latabase datastore to receive data.	
Datastore	e location type:	POD		Optional - s	pecify location type to use in datastore (default=time series location type).	
Datastor	re data source:	WISKI		Optional - s	pecify data source to use in datastore (default=time series data source).	
Datast	ore data type:	FieldDeliveries		Optional - s	pecify data type to use in datastore (default=time series data type).	
Data	astore interval:	Year		Optional - s	pecify data interval to use in datastore (default=time series data interval).	
Data	store scenario:	Irrig		Optional - s	pecify scenario to use in datastore (default=time series scenario).	
Datastore	e missing value:	null		Optional - v	alue to write for missing data (default=time series missing value).	
D	atastore units:	af		Optional - s	pecify units to use in datastore (default=time series units).	
	Write mode:	DeleteAllThenIns	ert 💌	Optional - h	ow to write data records (default=InsertUpdate).	
WriteTimeSeriesToDataStore(DataStore="INSIGHT-FABAnalysis-2012",DataStoreLocation						
Command:	Command: Type=POD, DataStoreDataSource=WISKI, DataStoreDataType=FieldDeliveries, DataStoreSce					
nario=Irrig,DataStoreUnits=af,DataStoreMissingValue=null,WriteMode=DeleteAllT		reMissingValue=null,WriteMode=DeleteAllThenI				
	inser of					
				Cancel		
					WriteTimeSeriesToDataSt	

WriteTimeSeriesToDataStore() Command Editor

The command syntax is as follows:

```
WriteTimeSeriesToDataStore(Parameter=Value,...)
```

Parameter	Description	Default	
TSList	Indicates the list of time series to be processed,	AllTS	
	one of:		
	• AllMatchingTSID – all time series that		
	match the TSID (single TSID or TSID with		
	wildcards) will be processed.		
	• AllTS – all time series before the command.		
	• EnsembleID – all time series in the		
	ensemble will be processed.		
	 FirstMatchingTSID – the first time 		
	series that matches the TSID (single TSID or TSID with wildcards) will be processed		
	• LogtMatchingTSID the last time series		
	that matches the TCID (single TSID or TSID)		
	with wildcards) will be processed		
	 SelectedTS - the time series are those 		
	selected with the Select TimeSeries ()		
	command		
TSID	The time series identifier or alias for the time	Required if	
	series to be processed, using the * wildcard	TSList=*TSID.	
	character to match multiple time series.	101100 10121	
EnsembleID	The ensemble to be processed, if processing an	Required if TSList=	
	ensemble.	EnsembleID.	
OutputStart	The date/time for the start of the output.	Use the global output	
OutputEnd	The date/time for the end of the output	Use the global output	
	The date, time for the end of the output.	period.	
DataStore	The name of a database datastore to receive data.	None – must be specified.	
DataStore	The location type to match in the datastore	Location type from	
LocationType		time series is used.	
DataStore	The data source (provider) to match in the	Data source from time	
DataSource	datastore.	series is used.	
DataStore	The data type to match in the datastore.	Data type from time	
DataType		series is used.	
DataStore	The data interval to match in the datastore.	Data interval from time	
Interval		series is used.	
DataStore	The scenario to match in the datastore.	Scenario from time	
DataStore	The value to write to the detectors to indicate a	Series is used.	
MissingValue	missing value in the time series. Specify pull to	the time series will be	
hibbing varae	write null to the database	used (e.g. $N \ge N = 999$)	
DataStoreUnits	Units to use for time series in the database	used (e.g., main, -999).	
	currently not used. Time series data must match		
	the time series as defined in the database.		
WriteMode	The method used to write time series data	InsertUpdate	
	records, recognizing the databases use insert and	-	

Parameter	Description	Default
	update SQL statements. Note that any insert/update actions only occur on exact matches	
	of date/time not on a period. For example	
	DeleteInsert only deletes records that match	
	the specific date/time of a value in the time series.	
	Specify WriteMode as:	
	• DeleteAllThenInsert – delete all the data records for the time series and then insert the time series data records, useful for bulk loading	
	• DeletePeriodThenInsert – delete the data records in the specified output period and then insert the time series data records, useful for bulk loading	
	• DeleteInsert – delete the data first and then insert (all values will need to be matched to delete)	
	• Insert – insert the data with no attempt to update if the insert fails	
	• InsertUpdate – try inserting the data first and if that fails try to update	
	• Update – update the data with no attempt to insert if the update fails	
	• UpdateInsert – try updating the data first and if that fails try to insert	

This page is intentionally blank.

Command Reference: WriteTimeSeriesToJson()

Write time series to a JSON format file

Version 10.21.00, 2013-07-01

This command is under development. The JSON format will change as feedback is received and additional time series information is added to the output (e.g., comments, properties).

The WriteTimeSeriesToJson() command writes time series to a file using JSON (JavaScript Object Notation) format. The file can be included in a JavaScript script to instantiate data objects. The following example illustrates the format of the JSON file, with two hour-interval time series, one without data flags, and one with data flags. The JSON format closely matches time series data management conventions used by TSTool. In the future, support for writing time series data values in parallel (via overlap=true property for the list) may be implemented in order to save space in the file. JSON files can be viewed/edited by online tools such as http://jsoneditoronline.org.

```
"timeSeriesList": {
  "numTimeSeries": 2,
  "overlap": false,
  "timeSeries": [
   ł
    "timeSeriesMeta": {
     "tsid": "MyLoc1..MyDataType.Hour",
     "alias": "MyLoc1",
     "description": "Test data, pattern",
     "locationType": "",
     "locationId": "MyLoc1",
     "dataSource": "",
     "dataType": "MyDataType",
     "scenario": "",
     "missingVal": -999.0,
     "units": "CFS",
     "unitsOriginal": "CFS"
     "start": "1950-01-01 00",
     "end": "1950-01-03 12",
     "startOriginal": "1950-01-01 00",
     "endOriginal": "1950-01-03 12",
     "hasDataFlags": false
    },
    "timeSeriesData": [
     { "dt": "1950-01-01 00", "value": 5.0000 },
      "dt": "1950-01-01 01", "value": 10.0000 },
     .
{ "dt": "1950-01-01 02", "value": 12.0000 },
... omitted ...
      "dt": "1950-01-03 11", "value": 75.0000 },
      "dt": "1950-01-03 12", "value": 5.0000 }
     ł
    ]
   },
   {
    "timeSeriesMeta": {
     "tsid": "MyLoc2..MyData.Hour",
     "alias": "MyLoc2",
     "description": "Test data, pattern",
     "locationType": "",
     "locationId": "MyLoc2",
     "dataSource": "",
     "dataType": "MyData",
     "scenario": "",
     "missingVal": -999.0,
```

WriteTimeSeriesToJson

```
"units": "CFS",
      "unitsOriginal": "CFS",
      "start": "1950-01-01 00",
      "end": "1950-01-04 12",
      "startOriginal": "1950-01-01 00",
      "endOriginal": "1950-01-04 12",
      "hasDataFlags": true
     },
     "timeSeriesData": [
      { "dt": "1950-01-01 00", "value": 7.0000, "flag": "A" },
{ "dt": "1950-01-01 01", "value": 12.0000, "flag": "B" },
{ "dt": "1950-01-01 02", "value": 14.0000, "flag": "" },
...omitted...
      { "dt": "1950-01-04 11", "value": -999.0000, "flag": "D" },
       { "dt": "1950-01-04 12", "value": 77.0000, "flag": "E" }
     ]
    }
  ]
 }
```

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

👌 Edit WriteTimeSeriesToJson() (Command	·	
THIS COMMAND IS UNDER DEVELO Write time series to a JSON format file The JSON file structure dosely matche The working directory is: C:\DevRiv\TS The output filename can be specified u Enter date/times to a precision approp	DPMENT which can be the TSTool in STool_SourceBu using \${Propert priate for outpu	used for webs ternal represen uild\TSTool\test y} notation to t time series.	ite integration. ntation for lists of time series. t\regression\commands\general\WriteTimeSeriesToJson utilize global properties.
TS list:		•	Optional - indicates the time series to process (default=AllTS).
TSID (for TSList=AllMatchingTSID):			
EnsembleID (for TSList=EnsembleID):			
JSON file to write:	Results/Test_WriteTimeSeriesToJson_Hour_out.json Browse		
Output precision:			Optional - digits after decimal (default=4).
Missing value:			Optional - value to write for missing data (default=initial missing value).
Output start:			Optional - override the global output start (default=write all data).
Output end:			Optional - override the global output end (default=write all data).
Command:	WriteTime r_out.jse	eSeriesTo on")	Json(OutputFile="Results/Test_WriteTimeSeriesToJson_Hou
	1	Add Workin	g Directory Cancel OK

WriteTimeSeriesToJson() Command Editor

```
WriteTimeSeriesToJson(Parameter=Value,...)
```

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one	AllTS
	of:	
	 AllMatchingTSID – all time series that 	
	match the TSID (single TSID or TSID with	
	wildcards) will be processed.	
	• AllTS – all time series before the command.	
	• EnsembleID – all time series in the ensemble	
	will be processed.	
	• FirstMatchingTSID – the first time series	
	that matches the TSID (single TSID or TSID	
	with wildcards) will be processed.	
	• LastMatchingTSID – the last time series that	
	matches the TSID (single TSID or TSID with	
	wildcards) will be processed.	
	• SelectedTS – the time series are those	
	selected with the SelectTimeSeries()	
	command.	
TSID	The time series identifier or alias for the time series	Required if
	to be processed, using the * wildcard character to	TSList=*TSID.
	match multiple time series.	D 110 1
Ensempleid	The ensemble to be processed, if processing an	Required if TSList=
		EnsembleID.
OutputFile	The JSON output file. The path to the file can be	None – must be specified.
	absolute or relative to the working directory	
	(command the location). Global properties can be used to specify the fileneme, using the	
	S Property Live menane, using the	
Precision	The number of digits after the decimal for numerical	A (in the future may
1100101011	output.	$\frac{1}{2}$ (in the future may default based on data type)
MissingValue	The value to write to the file to indicate a missing	As initialized when
	value in the time series, must be a number or NaN.	reading the time series or
		creating a new time series,
		typically -999, NaN, or
		another value that is not
		expected in data.
OutputStart	The date/time for the start of the output.	Use the global output
		period.
OutputEnd	The date/time for the end of the output.	Use the global output
		period.

This page is intentionally blank.

Command Reference: WriteTimeSeriesToKml()

Write time series to a KML format file

Version 10.21.00, 2013-07-01

This command is under development. Additional capabilities will be implemented as resources allow and requirements are identified. The WriteTimeSeriesToKml() command writes time series to a file using KML (Keyhole Markup Language) format (see:

https://developers.google.com/kml/documentation). The file can be used with Google Earth, Google Maps, and other spatial viewing tools. This command currently provides very basic capabilities to visualize the locations where time series data are collected, in particular for point data. In the future the command will be updated to have additional features, including:

- Providing the option to output the time series using the timestamp and timespan KML features.
- Providing the option to specify style information with a table, for example using the data type to indicate the symbol and icon.
- Providing additional time series properties for visualization.

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

◆ Edit WriteTimeSeriesToKml() Command		
THIS COMMAND IS UNDER DEVELOPMENT Write time series to a KML format file, which can be used for map integration. Longitude, latitude, and elevation are taken from time series properties. In the future, a table will be used to set map style information. The working directory is: C: \DevRiv\TSTool_SourceBuild\TSTool\test\regression\commands\general\WriteTimeSeriesToKml The output filename can be specified using \${Property} notation to utilize global properties. Enter date/times to a precision appropriate for output time series.		
TS list:		Optional - indicates the time series to process (default=AlITS).
TSID (for TSList=AllMatchingTSID);		4
EnsembleID (for TSList=EnsembleID):		4
KML file to write:	Results/Test_WriteTimeSeriesToKml_Year_out.Kml Browse	
Lontitude property:	Longitude	Required - property containing longitude.
Latitude property:	Latitude	Required - property containing latitude.
Elevation property:	Elevation	Optional - property containing elevation.
Output precision:		Optional - digits after decimal (default=4).
Missing value:		Optional - value to write for missing data (default=initial missing value).
Output start:		Optional - override the global output start (default=write all data).
Output end:	Optional - override the global output end (default=write all data).	
Command:	<pre>WriteTimeSeriesToKml(OutputFile="Results/Test_WriteTimeSeriesToKml_Year_ out.Kml",LongitudeProperty="Longitude",LatitudeProperty="Latitude",Eleva tionProperty="Elevation")</pre>	
Add Working Directory Cancel OK		

WriteTimeSeriesToKml() Command Editor

The command syntax is as follows:

WriteTimeSeriesToKml(Parameter=Value,...)

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of:	AllTS
	• AllMatchingTSID – all time series that match	
	the TSID (single TSID or TSID with wildcards)	
	will be processed.	
	• AllTS – all time series before the command.	
	• EnsembleID – all time series in the ensemble	
	will be processed.	
	• FirstMatchingTSID – the first time series that	
	matches the TSID (single TSID or TSID with	
	wildcards) will be processed.	
	• LastMatchingTSID – the last time series that	
	matches the TSID (single TSID or TSID with	
	wildcards) will be processed.	
	• SelectedTS – the time series are those selected	
	with the SelectTimeSeries() command.	
TSID	The time series identifier or alias for the time series to	Required if
	be processed, using the * wildcard character to match	TSList=*TSID.
TracembleTD	multiple time series.	
Ensempleid	The ensemble to be processed, if processing an	Required if TSList=
0		EnsembleID.
Outputfile	The KML output file. The path to the file can be absolute or relative to the working directory (command	None – must be
	file location). Global properties can be used to specify	specified.
	the filename using the $\$$ {Property} syntax	
Longitude	The name of the time series property containing the	Required
Property	longitude to use for the KML.	requirea
Latitude	The name of the time series property containing the	Required.
Property	latitude to use for the KML.	•
Elevation	The name of the time series property containing the	0
Property	elevation to use for the KML.	
Precision	The number of digits after the decimal for numerical	4 (in the future may
	output. Not currently enabled.	default based on data
		type)
Missing	The value to write to the file to indicate a missing	As initialized when
value	value in the time series, must be a number or NaN.	reading the time series
	Not currently enabled.	or creating a new time
		NaN, or another value
		that is not expected in
		data
OutputStart	The date/time for the start of the output, used with	Use the global output
	KML timestamp. Not currently enabled .	period.
OutputEnd	The date/time for the end of the output, used with	Use the global output
	KML timestamp. Not currently enabled.	period.

Command Reference: WriteWaterML()

Write time series to a WaterML XML format file

Version 10.05.00, 2012-03-06

This command is under development. In particular, an evaluation is determining how best to map internal time series properties to the WaterML specification, including selecting reasonable defaults while allowing override of defaults.

The WriteWaterML() command writes time series to a WaterML XML format file. See the **WaterML Input Type Appendix** for more information about the file format.

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

\delta Edit WriteWaterML() Command 🛛 🛛 🔀			
This command is under development. Functionality currently is very limited. Write time series to a WaterML format file, which can be specified using a full or relative path (relative to the working directory). The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ReadWaterML The output filename can be specified using \${Property} notation to utilize global properties. Enter date/times to a precision appropriate for output time series.			
TS list:	Optional - indicates the time series to process (default=AllTS).		
TSID (for TSList=AllMatchingTSID):		~	
EnsembleID (for TSList=EnsembleID):		~	
WaterML file to write:		Browse	
WaterML version:	Optional - WaterML version (default=most recent supported).		
Output precision:	Optional - digits after decimal (default=4).		
Missing value:	Optional - value to write for missing data (default=initial missing value).		
Output start:	Optional - override the global output start (default=write all data).		
Output end:	Optional - override the global output end (default=write all data).		
Command:	WriteWaterML()		
Add Working Directory Cancel OK			

WriteDateValue() Command Editor

The command syntax is as follows:

WriteWaterML(Parameter=Value,...)

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one	AllTS
	of:	
	• AllMatchingTSID – all time series that	
	match the TSID (single TSID or TSID with	
	wildcards) will be processed.	
	• AllTS – all time series before the command.	

	 EnsembleID - all time series in the ensemble will be processed. FirstMatchingTSID - the first time series that matches the TSID (single TSID or TSID with wildcards) will be processed. LastMatchingTSID - the last time series that matches the TSID (single TSID or TSID with wildcards) will be processed. SelectedTS - the time series are those selected with the SelectTimeSeries() 	
תופית	command.	Paguirad if
	to be processed, using the * wildcard character to match multiple time series.	TSList=*TSID.
EnsembleID	The ensemble to be processed, if processing an	Required if TSList=
	ensemble.	EnsembleID.
OutputFile	TheWaterML output file. The path to the file can be absolute or relative to the working directory (command file location). Global properties can be used to specify the filename, using the \${Property} syntax.	None – must be specified.
Version	The WaterML version to write. Currenly only version 1.1 is supported.	1.1
Precision	The number of digits after the decimal for numerical output.	4 (in the future may default based on data type)
MissingValue	The value to write to the file to indicate a missing value in the time series.	As initialized when reading the time series or creating a new time series, typically -999, NaN, or another value that is not expected in data.
OutputStart	The date/time for the start of the output.	Use the global output period.
OutputEnd	The date/time for the end of the output.	Use the global output period.

Documentation Binder Spine Labels

This page, when printed, can be used for a spine in a binder.

Colorado's Decision Support Systems (CDSS) TSTool – Time Series Tool – Command Reference