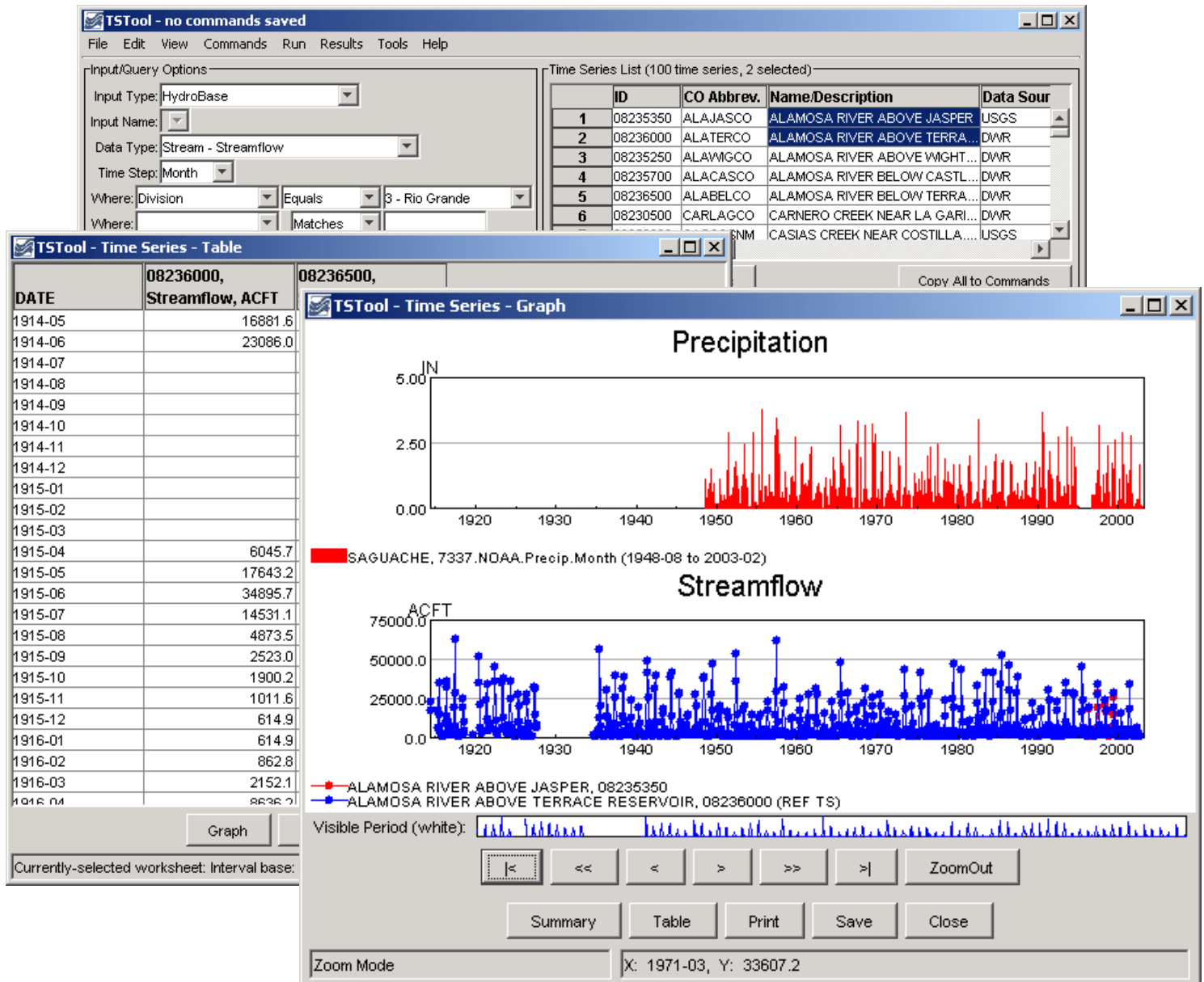


TSTool - Time Series Tool



Colorado Department of Natural Resources
Colorado Water Conservation Board
Division of Water Resources

Developed by:

 **Riverside Technology, inc.**

Version 9.09.00, 2010-09-30

This page is intentionally blank.

This document is formatted for double-sided printing.

Table of Contents

01_Cover_CDSS.pdf	1
DISCLAIMER for CDSS Products	17
1 Acknowledgements	19
2 Introduction	21
2.1 Time Series Objects and Identifiers	22
2.2 Date/Time Conventions	25
2.3 Time Scale for Time Series Data	26
2.4 Time Series Commands and Processing Sequence	26
2.5 Using Time Series Aliases	29
2.6 Time Series Ensembles	30
3 Getting Started	33
4 Commands	55
4.1 Create Time Series	57
4.2 Converting Time Series Identifier to Read Command	58
4.3 Read Time Series	59
4.4 Fill Time Series Data	60
4.5 Set Time Series Data	63

Table of Contents

4.6 Manipulate Time Series	63
4.7 Analyze Time Series	64
4.8 Models	64
4.9 Output Time Series	65
4.10 Commands for Specific Input Types	65
4.11 Commands for Ensemble Processing	66
4.12 Commands for Table Processing	66
4.13 General Commands – Comments	66
4.14 General Commands – File Handling	68
4.15 General Commands – Logging	68
4.16 General Commands – Running Commands and External Software	68
4.17 General Commands – Test Processing	69
5 Tools	71
6 Examples of Use	81
7 Using the Map	99
8 Troubleshooting	109
8.1 Obsolete Commands	113

Table of Contents

9 Quality Control	119
9.1 Quality Control for TSTool Software	119
9.2 Using TSTool to Quality Control Data	127
Command Glossary	129
Command Reference: #	139
Command Reference: /*	141
Command Reference: */	143
Command Reference: Time Series Identifier (TSID)	145
Command Reference: Add()	147
Command Reference: AddConstant()	149
Command Reference: AdjustExtremes()	151
Command Reference: AnalyzePattern()	153
Command Reference: ARMA()	157
Command Reference: Blend()	163
Command Reference: CalculateTimeSeriesStatistic()	165
Command Reference: TS Alias = ChangeInterval()	171
Irregular Time Series to Regular Time Series	171

Table of Contents

Regular Time Series to Regular Time Series	173
ACCM (Accumulation) to ACCM (Accumulation)	173
ACCM (Accumulation) to INST (Instantaneous)	174
ACCM (Accumulation) to MEAN	174
INST (Instantaneous) to INST (Instantaneous)	175
INST (Instantaneous) to ACCM (Accumulation)	176
INST (Instantaneous) to MEAN	176
MEAN to MEAN	178
MEAN to ACCM (Accumulation)	178
MEAN to INST (Instantaneous)	178
Command Reference: ChangePeriod()	185
Command Reference: CheckTimeSeries()	187
Command Reference: CompareFiles()	191
Command Reference: CompareTimeSeries()	193
Command Reference: ComputeErrorTimeSeries()	197
Command Reference: ConvertDataUnits()	199
Command Reference: TS Alias = Copy()	201

Table of Contents

Command Reference: CopyEnsemble()	203
Command Reference: CopyTable()	205
Command Reference: CreateEnsembleFromOneTimeSeries()	207
Command Reference: CreateFromList()	209
Command Reference: CreateRegressionTestCommandFile()	213
Command Reference: Cumulate()	217
Command Reference: Delta()	219
Command Reference: DeselectTimeSeries()	223
Command Reference: TS Alias = Disaggregate()	225
Command Reference: Divide()	229
Command Reference: Exit()	231
Command Reference: ExpandTemplateFile()	233
Command Reference: FillConstant()	235
Command Reference: FillDayTSFrom2MonthTSAnd1DayTS()	237
Command Reference: FillFromTS()	241
Command Reference: FillHistMonthAverage()	243
Command Reference: FillHistYearAverage()	245

Table of Contents

Command Reference: FillInterpolate()	247
Command Reference: FillMixedStation()	249
Best Fit Indicators	250
Mixed Station Analysis Tool	251
Command Editing	252
Command Reference: fillMOVE1()	257
Command Reference: FillMOVE2()	259
Command Reference: FillPattern()	263
60_Command_FillPrincipalComponentAnalysis.pdf	265
Command Reference: FillProrate()	267
Command Reference: FillRegression()	271
Command Reference: FillRepeat()	275
Command Reference: FillUsingDiversionComments()	277
Diversion Comment Not Used Flag	277
Structure Currently in Use Flag	277
Command Reference: Free()	283
Command Reference: FTPGet()	285

Table of Contents

Command Reference: InsertTimeSeriesIntoEnsemble ()	287
Command Reference: LagK()	289
Command Reference: ManipulateTableString()	293
Command Reference: Multiply()	295
Command Reference: TS Alias = NewDayTSFromMonthAndDayTS()	297
Command Reference: TS Alias = NewEndOfMonthTSFromDayTS()	301
Command Reference: NewEnsemble ()	305
Command Reference: TS Alias = NewPatternTimeSeries()	307
Examples	311
Examples	315
Command Reference: TS Alias = NewStatisticYearTS()	317
Example	320
Command Reference: NewTable ()	323
Command Reference: TS Alias = NewTimeSeries()	325
Command Reference: NewTreeView()	327
Command Reference: TS Alias = Normalize()	329
Command Reference: OpenCheckFile()	331

Table of Contents

Command Reference: OpenHydroBase()	333
Command Reference: ProcessTSProduct()	337
Command Reference: ReadDateValue()	341
Command Reference: TS Alias = ReadDateValue()	343
Command Reference: ReadDelimitedFile()	345
Command Reference: ReadHecDss()	351
Command Reference: ReadHydroBase()	353
Command Reference: TS Alias = ReadHydroBase()	357
Command Reference: ReadMODSIM()	361
Command Reference: TS Alias = ReadMODSIM()	363
Command Reference: ReadPatternFile()	365
Command Reference: TS Alias = ReadRiverWare()	367
Command Reference: ReadStateCU()	369
Command Reference: ReadStateCUB()	371
Command Reference: ReadStateMod()	373
Command Reference: ReadStateModB()	375
Command Reference: ReadTableFromDBF()	377

Table of Contents

Command Reference: ReadTableFromDelimitedFile()	379
Command Reference: TS Alias = ReadTimeSeries()	381
Command Reference: TS Alias = ReadUsgsNwis()	383
Command Reference: TS Alias = RelativeDiff()	385
Command Reference: RemoveFile()	389
Command Reference: ReplaceValue()	391
Command Reference: ResequenceTimeSeriesData()	393
Command Reference: RunCommands()	397
Command Reference: RunningAverage()	399
Command Reference: RunDSSUTL()	403
Command Reference: RunProgram()	407
Command Reference: RunPython()	411
Command Reference: Scale()	415
Command Reference: SelectTimeSeries()	417
Command Reference: SetAutoExtendPeriod()	419
Command Reference: SetAveragePeriod()	421
Command Reference: SetConstant()	423

Table of Contents

Command Reference: SetDataValue()	425
Command Reference: SetDebugLevel()	427
Command Reference: SetFromTS()	429
Command Reference: SetIgnoreLEZero()	433
Command Reference: SetIncludeMissingTS()	435
Command Reference: SetInputPeriod()	437
Command Reference: SetOutputPeriod()	439
Command Reference: SetOutputYearType()	441
Command Reference: SetPatternFile()	443
Command Reference: SetProperty()	445
Command Reference: SetTimeSeriesPropertiesFromTable()	447
Command Reference: SetTimeSeriesProperty()	449
Command Reference: SetToMax()	451
Command Reference: SetToMin()	453
Command Reference: SetWarningLevel()	455
Command Reference: SetWorkingDir()	457
Command Reference: ShiftTimeByInterval()	459

Table of Contents

Command Reference: SortTimeSeries()	461
Command Reference: StartLog()	463
Command Reference: StartRegressionTestResultsReport()	465
Command Reference: StateModMax()	467
Command Reference: Subtract()	469
Command Reference: TableMath()	471
Command Reference: TableTimeSeriesMath()	473
Command Reference: TimeSeriesToTable()	475
Command Reference: VariableLagK()	479
Command Reference: WebGet()	485
Command Reference: TS Alias = WeightTraces()	487
Command Reference: WriteCheckFile()	491
Command Reference: WriteDateValue()	493
Command Reference: WriteHecDss()	495
Command Reference: WriteProperty()	499
Command Reference: WriteRiverWare()	501
Command Reference: WriteStateCU()	503

Table of Contents

Command Reference: WriteStateMod()	505
Command Reference: WriteSummary()	507
Command Reference: WriteTableToDelimitedFile()	509
Command Reference: WriteTimeSeriesProperty()	511
Appendix: TSTool Installation and Configuration for CDSS	513
1. Overview	513
2. File Locations	513
3. Installing TSTool	514
4. Uninstalling TSTool Software	523
5. Running TSTool	524
6. TSTool Configuration	525
1. TSTool Version History	531
Appendix: ColoradoIPP Input Type	557
Overview	557
ColoradoIPP and Standard Time Series Properties	557
Limitations	557
Appendix: Colorado Satellite Monitoring System (SMS) Input Type	559

Table of Contents

Overview	559
Appendix: Colorado Water HydroBase Guest (ColoradoWaterHBGuest) Input Type	561
Overview	561
ColoradoWaterHBGuest Web Service and Standard Time Series Properties	561
Limitations	561
Appendix: Colorado Water Satellite Monitoring System (ColoradoWaterSMS) Input Type	563
Overview	563
ColoradoWaterSMS Web Services and Standard Time Series Properties	563
Limitations	564
Appendix: DateValue Input Type	565
Appendix: HEC-DSS Input Type	569
Overview	569
HEC-DSS Files and Standard Time Series Properties	569
Limitations	570
Appendix: HydroBase Input Type	571
Appendix: RiverWare Input Type	585
Appendix: StateCU Input Type	587

Table of Contents

Appendix: StateCUB Input Type_____	591
Appendix: StateMod Input Type_____	595
Appendix: StateModB Input Type_____	599
Appendix: USGSNWIS Input Type_____	603
Appendix: TSView - Time Series Viewing Tools_____	605
Overview_____	606
Time Series Terminology_____	606
Time Series Properties Interface_____	609
Time Series Traces_____	616
Time Series Views_____	617
Time Series Product Reference_____	647
Appendix: GeoView Mapping Tools_____	665
Appendix: Spatial Data Format – ESRI Shapefile_____	687
Documentation Binder Spine Labels_____	689

DISCLAIMER for CDSS Products

2002-02-16

CDSS products include data and software from State of Colorado sources and from external sources like the U. S. Geological Survey (USGS). The following disclaimer applies to CDSS products:

CDSS products and associated access are under development at this time. Access is provided solely to test and demonstrate CDSS capabilities. In the future, this access may be restricted or offered for a fee. The State assumes no legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed herein. It is the user's responsibility to determine the fitness of the data for a particular purpose.

This page is intentionally blank

1 Acknowledgements

Version 06.08.02, 2004-08-05

TSTool has been developed by Riverside Technology, inc. (RTi). Early development work was funded by the State of Colorado, Water Conservation Board, under the Colorado River Decision Support System (CRDSS). Later development occurred as part of CRDSS maintenance, the State of Colorado's Rio Grande Decision Support System (RGDSS) and the State of Colorado's South Platte Decision Support System (SPDSS), under the umbrella CDSS effort (Colorado's Decision Support Systems).

Enhancements to the CDSS version have also been made by RTi in order to support a wider variety of uses and data formats. Enhancements will continue to be made by RTi while offering backward compatibility as much as possible.

Users are encouraged to provide general feedback to RTi using the email address: support@riverside.com and to provide feedback specific to CDSS functionality (e.g., HydroBase, StateMod, StateModB, and StateCU input types) using the email address: cdss@state.co.us. RTi's web site is <http://www.riverside.com>. The CDSS web site is <http://cdss.state.co.us>.

The MOVE.1 and MOVE.2 procedures description was taken from:

Hirsch, R. M., 1982, A Comparison of Four Streamflow Record Extension Techniques: Water Resources Research, Vol. 18, No. 4, pages 1081-1088.

Additional information can be found in:

Guidelines for Determining Flood Flow Frequency, Bulletin 17B, USGS.

This page is intentionally blank.

2 Introduction

Version 09.08.00, 2010-09-08

TSTool can be thought of as a time series calculator. TSTool displays, manipulates, and analyzes time series data, either interactively or in batch (automated) mode. The TSTool GUI (Graphical User Interface) provides access to viewing and analysis features, command editors, and provides error feedback. Time series can be read from and written to a variety of file and database formats. Although a graphical user interface is provided, the heart of TSTool's analytical features is a command workflow processor. Depending on the task being performed, the command language may be used extensively or not at all. This flexibility makes TSTool useful for basic data viewing and advanced analysis. The documentation is divided into the following main sections.

Chapter 2 – Introduction provides background information on time series concepts and how TSTool processes time series.

Chapter 3 – Getting Started provides an overview of TSTool interface features.

Chapter 4 – Commands provides a summary of time series processing commands.

Chapter 5 – Tools provides information about analysis tools.

Chapter 6 – Examples of Use provides examples of how TSTool is commonly used.

Chapter 7 – Using the Map provides information about using the map interface to link time series with spatial data.

Chapter 8 – Troubleshooting provides troubleshooting information, including a list of obsolete commands.

Chapter 9 – Quality Control provides guidelines and examples for using TSTool to quality control data processing.

The **Command Reference** provides a complete command reference, with commands listed in alphabetical order. Because some commands are used in more than one situation, this allows the commands to be fully documented once, and referred to as needed.

The **Installation and Configuration** appendix provides information about installing and configuring TSTool.

The **Release Notes** appendix summarizes TSTool changes over time.

Several appendices provide information about supported input types (appendices are inserted as additional input types are added).

The **TSView Time Series Viewing Tools** appendix provides a general reference for time series viewing features. These features are used throughout TSTool and other software developed by RTi.

The **GeoView Mapping Tools** appendix provides a general reference for the GeoView map interface. The mapping interface is being phased in and is used by other software developed by RTi.

This documentation can be printed double-sided and is best viewed as PDF to use the navigable table of contents and bookmarks.

2.1 Time Series Objects and Identifiers

TSTool considers time series as objects that are queried, manipulated, viewed, and output. A time series is defined as a series of date/time versus data pairs. Data generally consist of floating point values; however, time series may contain other data (e.g., data quality flags). TSTool treats time series as either regular interval (equal spacing of date/time) or irregular interval (e.g., infrequent measurements, sometimes referred to as *observations*). Regular time series lend themselves to simpler storage and faster processing because date/time information only needs to be stored for the endpoints and processing is less complicated.

TSTool defines each time series as having an interval base and multiplier (e.g., 1Month, 24Hour). In many cases, the multiplier is 1 and is not shown in output (e.g., Month rather than 1Month is shown). In addition to a period of record, interval, and data values, time series have attributes, or metadata, that include:

- Units (e.g., CFS)
- Data type (e.g., Streamflow)
- Data limits (the maximum, minimum, etc.)
- Description (generally a station or structure name)
- Missing data value (used internally to mark missing data and trigger data filling, often -999 or in some cases NaN [Not a Number])
- Comments (often station comments, if available)
- Genesis history (a list of comments about how the time series was created and manipulated)

To manage time series, TSTool associates each time series with an identifier that uses the notation:

`Location.Source.Type.Interval.Scenario[Seq]~InputType~InputName`

`Location.Source.Type.Interval.Scenario[Seq]~DataStoreName`

The first five parts (`Location.Source.Type.Interval.Scenario`) are used to identify time series, with further explanation below:

- `Location` – typically a physical location identifier, such as a station, basin, or sensor identifier.
- `Source` – a data provider identifier, usually a government or system identifier (e.g., USGS, NWS), necessary because sometimes the provider for data changes for a location.
- `Type` – the data type, typically specific to the data (e.g., Streamflow, Precip) – TSTool does not try to institute global data type definitions).
- `Interval` – the data interval, indicating the time between data values (e.g., 6Hour, Day, Irregular).
- `Scenario` – an optional item that indicates a scenario (e.g., Hist, Filled, Max, Critical).
- `Seq` – an optional item used in cases where multiple time series traces may be available, with all other identifier information being equal (e.g., for simulations where multiple versions of input are used or for cases when a historical time series is cut into annual traces, collectively known as

ensembles). Typically the sequence number is a four-digit year corresponding to the data input year.

The last part (~InputType~InputName or DataStoreName) is used indicate input information, which allows TSTool locate and read the time series from a file or database. The input information was introduced starting with TSTool version 5.04.00. The data store convention was introduced in TSTool version 9.08.00 and allows any name to be used to define a data store – the details of the configuration are defined in a properties file. This allows more flexibility in defining data connections. The data store convention will be phased in as the software is enhanced.

A summary of input types that are currently supported or under development is listed in the following table (see the input type appendices for more information about how time series identifiers are formatted for specific input types). Features for these input types may or may not be available, depending on the TSTool configuration (see the **TSTool Installation and Configuration Appendix**). The main constraint on whether an input type is considered for implementation in TSTool is whether the input type is a standard that is used in a relatively wide audience or for key applications. By supporting general formats, TSTool can support the largest group of users and provide the most useful general features.

Input Types for TSTool

Input Type	Description
ColoradoBNDSS	State of Colorado Basin Needs Decision Support System (BNDSS) database (under development).
ColoradoWaterHBGuest	Web service for State of Colorado's historical data.
ColoradoWaterSMS	Web service for State of Colorado's real-time data.
DateValue	General delimited date/value file with extended header information, able to store one or more time series.
Delimited	Generic column-delimited format (see the <code>ReadDelimitedFile()</code> command).
DIADvisor	DIADvisor real-time environmental monitoring software, from OneRain, Inc.
HEC-DSS	Army Corp of Engineers binary time series database file used with Hydrologic Engineering Center (HEC) software.
HydroBase	State of Colorado database.
MexicoCSMN	Hydrometeorological database for Mexico Coordinación Servicio Meteorológico Nacional (CSMN, similar to US National Weather Service).
MODSIM	Colorado State University MODSIM model, version 7.
NWSCard	National Weather Service River Forecast System (NWSRFS) card file format for hourly data.
NWSRFS_ESPTTraceEnsemble	NWSRFS Ensemble Streamflow Prediction binary files.
NWSRFS_FS5Files	NWSRFS binary FS5Files preprocessor and processed database.
RiversideDB	Riverside Technology, inc. database used for real-time and historical time series data (e.g., use with RiverTrak [®] System software).
RiverWare	University of Colorado Center for Advanced Decision Support for Water and Environmental Systems (CADSWES) RiverWare model data format.
SHEF	Standard Hydrologic Exchange Format, a common data format used by United States government agencies.

Input Type	Description
StateCU	State of Colorado consumptive use model time series and report formats.
StateCUB	State of Colorado consumptive use model binary output file.
StateMod	State of Colorado StateMod model time series file format.
StateModB	State of Colorado StateMod model output binary file.
USGS NWIS	United States Geological Survey National Water Information System format

An example of a time series identifier for a monthly streamflow time series in HydroBase is:

```
09010500.USGS.Streamflow.Month~HydroBase
```

The same time series for a USGS NWIS input source might be identified using:

```
09010500.USGS.Streamflow.Month~USGSNWIS~C:\temp\09010500.txt
```

In this example, the optional scenario (fifth part) and sequence number are not used. This identifier string can be saved in a command file or time series product description file, which can be processed again later. The identifier string allows TSTool to determine how to re-query the time series. The time series identifier is useful for managing time series. The TSTool GUI typically handles creation of all time series identifiers; however, identifiers can be created with an editor once the format is familiar. The path to files can be absolute or relative to the command file. The latter is recommended to improve portability of files between computers.

Because time series identifiers are somewhat cumbersome to work with, TSTool allows a time series *alias* to be used instead. For example, the following command illustrates how a HydroBase time series can be read and associated with an alias:

```
TS X = ReadTimeSeries("09010500.USGS.Streamflow.Month~HydroBase")
```

This allows the time series to be referred to as X during further processing (e.g., when manipulated with commands). Whether full identifiers or aliases are used, the overall identifier must be unique during processing to guarantee that time series commands are processed as desired (duplicate aliases and identifiers can be present but the first one found will be used - see **Section 2.4 Time Series Commands and Processing Sequence** for an example). TSTool ignores upper/lower case when comparing identifiers, aliases, and other commands, although it is good practice to be consistent.

When editing commands, TSTool does not normally show the input type and input name parts of the identifier because this information is most appropriate for read commands. There are cases where two time series identifiers will be the same except for the input type and name. In these cases, an alias should be assigned when reading the time series and the alias used in later commands. If for some reason an alias cannot be used, the input type and name may need to be manually added if the command editors do not display by default (e.g., in cases where the identifiers cannot be determined without actually running a command).

Newer commands also allow the alias to be assigned even when reading multiple time series. This convention uses an `Alias` parameter rather than the `TS Alias =` syntax and may be phased in as the preferred syntax. For example, the alias may be assigned to the location identifier only.

2.2 Date/Time Conventions

TSTool uses date/time information in several ways:

1. Data values in time series are associated with a date/time and the precision of all date/time information should be consistent within the time series, as discussed below,
2. The data interval indicates the time spacing between data points and is represented as a multiplier (optional if 1) and a base (e.g., Day, 24Hour),
3. The period of a time series is defined by start and end date/time values, using an appropriate precision,
4. An analysis period may be used to indicate when data processing should occur,
5. Output is typically formatted for calendar year (January to December), water year (October to November), or irrigation year (November to October) – calendar year is the default but can be changed in some commands and output.

A date/time has a precision. For example, 2002-02 has a monthly precision and 2002-02-01 has a daily precision. Each date/time object knows its precision and “extra” date/time information is set to reasonable defaults (e.g., hour, minute, and second for a monthly precision date/time are set to zero and the day is set to 1). The date/time precision is important because TSTool uses the date/time objects to iterate through data, to compare dates, and to calculate a plotting position for graphs. Specifying date/time information with incorrect precision may cause inconsistent behavior.

The TSTool documentation and user interface typically use ISO 8601 International Standard formats for date/time information. For example, dates are represented using YYYY-MM-DD and times are represented using hh:mm:ss. A combined date/time is represented as YYYY-MM-DD hh:mm:ss. In order to support common use, TSTool also attempts to handle date/time information that uses United States and other date formats. In such cases, the length of the date/time string and the position of special characters are used to make a reasonable estimate of the format. Using ambiguous formats (e.g., two-digit years that may be confused with months) may cause errors in processing. Adhering to the ISO 8601, standard formats will result in the fewest number of errors. The appendices for various input types discuss issues with date/time formats.

Plotting positions are computed by converting dates to floating point values, where the whole number is the year, and the fraction is the fractional part of the year, considering the precision. The floating-point date is then interpolated to the screen pixels, as integers. In most cases, the high-precision date/time parts are irrelevant because they default to zero. However, in some cases the precision can impact plots significantly. For example, when plotting daily and monthly data on the same graph, the monthly data will be plotted ignoring the day whereas the daily values correspond days 1 to 31. The ability to plot monthly data mid-month or end-of-month has not been implemented. The **TSView Time Series Viewing Tools Appendix** provides examples of plots.

The date/time precision is very important when performing an analysis or converting between time series file formats. For example, a file may contain 6Hour data using a maximum hour of 24 (e.g., 6, 12, 18, 24). When reading this data, TSTool will convert the hour 24 values to hour 0 of the next day. Consequently, the hour and day of the original data will seemingly be shifted, even though the data are actually consistent. This shift may also be perceived when converting from hourly data to daily data because the hour can have a value of 0 to 23, whereas days in the month start with 1. The perceived shift is purely an artifact of time values having a minimum value of zero.

2.3 Time Scale for Time Series Data

The time scale for time series data gives an indication of how the data value were measured or computed. The time scale is generally determined from the data type (or the data type and interval) and can be one of the following (the abbreviations are often used in software choices):

- **Instantaneous (INST):** The data value represents the data observed at the time associated with the reading (e.g., instantaneous temperature, streamflow, or snow depth). Instantaneous data may be of irregular or regular interval, depending on the data collection system. If irregular, the precision of the date/time associated with the reading may vary (e.g., automated collection systems may have very precise times whereas infrequently recorded field measurements may only be recorded to the nearest day).
- **Accumulated (ACCM):** The data value represents the accumulation of the observed data over the preceding interval. The date/time associated with the data value corresponds to the end of the interval. For example, precipitation (rain or snow recorded as melt) is often recorded as an accumulation over some interval. Accumulated values are typically available as a regular time series, although this is not a requirement (e.g., precipitation might be accumulated between times that a rain gage is read and emptied).
- **Mean (MEAN):** The data value represents the mean value of observations during the preceding interval. The date/time associated with the data value corresponds to the end of the interval. The mean includes values after the previous timestamp and including the current timestamp. The computation of mean values may be different depending on whether the original data are irregular or regular. For example, if the original data are regular interval, then equal weight may be given to each value when computing the mean (a simple mean). If the original data are irregular interval, then the weight given to each irregular value may depend on the amount of time that a value was observed (a time-weighted mean, not a simple mean).

Without having specific information about the time scale for data, TSTool assumes that all data are instantaneous for displays. If time series are graphed using bars, an option is given to display the bar to the left, centered on, or to the right of the date/time. If time series are graphed using lines or points, the data values are drawn at the date/time corresponding to data values. This may not be appropriate for the time scale of the data. In most cases, this default is adequate for displays. Graphing data of different time scales together does result in visual inconsistencies. These issues are being evaluated and options may be implemented in future releases of the software. In particular, an effort to automatically determine the time scale from the data type and interval is being evaluated. This can be difficult given that data types are not consistent between input types and time scale may be difficult to determine when reading time series. Refer to the input type appendices for information about time scale.

The time scale is particularly important when changing the time interval of data. For example, conversion of instantaneous data to mean involves an averaging process. Conversion of instantaneous data to accumulated data involves summing the original data. Commands that change interval either operate only on data of a certain time scale or require that the time scale be specified to control the conversion. Refer to the command documentation for specific requirements.

2.4 Time Series Commands and Processing Sequence

Although TSTool can be run in batch mode (see **Chapter 3 – Getting Started**), you should be able to perform all time series viewing and manipulation within the GUI. Commands are used to read, manipulate, and output time series. Commands are processed sequentially from the first to the last

commands using the steps described below. This section describes in detail the processing sequence. See the examples in **Chapter 6 – Examples of Use** for illustrations of the processing sequence.

Note that older versions of TSTool (before version 5.xx.xx) did not allow multi-step manipulation and therefore time series were read and manipulated in one step. This convention had limitations and has been changed to allow multi-step operations on time series, allowing more options for filling and manipulation. Old command files are supported as much as possible but some updates to old command files may be required.

TSTool commands fall into three main categories:

1. Time series identifiers (see Section **2.1 – Time Series Objects and Identifiers**), which are equivalent to time series “read” commands (where the identifier input type is used to determine which read command to use),
2. General commands, which are used to set properties like the period for output, and,
3. Time series commands, which are used to read and manipulate time series and output results.

Commands are processed sequentially and can be repeated as necessary. A typical user starts learning TSTool by performing simple queries and displaying results while gradually utilizing more commands. The current software uses command syntax as follows:

```
Command (Param1=Value1, Param2="Value" , ...)
```

Values that may contain spaces or commas are normally surrounded by double quotes. This notation is useful for the following reasons:

- The parameter names are included in the command, in order to make the command more readable.
- Because the parameter name is included, the parameters can generally be in any order. The command editor dialogs will enforce a default order.
- Parameters that have default values can be omitted from the parameter list, shortening commands.
- New parameters can be added over time, without requiring parameter order to change in existing commands.

The above notation is being used for new commands and older commands are being updated to the new syntax as new software releases are made. Command editor dialogs will update old commands to the new syntax and the processing code will recognize old and new command syntax. The **Command Reference** illustrates the current command syntax.

The following sequence occurs when processing commands:

1. **Parse the command.** A time series identifier or command is parsed to determine how to execute the command. Example commands are shown below. If the command is a general command, the action is taken and a new command is read in step 1 (general commands can be specified multiple times to change properties throughout a run). If the command results in reading or creating a time series, steps 2 - 4 are executed, as described below. If a command is a time series manipulation command, step 4 is executed.

```
# Example commands
08235350.USGS.Streamflow.Month~HydroBase
08236000.USGS.Streamflow.Month~HydroBase
Add(TSID="08235350.USGS.Streamflow.Month",HandleMissingHow=IgnoreMissing,
```

```

TSList=SpecifiedTSID,AddTSID="08236000.DWR.Streamflow.Month")
08235350.USGS.Streamflow.Month~HydroBase

```

2. **Read Time Series.** TSTool recognizes that certain commands should read a new time series and will perform the appropriate action. For example, in the above example, the time series identifier 08235350.USGS.Streamflow.Month~HydroBase indicates that the corresponding time series should be read from a HydroBase database. The input type in the identifier (information after the ~) is used to determine how to read the time series. Unless the SetInputPeriod() command has been used, **the entire time series period is read in this step** because data filling steps may require the full period (e.g., to determine regression relationships or long-term monthly average).

Commands that do not cause a time series to be read (but instead to be manipulated) are described in step 4.

If the input type, and if needed, input name, are specified in the identifier, they are only used in the initial read. Additional manipulation commands only use the first five parts of the identifier or the time series alias to identify the time series. If the same time series needs to be read from two input types (e.g., to compare whether a time series was properly loaded into a database from a file), use a different time series alias for each time series to uniquely identify each time series. This may require using a specific variant of a read command that assigns an alias.

At the end of this step, a new time series will exist in TSTool's memory.

3. **Compute Data Limits.** The time series data limits are computed because they may be needed later for filling. This information includes the long-term monthly averages. These limits are referred to as the original data limits.
4. **Access and Manipulate Time Series.** Commands that manipulate time series (fill, add, etc.) do not automatically read the time series or make another copy. Instead, time series that are in memory are located and manipulated. The following example illustrates how the time series identified by 08235350.USGS.Streamflow.Month has its data values modified by adding the data from the time series identified by 08236000.USGS.Streamflow.Month.

```

# Example commands
08235350.USGS.Streamflow.Month~HydroBase
08236000.USGS.Streamflow.Month~HydroBase
Add(TSID="08235350.USGS.Streamflow.Month",HandleMissingHow=IgnoreMissing,TSList
=SpecifiedTSID,AddTSID="08236000.DWR.Streamflow.Month")
08235350.USGS.Streamflow.Month~HydroBase

```

To locate a time series so that it can be modified, TSTool first checks the alias of known time series (those that have been defined in previous commands) against the current time series of interest (TSID="08235350.USGS.Streamflow.Month"), assuming that this string is an alias. If the alias is not found, it checks the full identifier of known time series against the current time series of interest. In this example, time series 08235350.USGS.Streamflow.Month was read in the first step and is therefore found as a match for the identifier. Similarly, the second time series in the command (08236000.USGS.Streamflow.Month) is found and is used to process the command, resulting in a modification of the first time series. **Sequential manipulations of the same time series can occur (e.g., fill by one method, then fill by another).**

To locate time series in memory, TSTool generally looks through the list of time series, searching backwards from the current command being processed. Alternatively, the `TSList` parameter for the command will be used if available. It is possible to use the same identifier more than once in a command file while allowing localized processing of each time series; however, this may lead to confusion and should be avoided. In the above example, the time series identified by `08235350.USGS.Streamflow.Month~HydroBase` is read twice, once to be acted on by the `Add()` command, and once with no manipulation (e.g., to compare the "before" and "after").

During processing, extra time series can accumulate and will be available for output. Use the `Free()` command to free time series that are no longer needed. This removes the time series from memory. See also the `DeselectTimeSeries()` and `SelectTimeSeries()` commands. Output commands also may use the `TSList` parameter to indicate which time series are to be output.

5. After processing the time series, a list of available time series that are in memory are listed in the GUI. One or more of these time series can be selected and viewed using the **Results** menu or analyzed using the **Tools** menu (also right click on time series listed in the results menu at the bottom of the main window). Time series can also be saved in some of recognized input type formats using the **File...Save...Time Series As** menus.

If running in batch mode using the `-commands` option, all of the above steps occur in sequence and the GUI interfaces are not displayed. Old command files should be updated to reflect the new processing sequence. Processing the example shown above results in three time series in memory:

1. A time series identified by `08235350.USGS.Streamflow.Month`, containing the sum of the two time series.
2. A time series identified by `08236000.USGS.Streamflow.Month`, containing the input to the `Add()` command.
3. A time series **also** identified by `08235350.USGS.Streamflow.Month`, containing the original data from the time series that is added to. This contains the original data because a time series identifier by itself in a command list will cause the time series to be read.

These time series can be graphed or saved in an output file.

2.5 Using Time Series Aliases

The previous sections discussed time series identifiers and processing time series. The concept of a time series alias was described as a "shortcut" when identifying a time series. Aliases are useful when creating more complicated lists of commands, where using full time series identifiers become cumbersome. Aliases are typically assigned when creating new time series using the following command syntax:

```
TS Alias = SomeCommandThatCreatesATimeSeries()  
  
SomeCommandThatCreatesTimeSeries(Alias="some pattern")
```

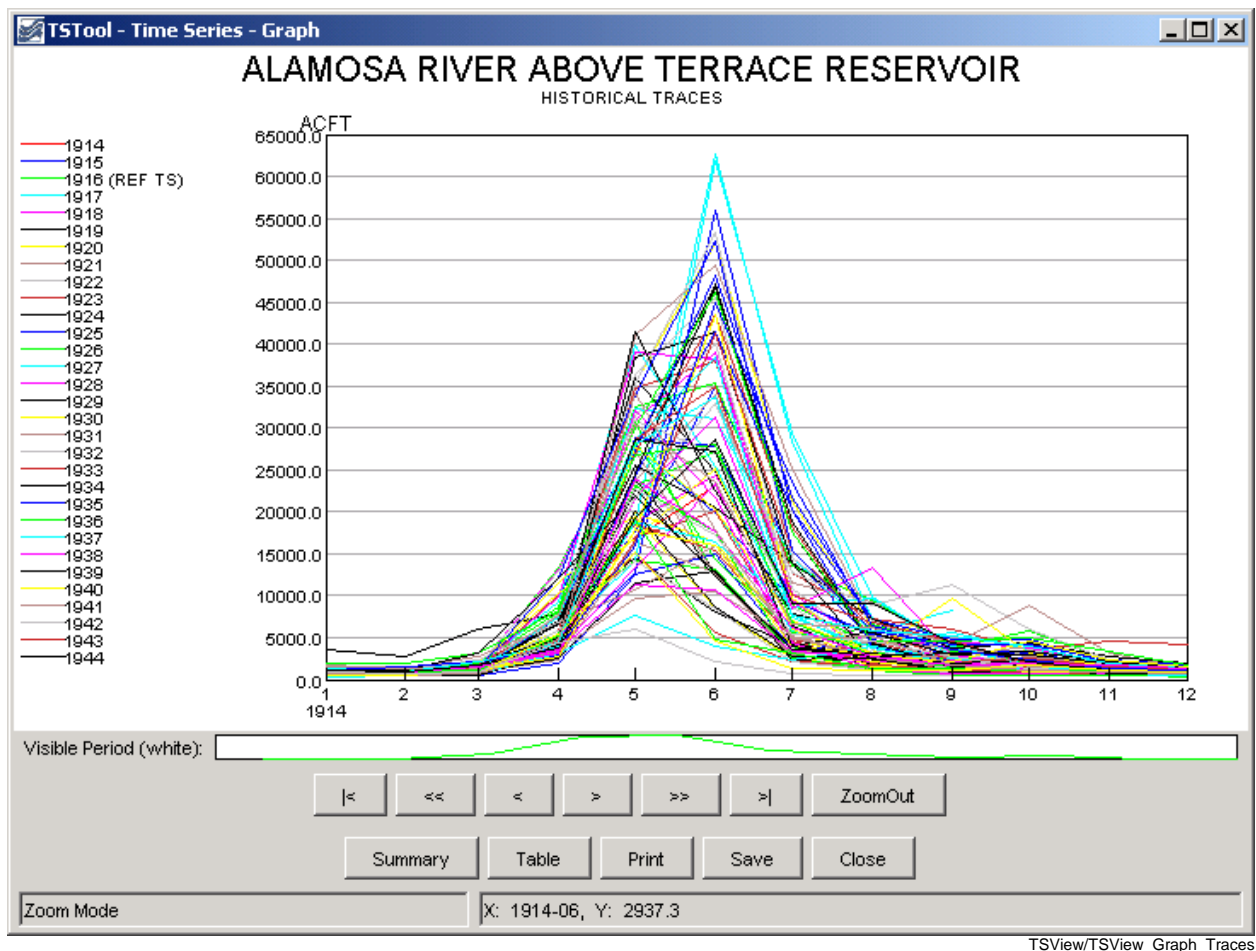
Most supported time series input types do not inherently use aliases (an exception is the `DateValue` format, which will initialize a time series' alias if the input `DateValue` file specifies the information). Instead, time series typically are identified by the location part of the time series identifier (e.g., a station identifier). Although time series can use aliases to simplify processing, the location part of the identifier will generally be used when outputting time series to files or databases.

The time series manipulation features of TSTool facilitate using variations of an input time series for analysis. For example, a time series may be read and manipulated to produce several variants, which are then written for use in a model or analysis. The `TS Alias = Copy()` command could be used to create copies of the original time series. An alternative is to use the `TS Alias = NewTimeSeries()` command to create a new time series (specifying a location part of the time series identifier that is suitable for output), and then use the `SetFromTS()` command to copy all or part of the original time series into the new time series. These two commands therefore allow one time series to be read and copied into new time series, each of which has a new location in the time series identifier. Other commands also allow aliases to be assigned as time series are created.

If specified, time series aliases are generally output in the legends of graphs and other data products.

2.6 Time Series Ensembles

A time series ensemble is a group of related, typically overlapping, time series. Ensembles can be used to manage related scenarios (e.g., input and results of model scenarios) or as a way of shifting a historical time series so that years overlap. Many commands operate on a list of time series by using the parameters `TSList=EnsembleID` and `EnsembleID="SomeID"`. Statistics time series can be derived from ensembles, for example to calculate the average condition over time (although care must be taken in whether this can be interpreted as a time series of related values, for use as input to a process). Ensembles are assigned unique identifiers and are displayed at the bottom of the TSTool main window in a separate results tab. The following figure illustrates an ensemble of annual time series created from a long historical time series.



Example Trace Ensemble Plot Showing Historical Years

This page is intentionally blank.

3 Getting Started

Version 08.15.03, 2008-06-10,

This chapter provides an overview of the TSTool graphical user interface. The TSTool GUI has three main functions:

1. Display and analyze time series data. In this capacity, a graph or summary can be created and then TSTool can be closed.
2. Format lists of time series for use with simulation models or other software. In this capacity, time series that are read and displayed can be incorporated into a command file, which can be run to generate model files.
3. Read time series and produce time series products (e.g., image files containing graphs), for use on web sites or to facilitate review of database contents or model output. In this capacity TSTool is used to generate data products in a streamlined fashion.

The remainder of this chapter provides an overview of the graphical user interface, in the order of the menus on the menu bar.

3.1 Starting TSTool

Within the State of Colorado's CDSS, TSTool can be started using **Start...All Programs...CDSS...TSTool** (or **Start...Programs...CDSS...TSTool**).

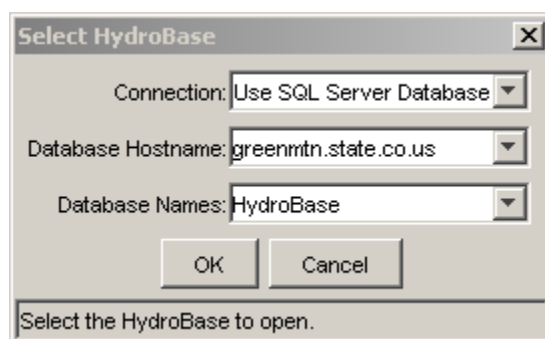
If a command file has been created, it can be processed in batch mode using the following command line:

```
tstool -commands commands.TSTool
```

It is customary to name command files with a *.TSTool* file extension.

3.2 Select HydroBase Dialog

If the HydroBase input type is enabled (see the **HydroBase Input Type Appendix**), the HydroBase login dialog will be automatically shown when TSTool starts in interactive mode. The dialog is used to select a server and database for the State of Colorado's HydroBase database. A HydroBase database can also be selected from the **File...Open...HydroBase...** menu.



Menu_Open_HydroBase

Select HydroBase Database Dialog

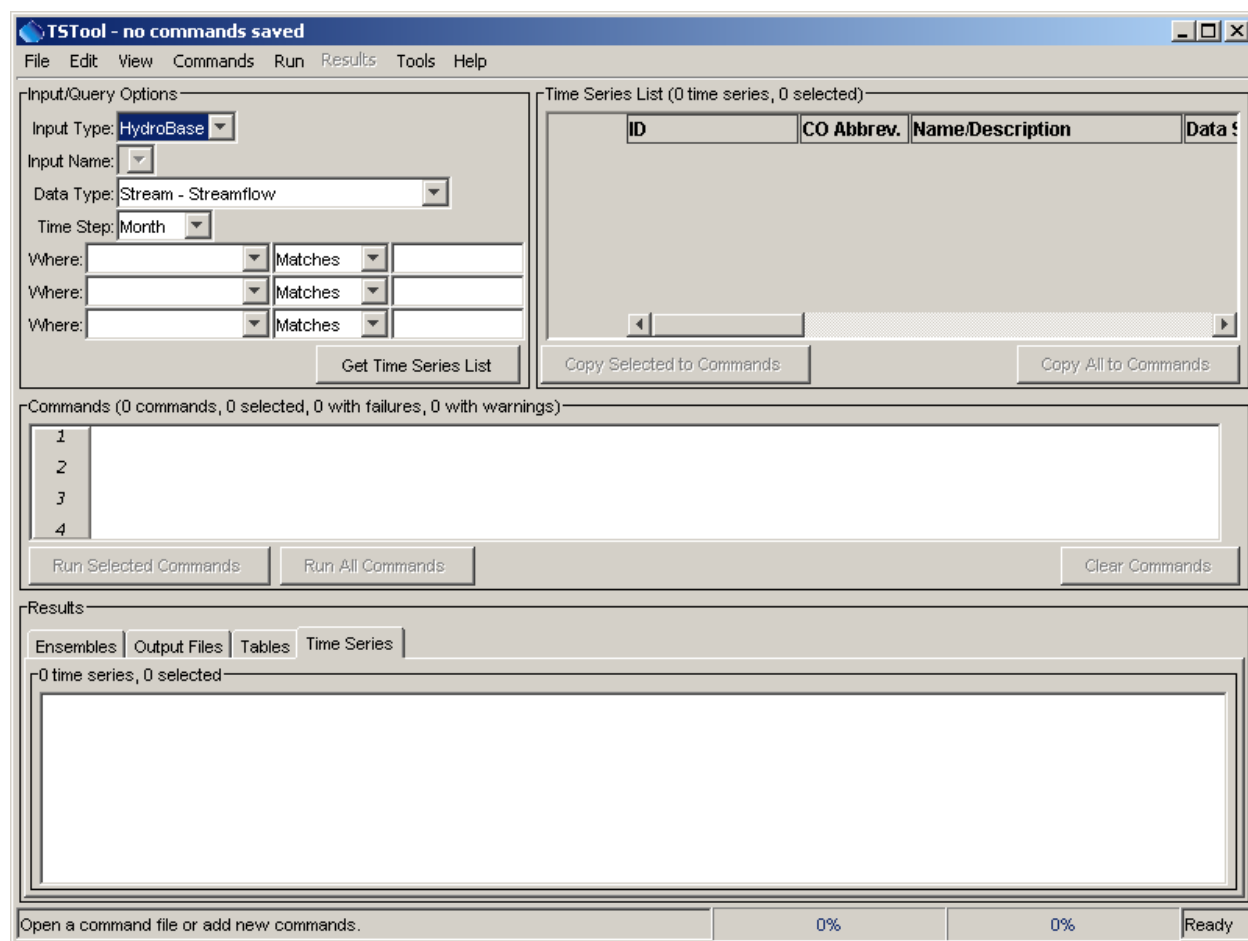
You can also cancel the login, in which case HydroBase features will be disabled but you will be able to work with other input types.

3.3 Main Interface

The following figure illustrates the main TSTool interface during a typical session, immediately after starting and completing the HydroBase database login. The main interface is divided into three main areas:

- **Input/Query Options** (top left) and **Time Series List** area (top right)
- **Commands** (middle)
- **Time Series Results** (bottom)

Status and progress information is displayed at the bottom of the main window and also in the borders around main panels (e.g., to show how many items are in a list and how many are selected).



GUI_MainBlank

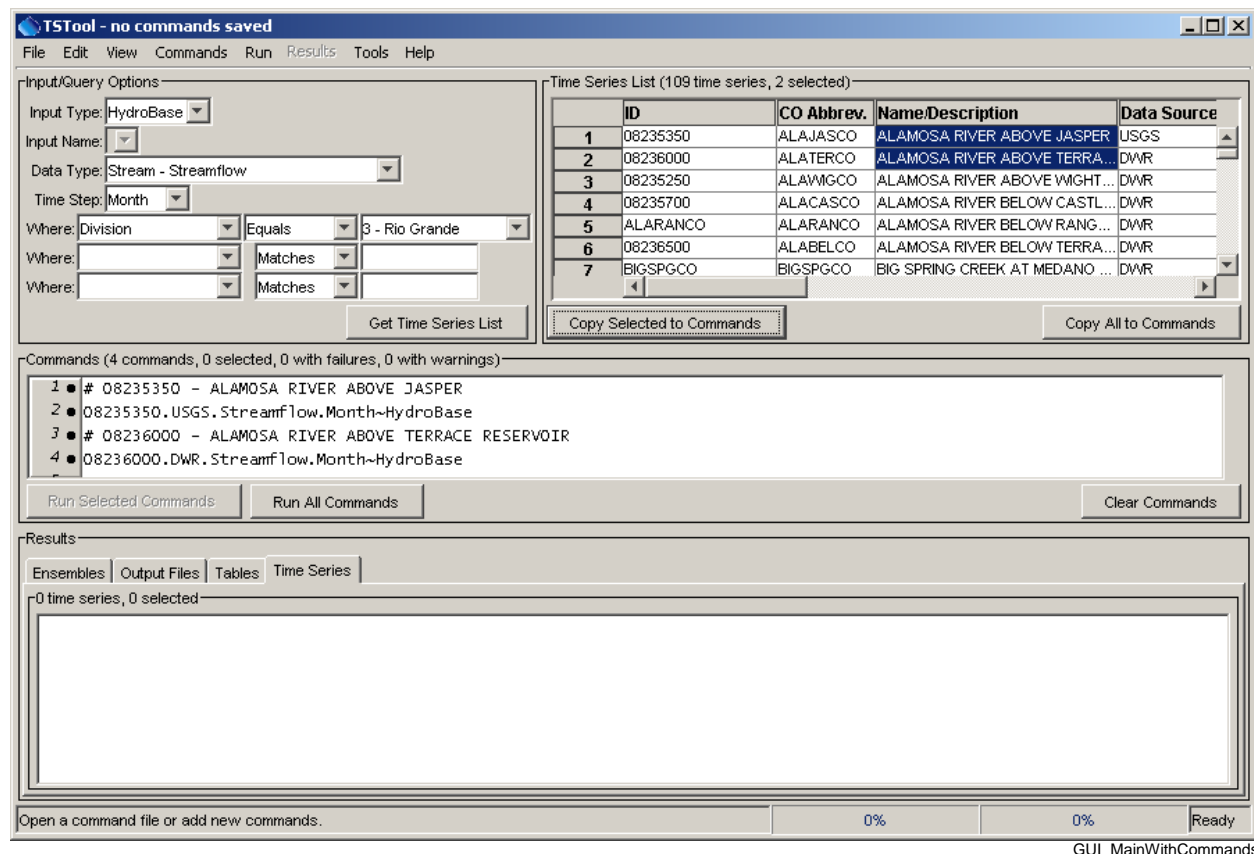
Initial TSTool Interface

3.3.1 Input/Query Options and Time Series List Area

The upper part of the main window contains the **Input/Query Options** and **Time Series List** area. The **Input/Query Options** choices help select time series information from input types. The interactive interface is useful when searching a database or selecting a time series from a file that includes multiple time series. An alternative to the following interactive approach is to use read commands from the **Commands** menu (see the **Commands** chapter), which is appropriate for more complicated analysis. To select time series, execute the following steps:

1. Select the **Input Type**. Input types define the storage format (e.g., database or file) for time series data. The DateValue input type is the default. More specific input types (e.g., the HydroBase database) may be the default if enabled. See the appropriate appendix for a description about supported input types. Depending on the input type, some of the remaining selection choices may be disabled or limited. In most cases, TSTool will assume that you are correctly associating an input type with the actual input that is selected (i.e., that you will select a file that matches the input type when **Get Time Series List** is pressed – see below). Selecting some input types may prompt for a file, which is then listed in the **Input Name** choices.
2. Select the **Data Type** (if appropriate for the input type). For example, select **Streamflow** or **Diversion** if using a HydroBase input type. For some input types, the data type will be listed as **Auto**, indicating that the data type automatically will be determined from the input itself.
3. Select the **Time Step** (if appropriate for the input type). The time step, also referred to as the data interval, will generally be limited by the input type. For example, if reading from the HydroBase database, the Streamflow data type will result in Day, Month, and Irregular (real-time) time steps being listed. The time step will be shown as **Auto** for some input and data types and will be determined as data are read.
4. Specify the **Where** and **Is** clause(s) for the query (if appropriate for the input type). This information will limit the number of time series that are returned. For some input types, choices will be displayed as choices, whereas for other input types, all time series for the input type will be listed.
5. Press the **Get Time Series List** button in the **Input/Query Options** area, and TSTool will display a list of matching time series in the **Time Series List**. If the input type is a file, you may first be prompted to select the file containing the time series. The **Time Series List** shows a list of matching time series, including standard time series information. As much as possible, the column headings are consistent between different input types. The results are typically sorted by name or identifier if from a database, or if read from a file are listed according to the order in the file. Right-click on the column headings and select **Sort Ascending** to sort by that column (the other columns will adjust accordingly). The sorts are alphabetical so some numeric fields may not sort as expected due to spaces, etc.
6. Move time series to the **Commands** list in the center of the main interface selecting rows in the Time Series list (must click in column two or greater) and then pressing the **Copy Selected to Commands** button. Or, if appropriate, press the **Copy All to Commands** button. The **Commands** and **Time Series Results** areas are discussed in the following section and can contain mixed data types and time steps. Analysis commands may require that the units are compatible, but general viewing tools do not require the same units. To mix data types, make multiple queries using the **Input/Query Options** and **Time Series List** areas and select from the lists as necessary, accumulating time series identifiers in the **Commands** list.

After providing selection information, the main interface might look like the following figure:



GUL_MainWithCommands

TSTool after Pressing *Get Time Series List* and selecting from *Time Series List*

3.3.2 Command List and Command Error Indicators

The **Commands** list occupies the middle of the main interface and contains:

- Time series identifiers corresponding to time series selected from the **Time Series List**, and
- Commands selected from the **Commands** menu (see **Chapter 4 – Commands**).

Time series identifiers are added to the **Commands** list by single clicking on items in the **Time Series List** and copying the identifiers to the **Commands** list. Time series identifiers are formatted by transferring information from the appropriate columns in the **Time Series List** to the **Commands** list.

An alternative to using the **Time Series List** to select time series is to use specific read commands from the **Commands** menu (e.g., use a `ReadDateValue()` command). Using read commands is useful when performing a more complicated analysis on specific input sources (e.g., specific data files) or when processing large amounts of data. The interactive interface is useful when searching a database or generating a simple graph.

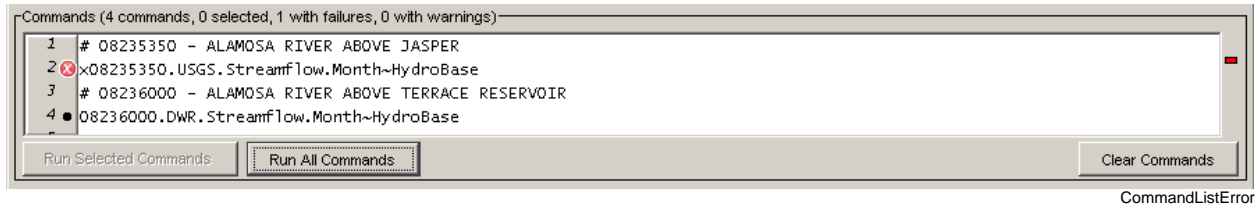
The **Commands** (and the **Time Series Results**) lists behave according to Windows conventions:

- **Single-click** to select one item.
- **Ctrl-click** to additionally select an item.
- **Shift-click** to select everything between the previous selection and the current selection.

Right-clicking over the **Commands** list displays a pop-up menu with useful command manipulation choices, some of which are further described in following sections (e.g., edit menu choices are discussed in **Section 3.5 - Edit Menu**). A summary of the pop-up menu choices is as follows:

Menu Choice	Description
Show Command Status Success/Warning/Failure	Displays the status of a command for each phase of command processing (more discussion below after table).
Edit	Edit the selected command using custom edit dialogs, which provide error checks and format commands. Double-clicking on a command also results in editing the command.
Cut	Cut the selected commands for pasting.
Copy	Copy the selected commands for pasting.
Paste	Paste commands that have been cut/copied, pasted after the selected row.
Delete	Delete the selected commands (currently same as Cut).
Find Commands(s)	Find commands in the command list. This displays a dialog. Use the right-click in the found items to go to or select found items.
Select All	Select all the commands.
Deselect All	Deselect all the commands. This is useful because only selected commands are processed (or all if none are selected). It is therefore important not to unknowingly have one or a few commands selected during processing.
Convert Selected Commands to # Comments	Convert selected commands to # comments.
Convert Selected Commands from # Comments	Convert # comments to commands.
Run All Commands (create all output)	Run all commands and create output (e.g., graphs and files).
Run All Commands (ignore output commands)	Run all commands but skip any output commands. This is useful if a batch command file has been read and time series are to be listed in the GUI but output products are not to be generated automatically.
Run Selected Commands (create all output)	Run selected commands and create output (e.g., graphs and files).
Run Selected Commands (ignore output commands)	Run selected commands but skip any output commands. This is useful if a batch command file has been read and time series are to be listed in the GUI but output products are not to be generated automatically.

Initial enhancements to command editing and error handling were implemented in version 7.00.00 and continue to be implemented. Commands are numbered to simplify editing. The command list also includes left and right gutters to display graphics that help with error handling. The following figure illustrates a command with an error (the first time series identifier has been edited to include an x, resulting in an invalid identifier).



Command List Illustrating Error

The following error handling features are available:

- Clicking on the left gutter will hide and un-hide the gutter.
- The graphic in the left gutter indicates the severity of a problem (see below for full explanation).
- The colored box on the right indicates the severity of a problem and, when clicked on, positions the visible list of commands to display the command corresponding to the problem.
- Commands have three phases: 1) initialization, 2) discovery, 3) run. Initialization occurs when reading a command file or adding a new command. The discover phase is executed only for commands that generate time series for other commands and provides other commands with identifiers used in command editing. The run phase is when commands are processed to generate output.
- Positioning the mouse over a graphic in the left or right gutter will show a popup message with the problem information. The popup is only visible for a few seconds so use the right-click popup menu **Show Command Status (Success/Warning/Failure)** for a dialog that does not automatically disappear.

The meaning of the gutter symbols is described in the following table. The goal is to display the most severe status to indicate problems; however, some commands are still being enhanced to support this feature and may not display a status until after they are run.

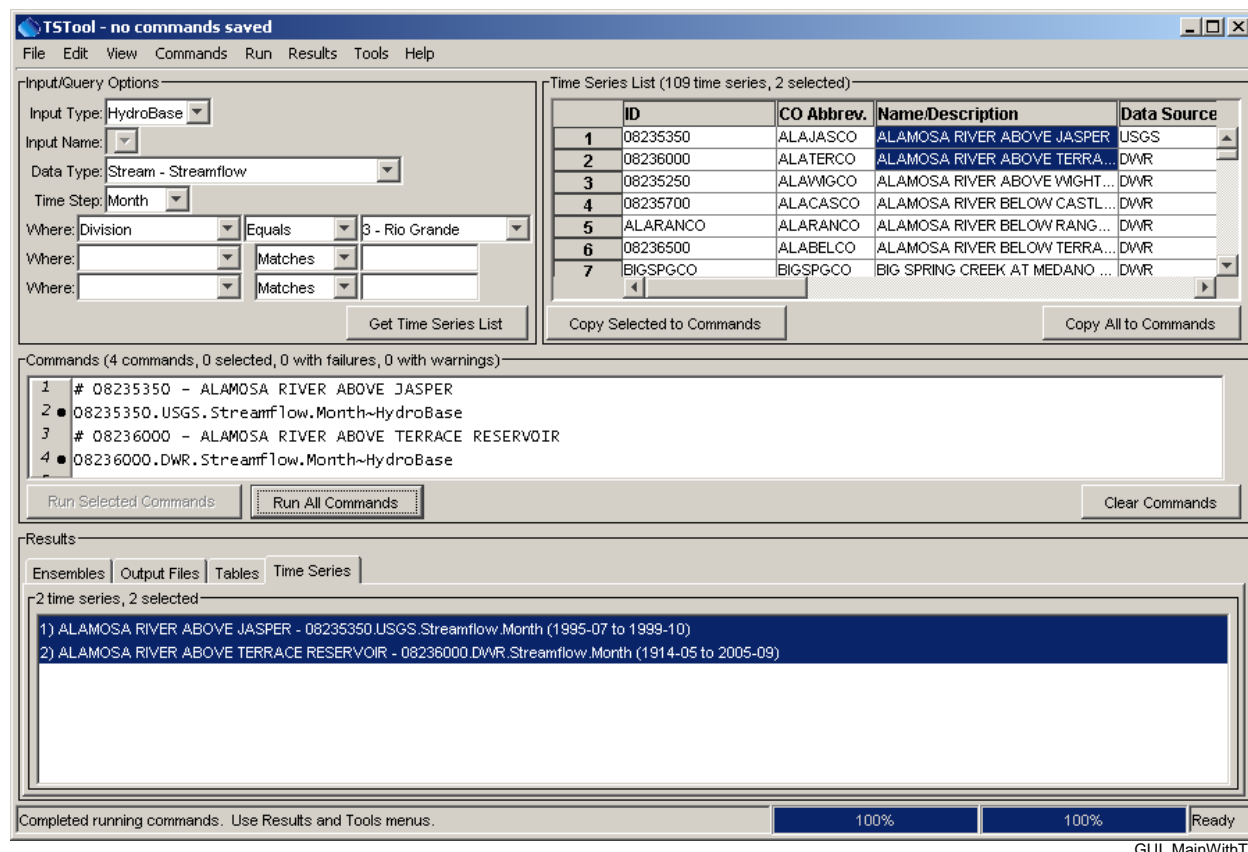
Command List Error Handling Graphics

Command List Left Gutter Graphic	Description
No graphic	Command is successful (a warning or failure has not been detected).
■	The status is unknown, typically because the command has not been updated to fully take advantage of error handling features.
⚠	The command has a problem that has been classified as non-fatal. For example, a command to fill data may be used but results in no data being filled. In general, commands with warnings need to be fixed unless work is preliminary.
✖	The command has failed, meaning that output is likely incomplete. A problem summary and recommendation to fix the problem are available in the status information. Commands with failures generally need to be fixed. Software support should be contacted if the fix is not evident.

Warning and failure messages continue to be enhanced as commands are updated to support the new error handling features.

3.3.3 Time Series Results

The commands in the **Commands** list are processed by pressing the **Run Selected Commands** or **Run All Commands** buttons below the commands list area (or by using the **Run** menu). The time series and other output that result from processing are listed in the bottom of the main interface, as shown in the following figure:



TSTool after Running Commands

The time series listed in the **Time Series Results** list can then viewed using the **Results** menu, analyzed further using the **Tools** menu, and output using the **File...Save...** menus. Only the selected time series will be output (or all if none are selected).

In addition to time series, the following results may also be available:

- **Ensembles** – groups of time series with an ensemble identifier. Time series in an ensemble are also shown in the **Time Series** tab. Right-click on item to access viewing and analysis options.
- **Output Files** – files that are created during processing. Single click on a file to view.
- **Tables** – column-oriented tables created during processing.

Two progress bars at the bottom of the main window are updated during processing. The left progress bar indicates the overall progress in processing the commands (100% means that all commands have been processed). The right progress bar is used with command that support incremental progress status

updating, a feature that will be phased in over time. For example, if a single command processes many time series, this progress bar can be used to indicate progress in the command.

Right-clicking over the **Time Series Results** list displays a pop-up menu with useful time series viewing choices, including a choice to view the time series properties. The right-click menu choices are summarized below:

Time Series Results List Popup Menu Choices

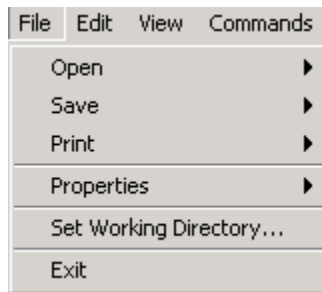
Menu Choice	Description
Graph - Bar (left of date)	Display bar graph for selected time series, drawing bars to the left of the date.
Graph - Bar (center on date)	Display bar graph for selected time series, drawing bars centered on the date.
Graph - Bar (right of date)	Display bar graph for selected time series, drawing bars to the right of the date.
Graph - Duration	Display a duration graph for the selected time series.
Graph - Line	Display a line graph for selected time series.
Graph - Line (log Y-axis)	Display a line graph for the selected time series, using a log10 y-axis.
Graph - Period of Record	Display a period of record graph for the selected time series.
Graph - Point	Display a graph using symbols but no connecting lines.
Graph - Predicted Value	Display a graph of data and the predicted values from regression.
Graph - Predicted Value Residual	Display a graph of data minus the predicted values from regression.
Graph - XY-Scatter	Display an XY-scatter plot for the selected time series.
Table	Display a scrollable table for the selected time series.
Report - Summary	Display a summary for selected time series.
Find Time Series...	Find time series in the time series list. This displays a dialog. Use the right-click in the found items to go to or select found items.
Select All for Output	Select all time series for output.
Deselect All	Deselect all time series for output.
Time Series Properties	Display the time series properties dialog (see the TSView Time Series Viewing Tools appendix for a complete description of the properties interface).

Viewing capabilities are described further in **Section 3.9 - Results Menu** and the **TSView Time Series Viewing Tools** appendix.

The remainder of this chapter summarizes the TSTool menus.

3.4 File Menu - Main Input and Output Control

The **File** menu provides standard input and output features as described below. Some menus are visible only when certain input types are enabled (see the **Installation and Configuration Appendix**). Some menus are only enabled when time series have been processed.

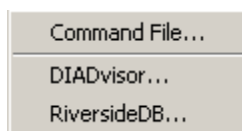


Menu_File

File Menu

3.4.1 File...Open – Open Command File or Databases

The **File...Open** menu displays menu items as follows:



Menu_File_Open

File...Open Menu

The **File...Open...Command File** menu item displays a dialog to select an existing command file. After a file is selected, the file contents replace the contents of the **Commands** list. If commands already exist in the **Commands** list and have been modified, you are given the option of saving the existing commands first. Opening a command file causes the working directory to be set to the directory from which the command file was read, as if the `setWorkingDir()` command was executed. Consequently, `setWorkingDir()` commands may not be needed.

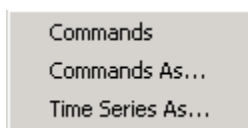
TSTool will attempt to automatically update older command files to new syntax if a command has changed. If a change occurs, the command file will be marked as modified and will need to be saved to reflect the changes. If an error occurs updating a command, it will be marked with an error and a comment will be inserted with the original command and indicating that an automated update could not occur. Unrecognized commands are marked with an error and will generate errors if run.

The **File...Open...HydroBase** menu item displays the **Select HydroBase** dialog discussed in **Section 3.2** (see also the **HydroBase Input Type Appendix**).

The **File...Open...RiversideDB** menu item displays a dialog to select a RiverTrak[®] System configuration file, which specifies the location of a RiversideDB database (see the **RiversideDB Input Type Appendix**).

3.4.2 File...Save – Save Command File, and Time Series

The **File...Save** menu displays the following menu choices:



Menu_File_Save

File...Save Menu

The **File...Save...Commands** and **File...Save...Commands As** menu items save the contents of the **Commands** list to a file. The name of the current command file is shown in the TSTool title bar. All commands are saved, even if only a subset is selected. Saving a command file causes the working directory to be set to the where the command file was written, as if the `setWorkingDir()` command was executed. Consequently, `setWorkingDir()` commands may not be needed.

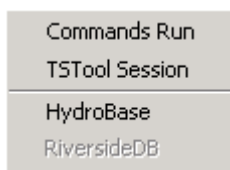
The **File...Save...Time Series As** menu item displays a file chooser dialog for saving time series in the **Time Series Results** list. See the **Input Type Appendices** for examples of supported file formats. Only the selected time series in the **Time Series Results** list are saved (or all, if none are selected). Not all formats are supported because in most cases the write commands are used to automate processing of time series.

3.4.3 Print Commands

The **File...Print...Commands** menu prints the contents of the **Commands** list. This is useful while editing and troubleshooting commands.

3.4.4 Properties for Commands Run, TSTool Session, and Input Types

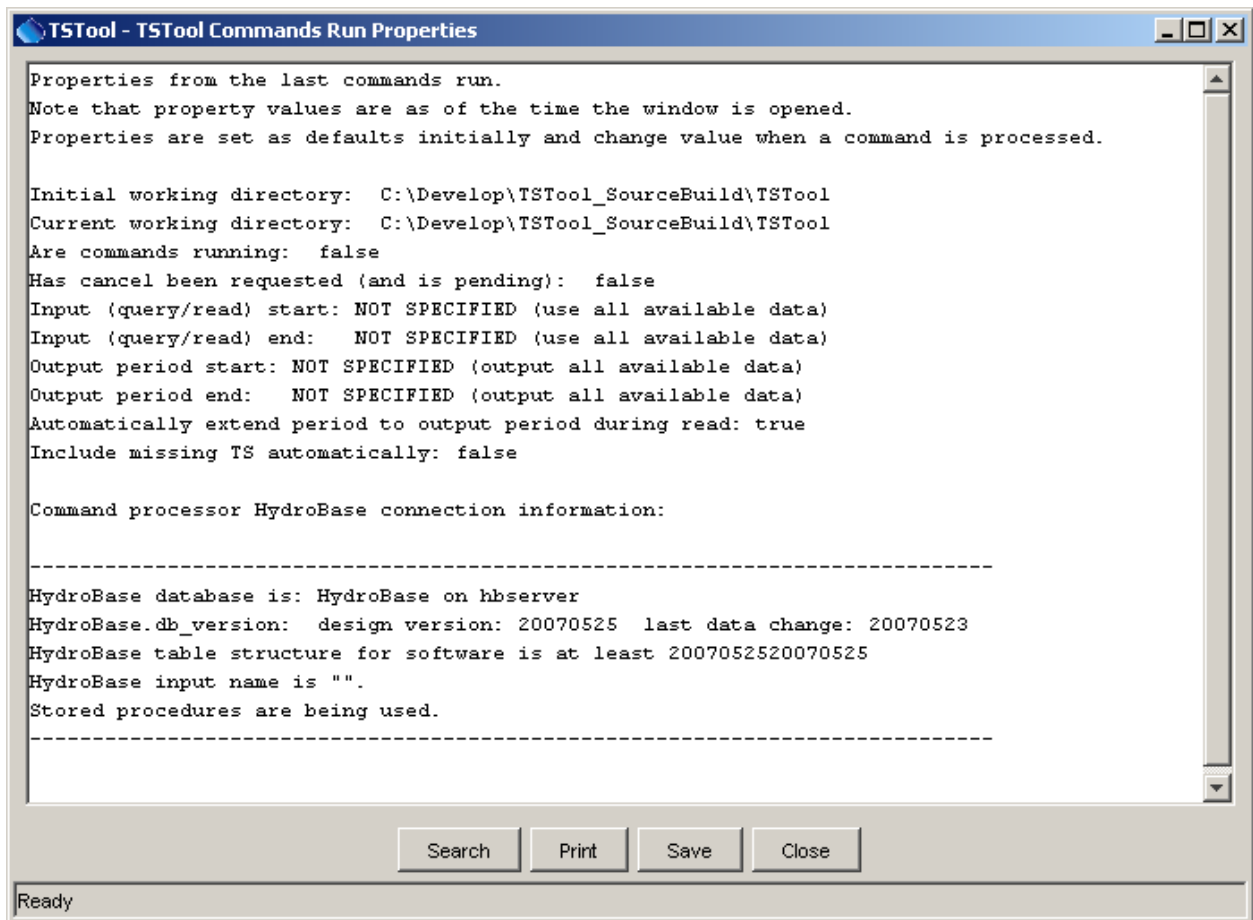
The **File...Properties** menu displays the following menu items:



Menu_File_Properties

File...Properties Menu

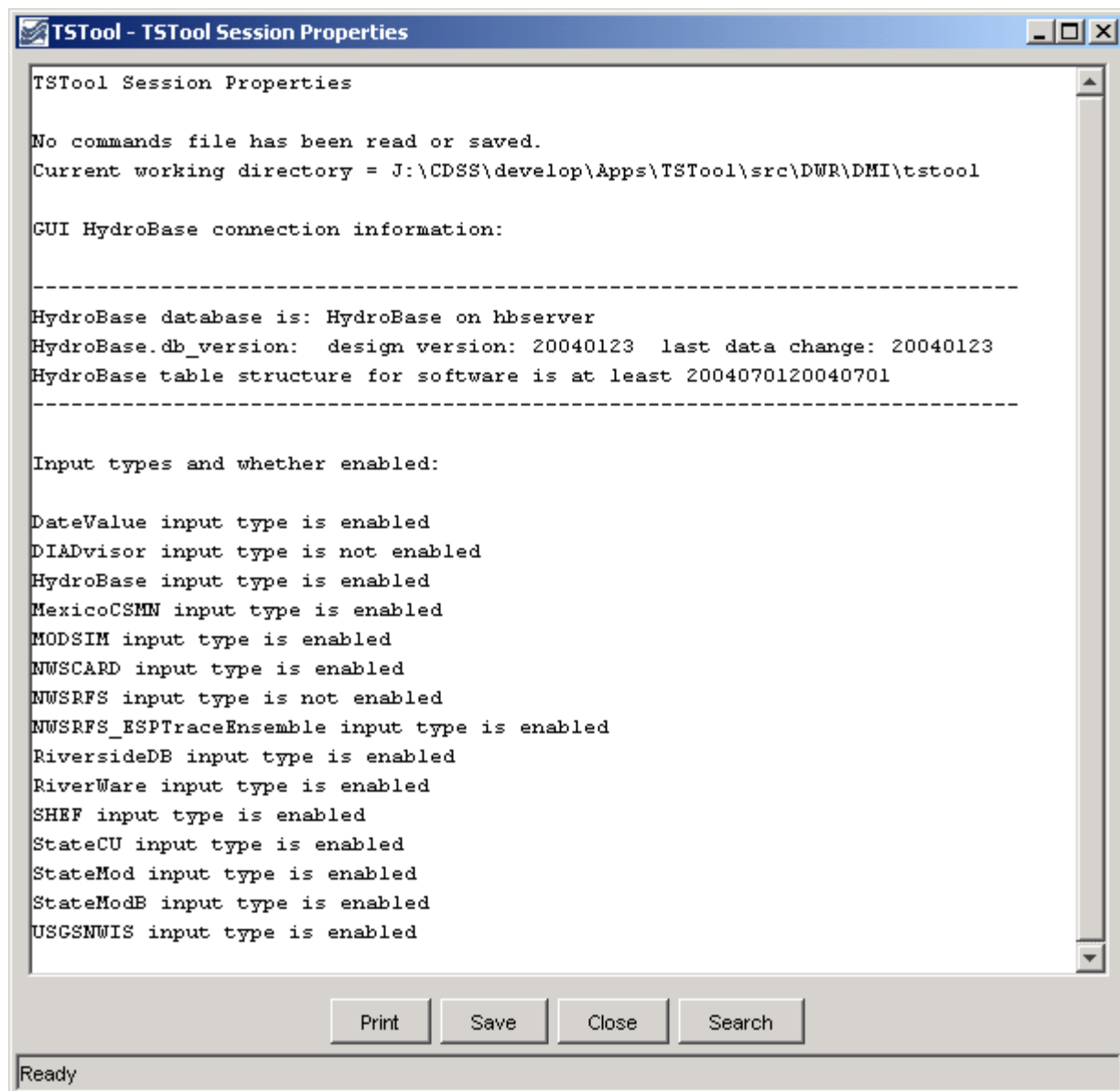
The **File...Properties...Commands Run** menu item displays information from the last time that the commands were run, including global properties that impact results:



Menu_File_PropertiesRun

Properties of the Last Commands Run

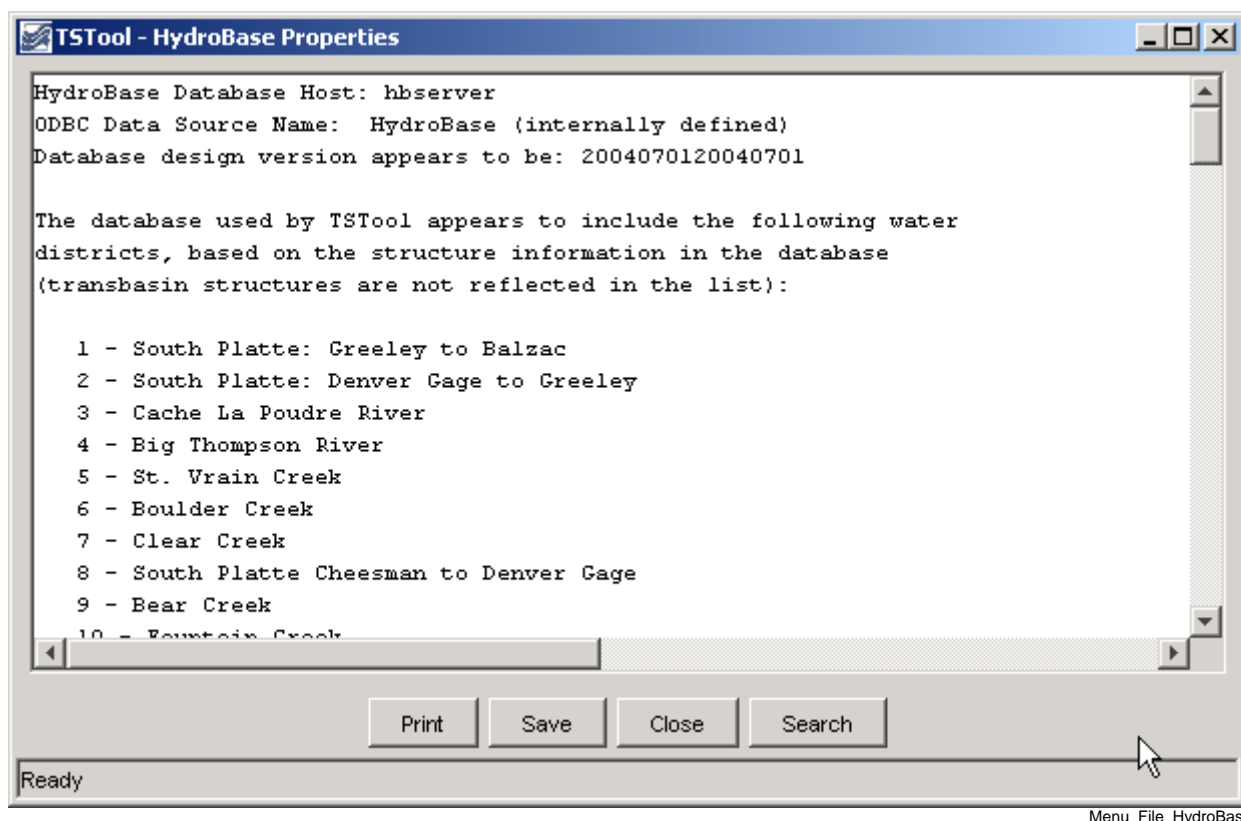
The **File...Properties...TSTool Session** menu item displays information about the current TSTool session, as follows:



TSTool Session Properties

Menu_File_Properties_TSToolSession

The **File...Properties...HydroBase** menu item displays HydroBase properties, including the database that is being used, database version, and the water districts that are in the database being queried. The water districts are determined from the structure table in HydroBase. The information that is shown is consistent with that shown by other State of Colorado tools and is useful for troubleshooting.



HydroBase Properties Dialog

The **File...Properties...RiversideDB** menu item displays RiversideDB properties, if a RiversideDB connection is in place.

3.4.5 Set Working Directory

The **File...Set Working Directory** menu item displays a file chooser dialog that allows you to select the working directory. The working directory, when set properly, can greatly simplify command files because relative file paths can be used for input and output. The working directory is normally set in one of the following ways, with the current setting being defined by the most recent item that has occurred:

1. The startup directory for the TSTool program,
2. The directory where a command file was opened,
3. The directory where a command file was saved,
4. The directory specified by a `SetWorkingDir()` command,
5. The directory specified by **File...Set Working Directory**.

The menu item is provided to allow the working directory to be set before a command file has been saved (or opened) and it typically eliminates the need for `SetWorkingDir()` commands in command files.

3.4.6 Exit

The **File...Exit** menu exits TSTool. You will be prompted to confirm the exit. If commands have been modified, you will be prompted to save before exiting. Commands may have been automatically updated by TSTool if an old command file was read.

3.5 Edit Menu – Editing Commands

The **Edit** menu can be used to edit the **Commands** list. Edit options are enabled and disabled depending on the status of the **Commands** list. Specific edit features are described below. Right clicking over the **Commands** list provides a popup menu with many choices described below.



Menu_Edit

Edit Menu

3.5.1 Cut/Copy/Paste/Delete

The **Edit...Cut** and **Edit...Copy** menu items are enabled if there are items in the **Commands** list. **Currently, these features do not allow interaction with other applications.** **Cut** deletes the selected item(s) from the **Commands** list and saves its information in memory. **Copy** just saves the information in memory. After **Cut** or **Copy** is executed, select an item in the **Commands** list and use **Paste** (see below).

Paste is enabled if one or more commands from the **Commands** list has been cut or copied. To paste the command(s), select commands in the **Commands** list and press **Edit...Paste**. The commands will be added after the last selected command. To insert at the front of the list, paste after the first command, and then cut and paste the first command to reverse the order.

The **Delete** choice currently works exactly like the **Cut** choice. Additionally, after lines in the **Commands** have been selected, you can press the **Clear Commands** button below the **Commands** list to cut/delete.

The **Clear Commands** button in the **Commands** area deletes the selected commands or all commands if none are selected. You will be prompted to confirm the clear if no commands are specifically selected.

3.5.2 Select All Commands/Deselect All Commands

The **Edit...Select All Commands** and **Edit...Deselect All Commands** menu items are enabled if there are items in the **Commands** list. Use these menus to facilitate editing. Note that when editing commands it is often useful to deselect all commands so that new commands are added at the end of the commands list.

3.5.3 Edit Command File

The **Edit...Command File** menu choice can be used to edit a command file using **Notepad** on Windows or **edit** on UNIX machines. Currently, there is no way to change the editor. You must re-read the command file into TSTool after using the editor for TSTool to recognize the time series commands in the command file. This feature is less useful than in the past because editor dialogs have now been implemented for all commands.

3.5.4 Edit Command

The **Edit...Command** menu can be used to edit an individual command. TSTool will determine the command that is being edited and will display the editor dialog for that command, performing data checks. **Most old commands will be automatically detected and will be converted to new command syntax.** This feature is also accessible by right clicking on the **Commands** list and selecting the **Edit** menu item.

3.5.5 Convert Selected Commands To/From Comments

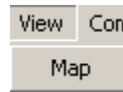
The **Edit...Convert selected commands to comments** menu can be used to toggle selected commands in the **Commands** list to comments (lines that begin with #). This is useful when temporarily disabling commands, rather than deleting them.

The **Edit...Convert selected commands from comments** menu can be used to toggle selected commands in the **Commands** list from comments back to active commands. This is useful when re-enabling commands that were temporarily disabled.

Note that the multi-line `/* */` comment notation can be inserted using the **Commands...General – Comments** menu.

3.6 View Menu – Display Map Interface

The **View** menu currently has limited choices:



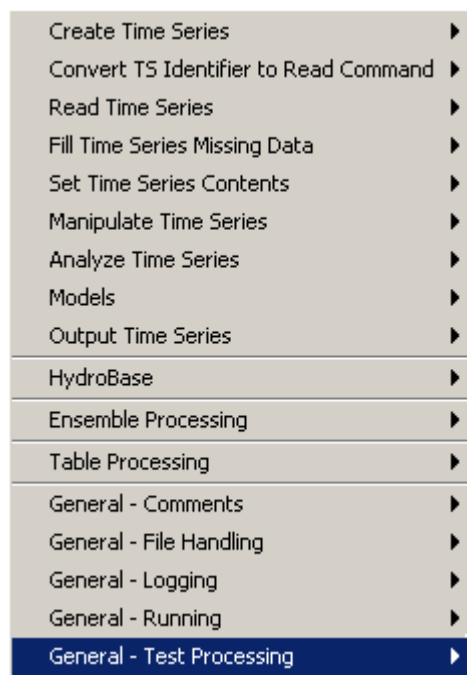
Menu_View

View Menu

The **View...Map** menu displays a map interface in a separate window. See the **Using the Map** chapter for more information.

3.7 Commands Menu

The **Commands** menu provides several menus (as shown in the following figure), which allow time series processing commands to be inserted into the **Commands** list.



Menu_Commands

Commands Menu

Time series commands are organized into the following categories:

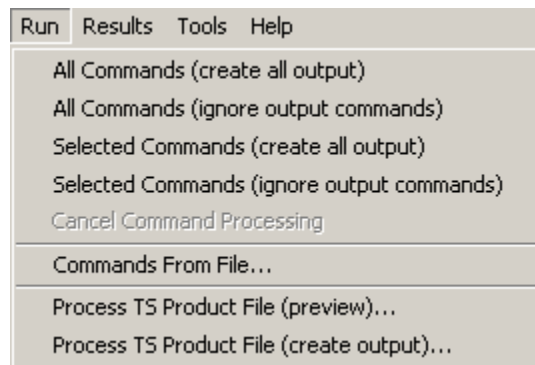
1. **Create Time Series** - create one or more new time series in memory
2. **Convert TS Identifier to Read Command** – convert a time series identifier in the **Commands** list area to a read command
3. **Read Time Series** – read time series from a file or database
4. **Fill Time Series Missing Data** - fill missing data
5. **Set Time Series Contents** – set time series data or properties
6. **Manipulate Time Series** - manipulate data by transforming the contents of the time series (e.g., scale a time series' data values)
7. **Analyze Time Series** – perform analysis on time series, without modifying the time series

8. **Models** – advanced or specific models that operate on time series data
9. **Output Time Series** – write time series results to a file or produce graphical products
10. **HydroBase** – commands specific to the HydroBase database
11. **Ensemble Processing** – commands that are specific to ensemble processing
12. **Table Processing** – commands that are specific to table processing
13. **General – Comments** – insert comments
14. **General – File Handling** – commands to manipulate files (e.g., remove)
15. **General – Logging** – commands for logging (e.g., open a log file, set message levels)
16. **General – Running** – commands to control running external programs
17. **General – Test Processing** – commands to process tests, to validate software and procedures

Chapter 4 – Commands discusses commands in more detail and the **Command Reference** at the back of this documentation provides a reference for each command.

3.8 Run Menu – Run Commands

The **Run** menu processes the **Commands** list to generate the **Time Series Results** for output.



Menu_Run

Run Menu

The **Run...All Commands (create all output)** menu will process all the commands in the **Commands** list and create output if appropriate. For example, the `writeStateMod()` command will write the time series that are in memory to a StateMod file.

The **Run...All Commands (ignore output commands)** menu will process the commands in the **Commands** list, ignoring commands that generate output products. With this option, you can process a command file prepared for batch mode, but only have the time series available for viewing in the GUI rather than generating the output files. For example, `writeStateMod()` commands will not be processed. This increases performance and minimizes creation of files.

The **Run...Selected Commands** menu items are similar to the above, except that only selected commands are run.

The **Run...Cancel Command Processing** menu items will be enabled if command processing is active, and allows the processing to be cancelled. Processing may continue until the current command finishes.

The **Run...Commands From File** choice will run a command file but not generate any time series for viewing in the GUI. This is equivalent to running in batch mode but initiating the run from the TSTool GUI.

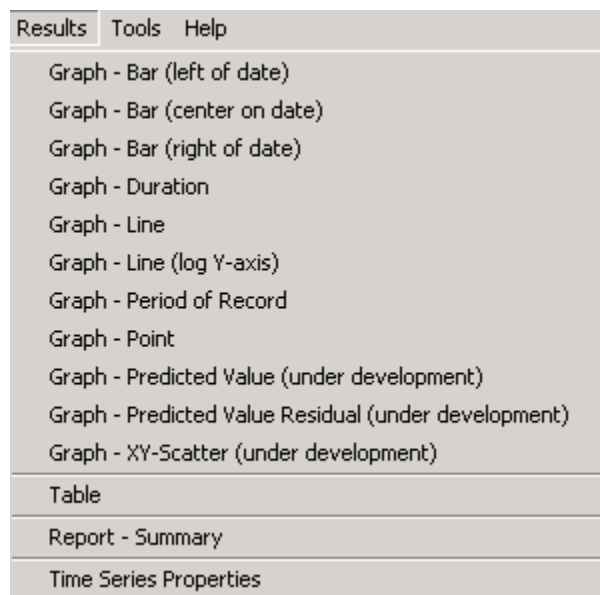
Menu items similar to the above are also available in a popup menu by right clicking on the **Commands** list.

3.8.1 Process TSProduct

The **Run...Process TS Product File** menu items can be used to create time series products by processing time series product definition files. The **TSView Time Series Viewing Tools Appendix** describes the format of these files. Time series product definition files can be saved from graph views using **Save As...Time Series Product**. The `ProcessTSProduct()` command provides equivalent functionality.

3.9 Results Menu – Display Time Series

The **Results** menu displays time series that are listed in the **Time Series Results** list. The time series can be viewed multiple times, using the same time series results.



Menu_Results

Results Menu

Graphing time series results in slightly different viewing options being available, depending on the type of graph. In many cases, you will be able to see three views of time series, consisting of a graph, summary, and table. Additionally, you can select the graph properties and choose the colors and symbols to be used for each time series. The **TSView Time Series Viewing Tools Appendix** describes in detail the graphing tools.

Most of the main **Results** menu choices are available in a popup menu that is displayed when right-clicking on the **Time Series Results** list.

3.9.1 Graph - Bar

Bar graphs are generated by selecting time series from the **Time Series Results** list and pressing **Results...Graph - Bar** menus. See the **TSView Time Series Viewing Tools Appendix** for information about bar graphs. The position of the bars relative to the date/time position depends on whether data are instantaneous, mean, or accumulated.

3.9.2 Graph - Duration

Duration graphs are generated by selecting time series in the **Time Series Results** list and selecting the **Results...Graph - Duration** menu. See the **TSView Time Series Viewing Tools Appendix** for information about duration graphs.

3.9.3 Graph - Line

A line graph is generated by selecting time series in the **Time Series Results** list and then selecting the **Results...Graph - Line** menu. See the **TSView Time Series Viewing Tools Appendix** for information about line graphs.

3.9.4 Graph - Line (log Y-axis)

Log Y-axis line graphs are generated by selecting time series in the **Time Series Results** list and then selecting the **Results...Graph - Line (log Y-axis)** menu. See the **TSView Time Series Viewing Tools Appendix** for information about log Y-axis line graphs.

3.9.5 Graph - Period of Record

The period of record graph is useful to display the availability of data over a period. Horizontal lines are drawn for each time series, with breaks in the line indicating missing data. An alternative to this graph type is to use the **Tools...Data Coverage by Year** report (see **Chapter 5 – Tools**). See the **TSView Time Series Viewing Tools Appendix** for information about period of record graphs.

3.9.6 Graph – Point

Point graphs are useful for data that are collected infrequently. For example, the interval of the data may be daily; however, values may only be available once per month, on various days of the months. See the **TSView Time Series Viewing Tools Appendix** for information about point graphs.

3.9.7 Graph – Predicted Value

The predicted value graph requires as input two time series. First, a regression analysis is performed, similar to the analysis done for the XY-Scatter plot. The original two time series are then plotted, additionally with the time series that would be generated using the regression results. The predicted time series and the original time series will be the same where their periods overlap, with only the predicted time series shown outside of that period.

3.9.8 Graph – Predicted Value Residual

The predicted value residual graph performs the same analysis as the predicted value graph. Where the original and predicted time series overlap, the difference is computed and plotted as a time series. The resulting bar graph therefore shows the relative goodness of fit of the estimated time series.

3.9.9 Graph - XY-Scatter

An XY-scatter graph is generated by selecting two or more time series from the **Time Series Results** list and then selecting the **Results...Graph - XY-Scatter** menu. See the **TSView Time Series Viewing Tools Appendix** for information about XY-Scatter graphs.

3.9.10 Table

A table display is generated by selecting one or more time series from the **Time Series Results** list and then selecting the **Results...Table** menu. See the **TSTool Time Series Viewing Tools Appendix** for information about table displays.

3.9.11 Report - Summary

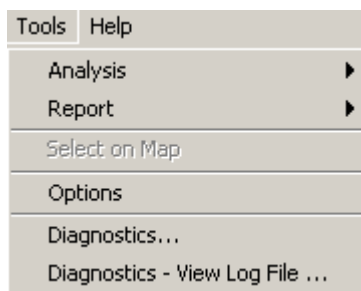
A summary report for time series can be generated by selecting time series in the **Time Series Results** list and then selecting the **Results...Report - Summary** menu. See the **TSTool Time Series Viewing Tools Appendix** for information about summary displays.

3.9.12 Time Series Properties

Time series properties include all the information other than the data values. For example, properties include the period of record, data units, processing history, etc

3.10 Tools Menu

The **Tools** menu lists tools that perform additional analysis on time series that are selected in the **Time Series Results** list. These features are similar to the **Results** features in that a level of additional analysis is performed to produce the data product.



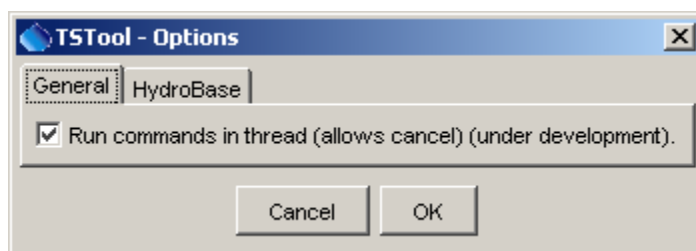
Tools Menu

Menu_Tools

Analysis tools are described in more detail in **Chapter 5 – Tools**. The following sections describe the **Tools...Options** and **Tools...Diagnostics** features.

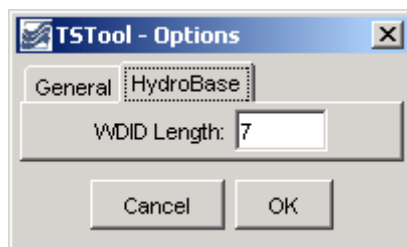
3.10.1 Options

The **Tools...Options** menu displays program options. The **General** tab configures whether processing is run in a thread (the default), meaning that the GUI will be responsive while processing occurs. Running without threading should only be used at the suggestion of support when troubleshooting.



Menu_Tools_Options_General

The **HydroBase** tab can be used to specify the total length of water district identifiers (WDIDs), for use with the HydroBase input type.



Tools_Options_HydroBase

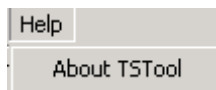
The WDIDs are used as the location part of the time series identifier. A water district identifier is comprised of a two-digit zero-padded water district (e.g., 01, 20) and a zero-padded identifier for structures within the water district. For example, ditch headgates are typically numbered 500 or greater, within each water district. For modeling purposes, the WDIDs are typically treated as character strings. To allow for distinct and unambiguous identifiers, WDIDs are typically padded with zeros to have consistent overall lengths. In the past, six characters were used for identifiers in model data sets. However, this length is insufficient to handle identifiers in some water districts and therefore the default in TSTool is seven characters. This menu item can be used to set the length if TSTool is being used to create time series and the default length is not compatible with the needed output. The WDID length is enforced when time series are listed. If necessary, the time series identifiers can be edited manually to add or remove padding zeros.

3.10.1 Diagnostics

The **Tools...Diagnostics** menu displays the diagnostics interface, which is used to set message levels and view messages as TSTool processes data. **The Tools...Diagnostics – View Log File** menu displays the log file viewer. These tools are useful for troubleshooting problems. Refer to **Chapter 5 Tools** for more information.

3.11 Help Menu

The help menu displays the version of TSTool.



Help Menu

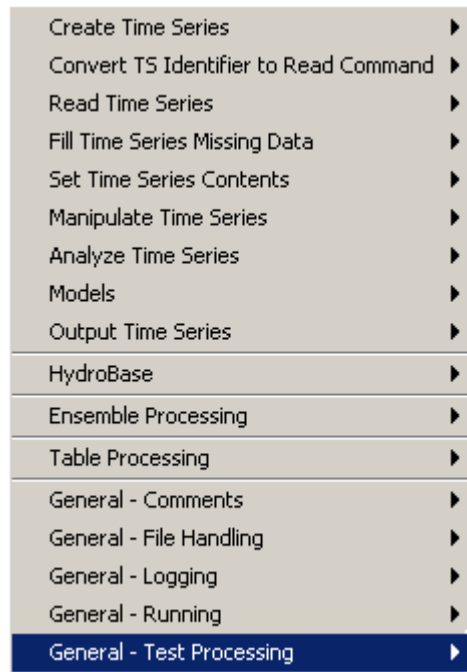
Menu_Help

The **Help...About TSTool** menu displays the program version number, for use in troubleshooting and support.

4 Commands

Version 09.06.01, 2010-02-22

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



Menu_Commands

Commands Menu

Commands are organized into the following categories:

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

The menus for each category are discussed in the following sections. Selecting a command menu item will display a command editor. The editor dialog and command syntax are described in detail in the **Command Reference** at the end of this documentation. Menus are enabled/disabled depending on the state of the application (e.g., whether time series are available).

Many commands allow a list of 1+ time series to be processed. The `TSList` command parameter is available for many commands to indicate how the list of time series to be processed is determined. For example, `TSList=AllTS` will process all available time series. Refer to the command reference for a description of available parameters for a command. The standard values for the `TSList` parameter are as follows:

TSList Parameter Value	Description
AllMatchingTSID	Process all time series that match the TSID parameter (typically can contain wildcards). For example, specify <code>TSID=A*</code> or <code>TSID=A*.*.*.Month</code>
AllTS	Process all time series.
EnsembleID	Process all time series for the ensemble identifier.
LastMatchingTSID	Process the last (previous to the current command) TSID that matches the TSID parameter.
SelectedTS	Process all time series that have been selected with <code>SelectTimeSeries()</code> commands.
SpecifiedTSID	Process all time series in the specified list of time series identifiers (for example when used with the <code>Add()</code> command the <code>AddTSID</code> parameter is used to provide the specified identifiers).

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

4.1 Create Time Series

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

```
CreateFromList()... <read 1(+) time series from a list of identifiers>
ResequenceTimeSeriesData()... <resequence years to create new scenarios>

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

Menu_Commands_CreateTimeSeries

Commands...Create Time Series Menu

These commands create new time series from external data (see `CreateFromList()`), by using user-supplied data (see `NewTimeSeries()`), or by operating on existing time series. Time series created from existing time series are fundamentally different from the original and cannot take its place with the same identifier. For example, the data interval or identifier is different in the new time series.

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

4.2 Converting Time Series Identifier to Read Command

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

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

Menu_Commands_ConvertTSID

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

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

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

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

4.3 Read Time Series

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

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

Menu_Commands_ReadTimeSeries

Commands...Read Time Series Menu

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

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

4.4 Fill Time Series Data

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

```
FillConstant()... <fill TS with constant>
FillDayTSFrom2MonthTSAnd1DayTS()... <fill daily time series using  $D1 = D2 * M1 / M2$ >
FillFromTS()... <fill time series with values from another time series>
FillHistMonthAverage()... <fill monthly TS using historic average>
FillHistYearAverage()... <fill yearly TS using historic average>
FillInterpolate()... <fill TS using interpolation>
FillMixedStation()... <fill TS using mixed stations (under development)>
FillMOVE2()... <fill TS using MOVE2 method>
FillPattern()... <fill TS using WET/DRY/AVG pattern>
FillProrate()... <fill TS by prorating another time series>
FillRegression()... <fill TS using regression>
FillRepeat()... <fill TS by repeating values>
FillUsingDiversionComments()... <use diversion comments as data - HydroBase ONLY>

SetAutoExtendPeriod()... <for data filling and manipulation>
SetAveragePeriod()... <for data filling>
SetIgnoreLEZero()... <ignore values  $\leq 0$  in historical averages>
SetMissingDataValue()... <for data filling>
SetPatternFile()... <for use with fillPattern() >
SetRegressionPeriod()... <for fillRegression()>
```

Menu_Commands_FillTimeSeries

Commands...Fill Time Series Missing Data Menu

Fill commands will only change values that are missing, whereas set commands (see next section) will change values regardless of whether they are missing or non-missing. These commands can be executed in sequence to apply multiple fill techniques to time series. Many commands accept the * wildcard for the time series identifier, allowing all time series to be filled.

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

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

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

TSTool handles missing data by internally assigning a special numeric value where data are missing. Different input types may have different missing data values but typically -999, a similar extreme value, or NaN (not a number) is used. If the output period is specified using `SetOutputPeriod()`, then extensions to the available time series period are filled with the missing data value. Data flags are supported for some input types. TSTool displays and output products indicate missing data by blanks, showing the missing data value, or a string (e.g., NC).

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

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

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

TSTool Fill Techniques and Associated Commands

Technique	Command	Typical Use
Constant	<code>FillConstant()</code>	Use when missing data can be estimated as a constant. For example, if only the early period of a "regulated" (e.g., reservoir) time series is missing, it may be appropriate to set the values to zero.
Monthly total, daily pattern	<code>FillDayTSFrom2MonthTSAnd1DayTS()</code>	Use to estimate a daily time series by applying the pattern of a related daily time series to monthly totals from the related and current time series. For example, use to estimate daily streamflow from monthly total values.
Fill from time series	<code>FillFromTS()</code>	Use non-missing values from a time series to fill missing values in another time series.
Historic Monthly Average	<code>FillHistMonthAverage()</code>	Use with monthly time series to estimate missing monthly values as the average of historic monthly values. For example, if applied to monthly precipitation data, a missing July value would be set to the average of observed July precipitation values (zero is an observation).

TSTool Fill Techniques and Associated Commands (continued)

Technique	Command	Typical Use
Historic Year Average	<code>FillHistYearAverage()</code>	Use with yearly time series to estimate missing data as the average of annual values.
Interpolation	<code>FillInterpolate()</code>	Use to estimate missing data by interpolating between non-missing values. For example, use to estimate reservoir level changes.
Mixed Station	<code>FillMixedStation()</code>	This command tries various combinations of <code>FillRegression()</code> and <code>FillMove2()</code> parameters with time series at different locations, to use the best combination.
Maintenance of Variance	<code>FillMOVE1()</code>	Use to estimate missing data using the Maintenance of Variance Extension (MOVE.1). For example, use to estimate unregulated streamflow from a related gage. This command is currently not enabled.
Maintenance of Variance	<code>FillMOVE2()</code>	Use to estimate missing data using the Maintenance of Variance Extension (MOVE.2). For example, use to estimate unregulated streamflow from a related gage. This approach has been shown to be slightly better than the MOVE.1 approach.
Historic Pattern Averages	<code>FillPattern()</code>	Similar to filling with historic averages with additional complexity of classifying historic months into categories. For example, historic averages for wet, dry, and average periods are computed and used as the historic averages. This command requires that the <code>SetPatternFile()</code> control command also be used.
Prorate	<code>FillProrate()</code>	Fill a time series by prorating known values with another time series.
Regression	<code>FillRegression()</code>	Use to estimate missing data by using ordinary least squares regression. For example, use to estimate streamflow from a related gage.
Repeat	<code>FillRepeat()</code>	Use when it can be assumed that the last observed value before a missing period is a good estimate for missing data. For example, use with "forecasted" data where no future value is available for interpolation.
Using diversion comments	<code>FillUsingDiversionComments()</code>	This command is only available with the HydroBase input type and uses diversion comments and the "not in use" flag to set additional diversion amounts to zero.

Fill commands can be layered (e.g., use `FillRegression()`, then `FillInterpolate()`, then `FillConstant()`). However, the analysis that occurs for each command may be impacted by earlier

fill commands. If necessary, use the `SetFromTS()` command to piece together the results of independent fill commands into a final time series. The **Results...Graph - XY-Scatter** output provides options for selecting different fill techniques and viewing analysis details.

4.5 Set Time Series Data

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

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

Menu_Commands_SetTimeSeries

Commands...Set Time Series Contents Menu

4.6 Manipulate Time Series

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

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

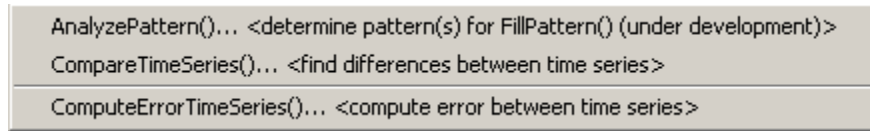
Menu_Commands_Manipulate

Commands...Manipulate Time Series Menu

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

4.7 Analyze Time Series

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

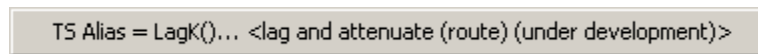


Menu_Commands_AnalyzeTimeSeries

Commands...Analyze Time Series Menu

4.8 Models

The **Commands...Models** menu inserts commands that perform tasks that are more complex than simple data processes. TSTool does not retrieve or save model states, leaving the handling of states to each model.



Menu_Commands_Models

Commands...Models Menu

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

4.9 Output Time Series

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

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

Menu_Commands_OutputTimeSeries

Commands...Output Time Series Menu

Commands that set global configuration values (e.g., output period) are listed at the start of the menu and commands that operate on time series are listed last.

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

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

4.10 Commands for Specific Input Types

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

4.11 Commands for Ensemble Processing

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

```
CreateEnsemble()... <convert 1 time series into an ensemble>
CopyEnsemble()... <create a copy of an ensemble>
ReadNwsrfsEspTraceEnsemble()... <read 1(+) time series from an NWSRFS ESP trace ensemble file>
TS Alias = NewStatisticTimeSeriesFromEnsemble()... <create a time series as a statistic from an ensemble - EXPERIMENTAL>
TS Alias = WeightTraces()... <weight traces to create a new time series>
WriteNWSRFSSESPTraceEnsemble()... <write NWSRFS ESP trace ensemble file>
```

Menu_Commands_EnsembleProcessing

Commands...Ensemble Processing Menu

4.12 Commands for Table Processing

The **Commands...Table Processing** menu provides commands specific to table processing. Tables are defined as row/column data (e.g., from delimited files or databases) where comments can be present in the header and data records, the header defines labels for columns, and columns contain consistent data types (i.e., a column has all dates, or all data values). Table commands are being phased in to support processing such as creating time series lists, rating curve conversions and other transformations, summarizing statistics, and report generation.

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

Menu_Commands_TableProcessing

Commands...Table Processing Menu

4.13 General Commands – Comments

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

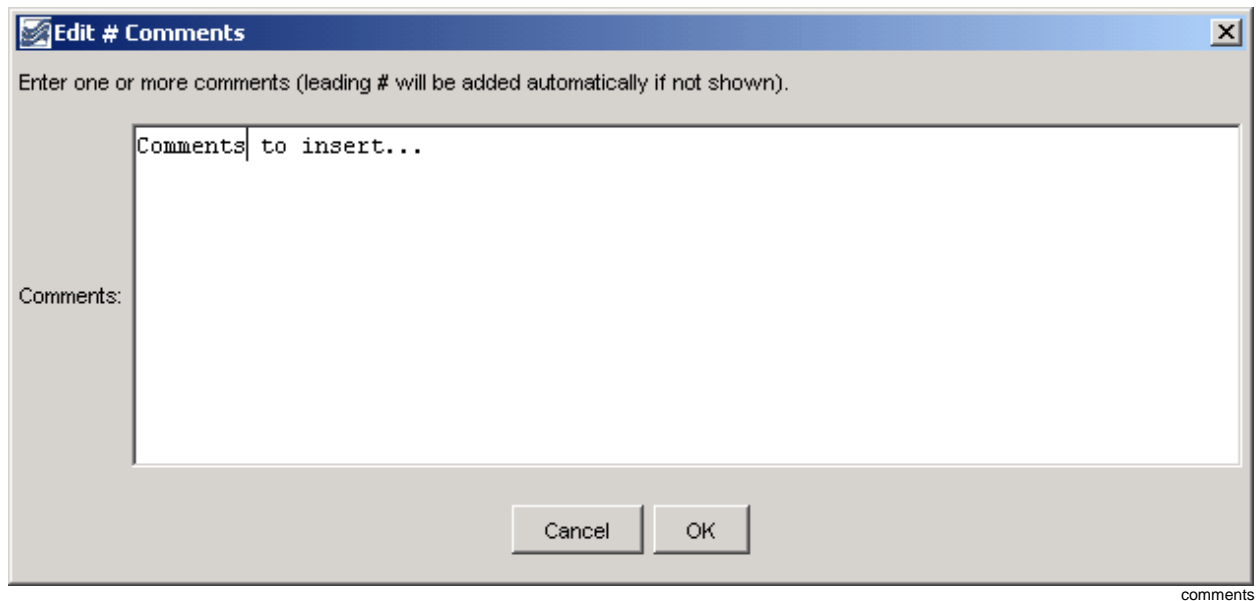
```
# comment(s)...
/* <start comment>
*/ <end comment>
```

Menu_Commands_General_Comments

Commands...General – Comments Menu

4.13.1 Inserting # Comments

Comments are inserted by selecting a line in the **Commands** list and then selecting **Commands...General – Comments...# Comment(s)**. The comments will be inserted before the selected commands. Unlike most other command editors, multiple command lines can be selected. The interface will automatically insert the # character. The following dialog is used to edit comments.

**Comment Editor**

4.13.2 Start /* */ Comments

Multiple commands can be commented using the following notation:

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

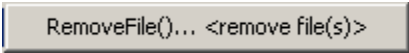
This syntax is consistent with a number of programming languages, including C, C++, and Java, and can be used to quickly disable multiple commands. Use the **Commands...General – Comments.../* <start comment>** menu to start a comment above the selected command. Matching start and end comments should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

4.13.3 End /* */ Comments

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

4.14 General Commands – File Handling

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



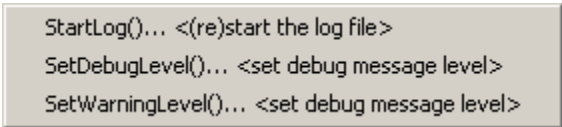
RemoveFile()... <remove file(s)>

Menu_Commands_General_FileHandling

Commands...General – File Handling Menu

4.15 General Commands – Logging

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



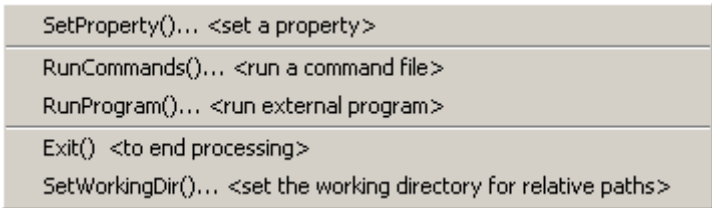
StartLog()... <(re)start the log file>
SetDebugLevel()... <set debug message level>
SetWarningLevel()... <set debug message level>

Menu_Commands_General_Logging

Commands...General – Logging Menu

4.16 General Commands – Running Commands and External Software

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



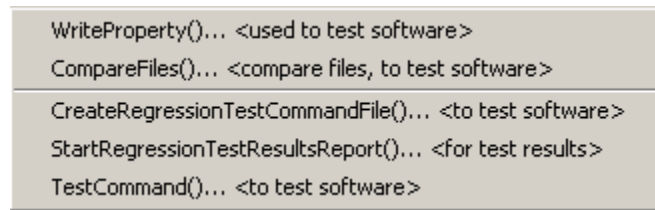
SetProperty()... <set a property>
RunCommands()... <run a command file>
RunProgram()... <run external program>
Exit() <to end processing>
SetWorkingDir()... <set the working directory for relative paths>

Menu_Commands_General_Run

Commands...General – Running Menu

4.17 General Commands – Test Processing

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



Menu_Commands_General_Testing

Commands...General – Test Processing Menu

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

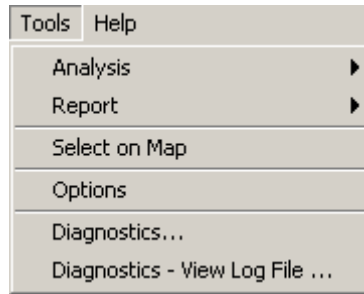
```
#
# Create the regression test runner for the
# TSTool/test/regression/TestSuites/commands_general files.
#
# Only command files that match Test_*.TSTool are included in the output.
# Don't append the generated commands, in order to force the old file to be
# overwritten.
#
CreateRegressionTestCommandFile(SearchFolder="..\..\..\commands\general",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

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

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

This page is intentionally blank.

This chapter discusses the tools available under the **Tools** menu. The **Tools** menu lists tools that perform additional analysis on time series that are selected in the **Time Series Results** list. These features are similar to the **Results** menu features in that a level of additional analysis is performed to produce the data product. Tools may or may not correspond to commands – often tools internally execute the features available in commands, in order to implement a more complicated analysis.

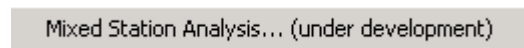


Menu_Tools

Tools Menu

5.1 Analysis Tools

Analysis tools analyze time series and typically produce an output report.



Menu_Tools_Analysis

Tools...Analysis Menu

Currently, only the Mixed Station Analysis tool is available, and it is under development (see the next section).

5.1.1 Mixed Station Analysis

This tool is under development.

The Mixed Station Analysis tool is an interactive tool that tries to find the best combination of time series necessary to fill data using regression or the MOVE2 method. The optimal results can then optionally be used as parameters for the `FillMixedStation()` command. The following figure illustrates the Mixed Station Analysis tool.

Mixed Station Analysis

This tool finds the best fit from independent time series to fill dependent time series.
 The dependent and independent time series can be selected using the TS list parameters:
 AllMatchingTSID - time series selected from the list below (* will analyze all previous time series)
 Right-click on the time series area to select or deselect all. Active only if the TS list selection is "MatchingTSID"
 The working directory is: J:\CDSS\develop\Apps\TSTool\test\Commands\fillRegression

Dependent TS list: AllMatchingTSID How to get the dependent time series to fill.

Dependent time series:
 06759500.USGS.Streamflow.Month
 06754000.DWR.Streamflow.Month

Independent TS list: AllMatchingTSID How to get the independent time series.

Independent time series:
 06759500.USGS.Streamflow.Month
 06754000.DWR.Streamflow.Month

Analysis method(s): OLSRegression

Number of equations: OneEquation

Transformation(s): None

Analysis period: to

Minimum Data Count: Minimum number of overlapping points required for analysis.

Minimum R: Minimum correlation required for a best fit. (default = 0.5)

Best Fit: R

Fill period: to

Intercept: Blank or 0.0 are allowed (Linear 'None' transformation only).

Output file: Browse

Close Analyze View Results Create fill commands Copy commands to TSTool Fill Dependents

Ready

Menu_Tools_MixedStationAnalysis

Mixed Station Analysis Interface

5.2 Report Tools

Report tools analyze time series, typically creating a summary report.

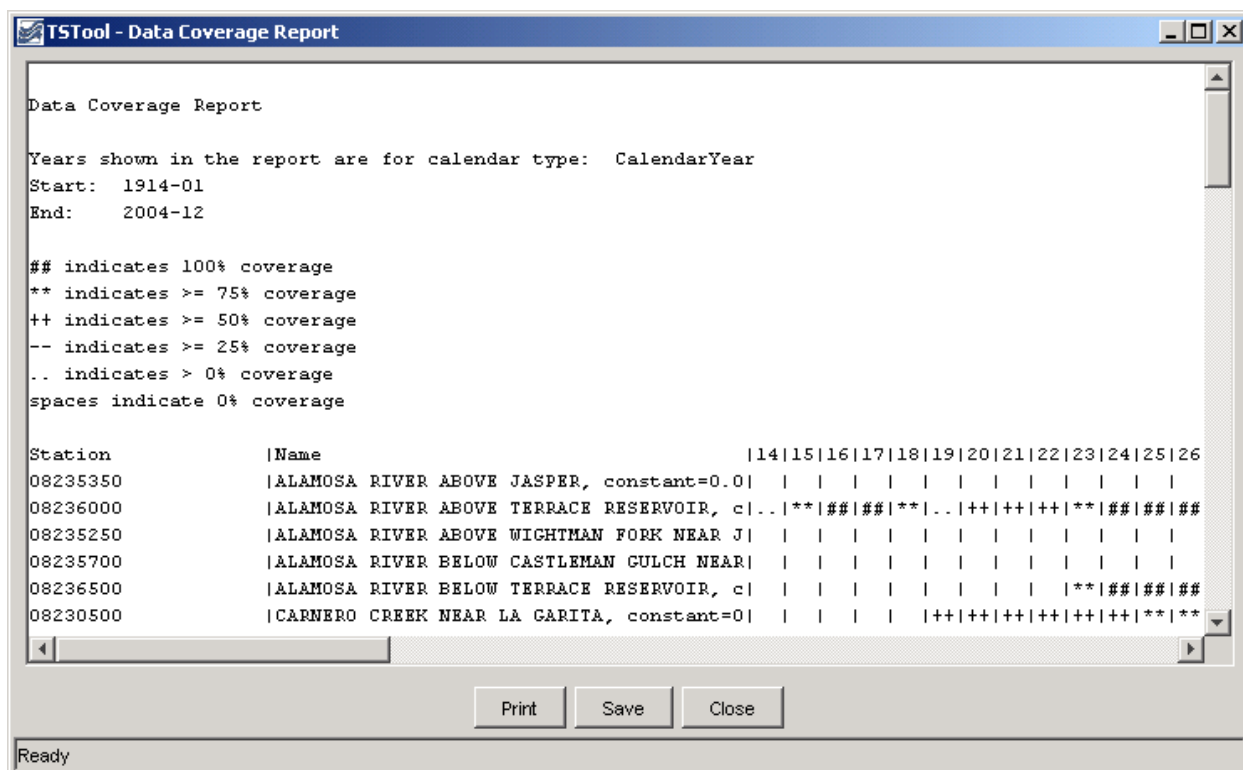
```
Data Coverage by Year...
Data Limits Summary...
Month Summary (Daily Means)...
Month Summary (Daily Totals)...
Year to Date Total... <Daily or real-time CFS Only!>
```

Menu_Results_Report

Tools...Report Menu

5.2.1 Data Coverage by Year Report

The **Tools...Report - Data Coverage by Year** menu processes the time series that have been selected and produces a report similar to the following (abbreviated). This report is useful, in particular, for evaluating data availability for multiple time series over a period. Although effort has been taken to make the report as compact as possible, it will likely need to be printed in landscape format on a large paper size.

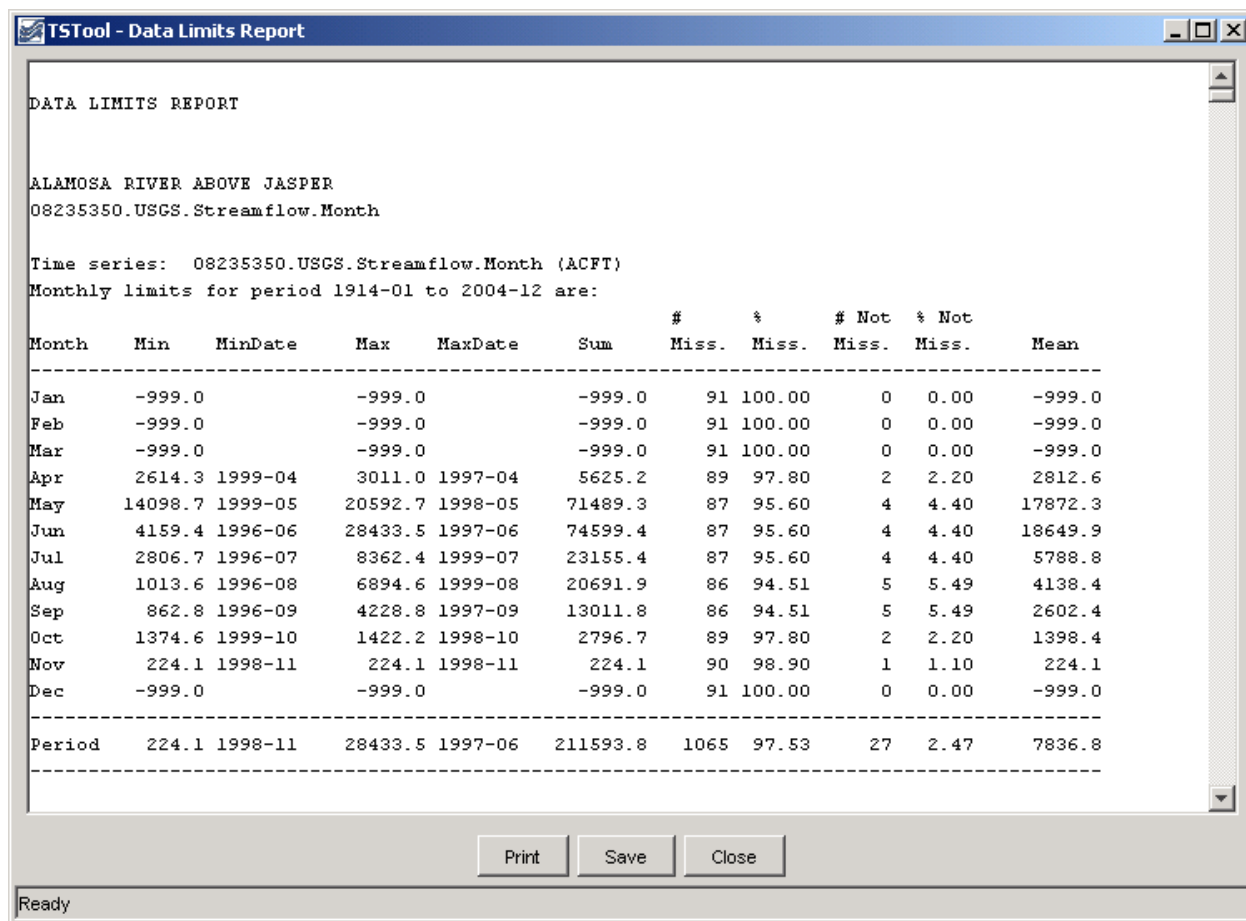


Menu_Results_Report_DataCoverage

Data Coverage by Year Report

5.2.2 Data Limits Summary Report

The **Tools...Report - Data Limits Summary** menu processes the time series that have been selected and produces a report similar to the following (abbreviated). The data limits summary for each time series is included. This report is useful, in particular, for evaluating data availability for specific time series. Currently, only monthly time series have detailed summaries. All other data intervals shown overall period summaries. The value -999 is used to indicate missing data.

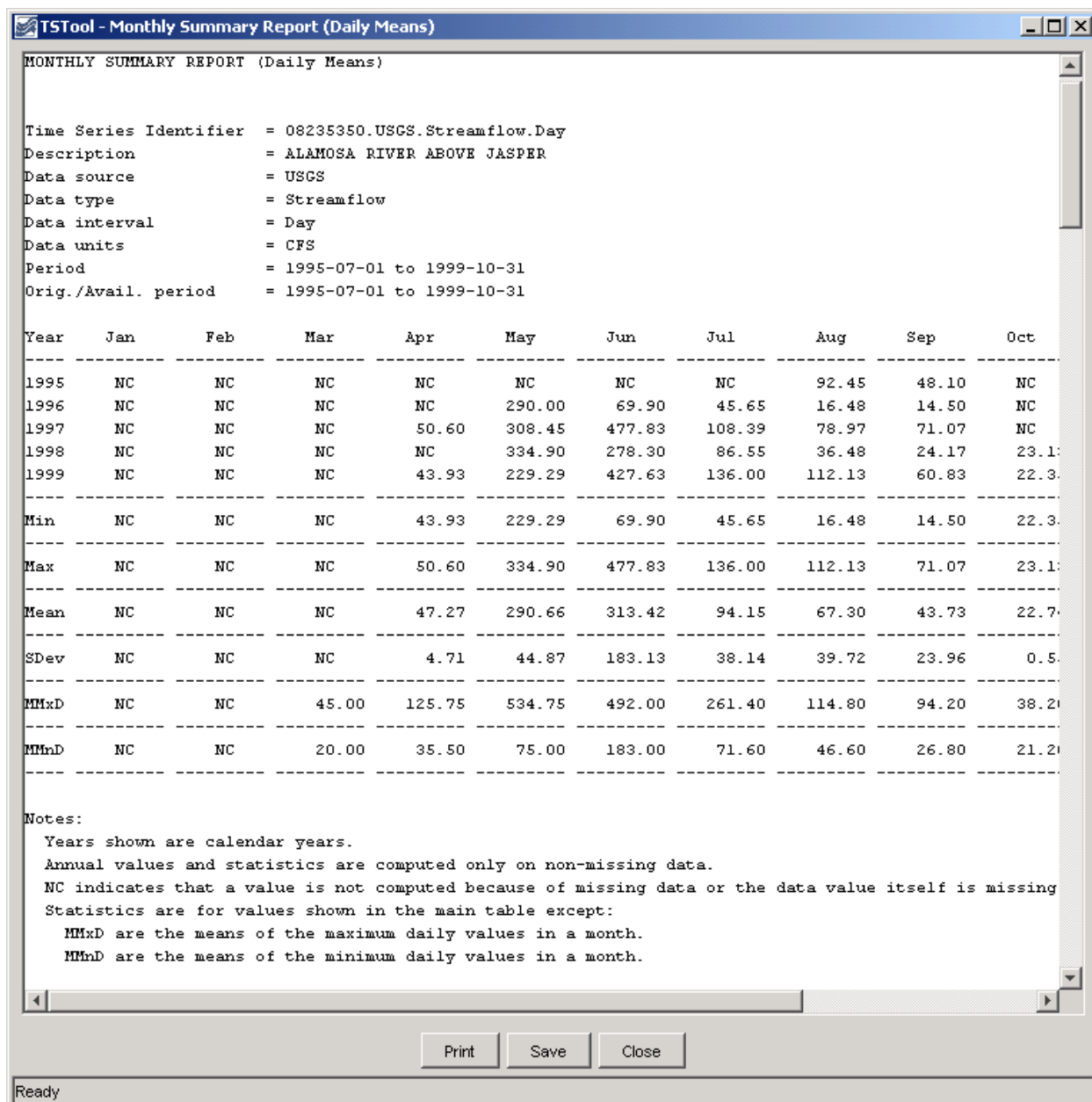


Menu_Report_DataLimits

Data Limits Summary Report

5.2.3 Month Summary Reports

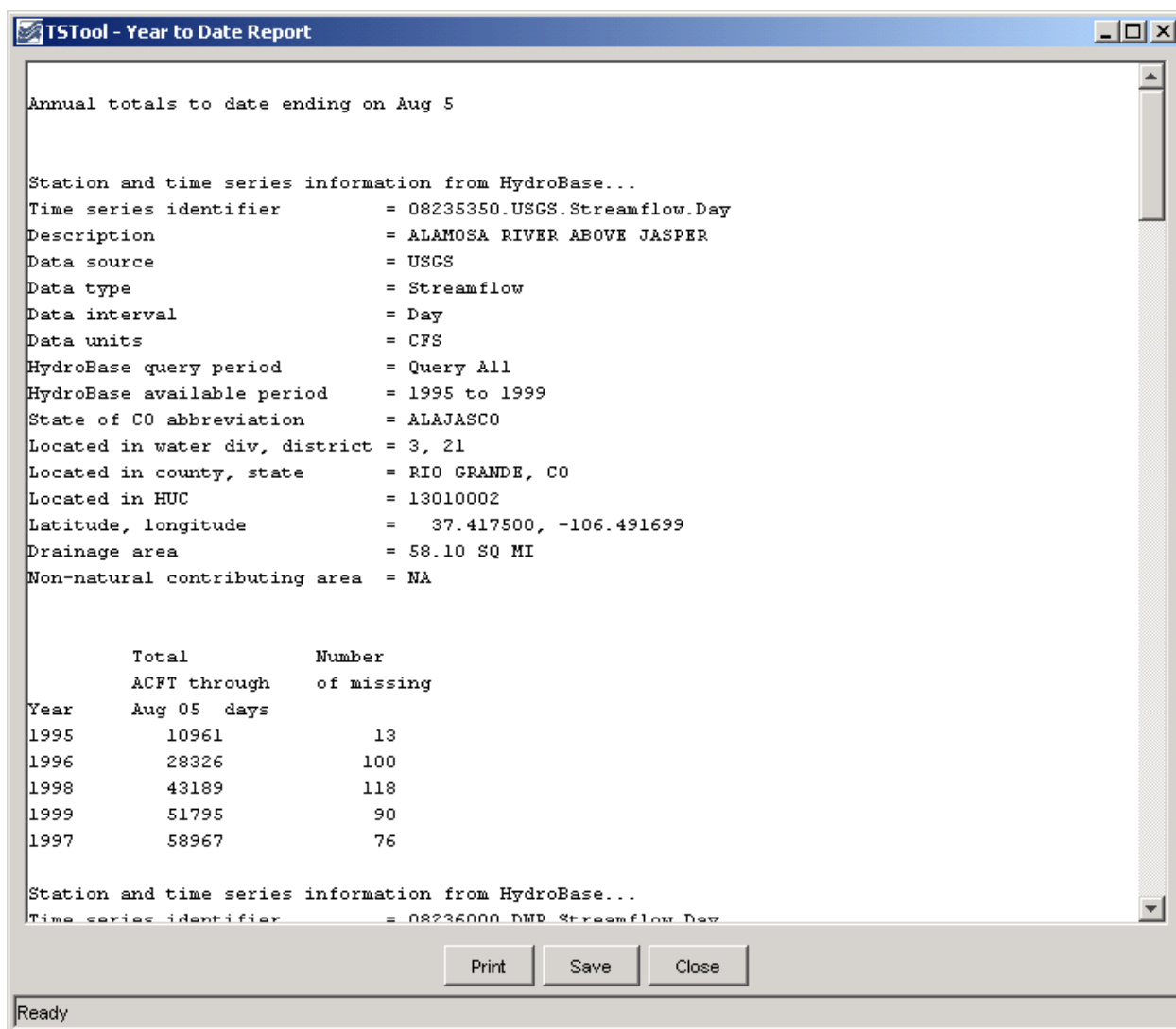
The **Tools...Report - Month Summary** menu process the time series that have been selected and produces a report similar to the following (abbreviated). This report is similar to default summary output for monthly time series; however, it is applied to shorter data intervals, including minute, hour, and day interval. Values are first accumulated to daily values (by averaging the values in a day if the **Daily Means** report is chosen or by totaling the values in a day if the **Daily Totals** report is chosen). For example, use total for precipitation and means for average flows or daily temperature. The daily values are then further accumulated to produce monthly values, again using means or totals. The report includes a header for the time series, the report, and footnotes. Values are only shown if full data are available for a month and statistics are computed using only complete months.



Monthly Summary (Daily Mean) Report

5.2.4 Year to Date Total Report

The **Tools...Report - Year to Date Total** menu processes the time series that have been selected and produces a report similar to the following (abbreviated). This report is useful, in particular, for comparing on a volumetric basis the different years of a time series over a full period. The year-to-date volumes are sorted; to find a particular year, use the **Search** button on the report display. The report information can then be used, for example, to select time series traces for analysis and output. Currently, this report can only be used to process daily CFS data. Real-time data can be analyzed by first converting to a daily interval using the `ChangeInterval()` command. **Warning: some years may have no data at the beginning of a year and the corresponding year-to-date totals will consequently be zero. Refer to the data coverage and data limit reports for more detail.**



Menu_Tools_Report_YearToDateTotal

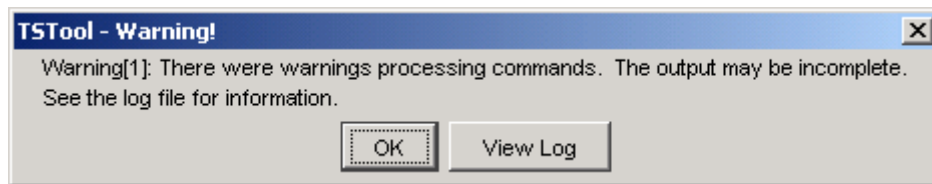
Year to Date Total Report

5.3 Map Tools

The **Tools...Show on Map** button is enabled when a map is displayed (using **View...Map**) and time series are listed in the upper right part of the main window. The locations corresponding to selected time series or all time series in this list can be displayed on the map. See **Chapter 8 – Using the Map** for more information.

5.4 Diagnostics

Diagnostics features are useful for troubleshooting. When an error occurs, a small warning dialog may be displayed, as shown in the following figure:



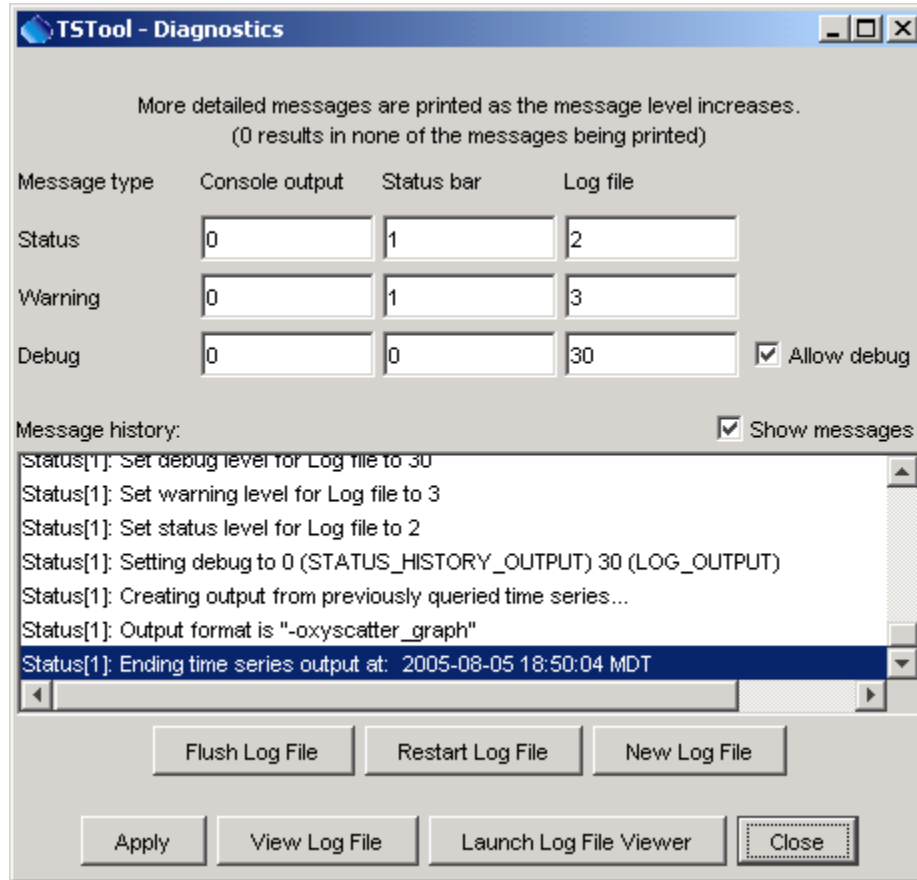
Diagnostics_Warning

Example Warning Message Dialog

If results are not as expected, also review the messages in the status bar at the bottom of the main or secondary windows.

5.4.1 Diagnostics Settings

The **Tools...Diagnostics** menu item displays the **Diagnostics** dialog, which is used to set message levels and view messages as the application runs. The **Diagnostics** dialog (see the following figure) can be used to evaluate a problem.



Diagnostics

Diagnostics Interface

The settings at the top of the dialog are used to specify the level of detail for messages printed to the console window, the status area at the bottom of the main window (and the **Diagnostics** dialog), and the log file. The log file contains warning, status, and debug messages, many of which are not normally displayed in the main interface. The log file is created in the *logs* directory under the installation directory. The **Diagnostics** interface features are as follows:

Status, Warning, Debug

Enter integer values, with larger numbers resulting in more output and slower performance. Zero indicates no output. If troubleshooting, a good guideline is to set the debug level to 10 or 30 (and select the **Allow Debug** checkbox). The default settings are often enough for normal troubleshooting and result in good software performance.

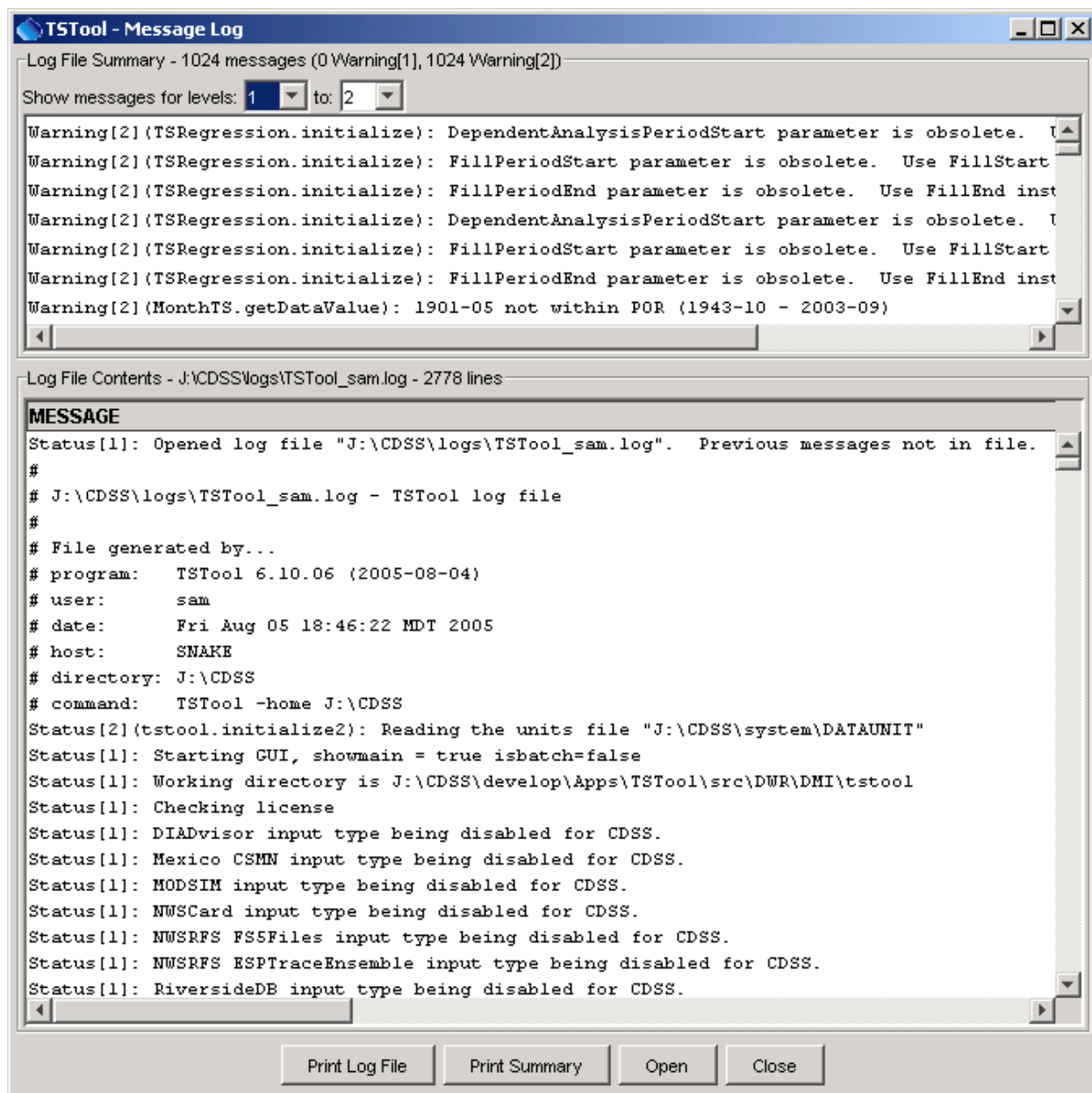
Allow Debug

Select to enable debug messages. Turning on debug messages will significantly slow down the software.

Show Messages	Select to display messages in the Diagnostics window.
Flush Log File	Force messages to be written to the log file. Messages can be buffered in memory and may not otherwise immediately be written to the log file.
Restart Log File	Restart the log file. This is useful if a long session has occurred and troubleshooting will occur on new actions.
New Log File	Open a new log file, with a new name.
Apply	Apply the settings in the Diagnostics dialog.
View Log File	View the log file in an integrated window. The View Log File button will be enabled if the log file has been opened.
Launch Log File Viewer	View the log file using a viewer from the operating system. On Windows computers, Notepad will be used.
Close	Apply the settings in the Diagnostics dialog and close the window.

5.4.2 Diagnostics – View Log File

The **Tools...Diagnostics – View Log File** menu item displays the integrated log file viewer. Selecting this menu item is equivalent to selecting the **View Log File** button in the **Diagnostics** dialog. The log file viewer will be displayed in a window as shown in the following figure:



Diagnostics_ViewLog

Log File Viewer Window

The log file messages can be scrolled. To find a string in the log file, right-click and select the **Find** menu item. The information in the log file can also be copied and pasted into email, when contacting support.

6 Examples of Use

Version 08.15.00, 2008-05-01

This chapter provides examples for reading and manipulating time series data using TSTool. The heading for the section gives an indication of the example purpose. Where appropriate, input and output types are indicated to help users find an appropriate example. General examples are listed first, followed by more complex examples.

6.1 General Examples

6.1.1 General – One-time Time Series Display/Analysis

6.1.2 General – Reproducing an Analysis with a Command File

6.2 Model Data Processing Examples

6.2.1 Modeling – Preparing Model Files Using a Command File

6.2.2 Modeling – Processing End of Month Reservoir Data (Input=HydroBase, Output=StateMod)

6.2.3 Modeling – Filling Reservoir Targets with a Pattern File (Input=HydroBase, Output=StateMod)

6.2.4 Modeling – Using a List File to Automate Time Series Processing (Input=HydroBase, Output=StateMod)

6.2.5 Modeling – Processing Frost Dates (Input=HydroBase, Output=StateCU)

6.2.6 Modeling – Filling Streamflow using MOVE2 (Input=HydroBase)

6.3 Time Series Trace Examples

6.3.1 Time Series Traces – Comparing Historical and Current Conditions (Input=HydroBase)

6.4 Time Series Product Examples

6.4.1 Time Series Product – Using TSTool to Display Graphs from Another Software Application

6.4.2 Time Series Product – Automating Graphs for Compare Observed and Simulated Time Series

6.1 General Examples

This section includes examples related to general TSTool use, which may be appropriate for general users.

6.1.1 General – One-time Time Series Display/Analysis

The following example session illustrates how to query time series data for display, analysis, and viewing.

1. Start TSTool. If the HydroBase or other database input types are enabled, select a database (see **Section 3.2- Select HydroBase Dialog**, for example).
2. To manipulate time series in any way, first select the time series of interest (see **Section 3.3 - Main Interface**). Pick appropriate input type, data type, and filter information. Press **Get Time Series List** to list the available time series. After pressing **Get Time Series List**, a list of time series will be shown in the upper-right corner of the interface.
3. Select one or more time series from the list and transfer to the **Commands** list as time series identifiers. Time series identifiers are explained in **Chapter 2 – Introduction**.
4. Press the **Run All Commands** button to query the time series. They should now be listed in the **Results** area.

5. Use the **Results** and **Tools** menus to view the time series or the **File...Save...Time Series As** menus to export as files. For example, display a line graph (using **Results...Graph - Line**) and then view the time series as a summary or table.
6. Go back to the **Commands** list and use the **Commands** menu to manipulate time series. For example:
 - Insert a `FillInterpolate()` command to fill data
 - Insert a `Cumulate()` or `RunningAverage()` command to transform the data
7. Repeat steps 4 – 5 to process and view time series.

6.1.2 General – Reproducing an Analysis with a Command File

To reproduce an analysis, save the commands shown in the **Commands** list to a command file and then reload and run the commands later. For example, assuming that steps similar to the previous section have been executed:

1. Use the **File...Save...Commands As** menu to save the commands. It is recommended that command files be saved with a file extension `.TSTool`.
2. Exit TSTool and restart (alternatively, clear the commands using the **Clear Commands** button).
3. Use **File...Open...Command File**. Select the file that you previously saved.
4. Then run the commands by pressing the **Run All Commands** button. Display the results using the **Results** menu.

As the above example shows, reproducing an analysis consists of saving a command file that can be reused later. The main complications in this approach are that the environment in which the commands are run may change over time. For example if using a HydroBase database, the database version, ODBC data source name, database host, or working directory may be differ between computers. It is recommended that commands use directories relative to a working directory (the folder where the command file is saved) and that the working directory is defined consistently on different computers that will use the commands. Using paths relative to the working directory will consequently allow command files to be portable. TSTool will internally set the working directory that the directory where a command file is opened or saved.

6.2 Model Data Processing Examples

Most computer models require data that adhere to a consistent format. TSTool facilitates processing model data files with features that:

- Allow a specific period of record to be output
- Fill missing data
- Produce time series in a specific order

The following examples illustrate how to use TSTool to process model data.

6.2.1 Modeling – Preparing Model Files Using a Command File

To prepare model files, multiple commands will usually process numerous time series. Modelers often run TSTool in batch mode from a command shell using a command like:

```
tstool -commands commandfile
```

However, it is recommended that command files be run using the GUI, if possible, in order to take advantage of additional error-checking and feedback features.

TSTool provides command editor dialogs for every command and helps ensure the integrity of commands by searching for input time series for each command. An effort has been made to make the current TSTool recognize and process old commands. However, there have been some changes that will require updates to commands. It is recommended that old command files be migrated to the new syntax using the following approaches:

1. Review the release notes appendix when installing software updates.
2. Be familiar with this TSTool User Manual and, in particular, the command reference.
3. Run an existing command file and review the log file for warnings about commands that need to be updated. Then edit the commands in the GUI (see the next step).
4. Open an existing command file using the **File...Open...Command File** menu . TSTool will attempt to convert commands to new syntax as the file is loaded. Some older commands will not be updated until they are edited in the command editors. Most command files focus on a particular data type and filling technique. Therefore most updates will generally involve only a few changes.

A number of commands have been added/enhanced to promote reuse of command files in both batch and GUI run modes. For example, the `OpenHydroBase()` indicates whether the command is active for batch and GUI run modes. Choosing the correct setting simplifies exchange of command files between users and operating environments.

When querying time series, select a subset of the commands for intermediate work to verify filling or other manipulation. General commands (e.g., `SetOutputPeriod()`) may be required even if a subset of time series is being processed, for example, to ensure that periods overlap.

TSTool by default reads all available data. However, the `SetInputPeriod()` is available to limit the period that is read. The `SetOutputPeriod()` is used to control the period for output products.

6.2.2 Modeling – Processing End of Month Reservoir Data (Input=HydroBase, Output=StateMod)

The following example illustrates how to create a monthly reservoir target file for StateMod using data from HydroBase. Note however that end of month data may not always be available in HydroBase due to data availability and quality control issues. If the data are not available in HydroBase, time series can be read from files.

```
# Reservoir target file commands
# Each reservoir needs a minimum (zero) and maximum time series (from HydroBase)
SetOutputPeriod(OutputStart="10/1974",OutputEnd="9/1991")
setOutputYearType(Water)
# CBT SHADOW MTN GRAND L
TS ShadowMtn = NewTimeSeries(NewTSID="513695.USBR.ResEOM.Month",Description="CBT SHADOW MTN
GRAND L",Units="AF",InitialValue=0)
513695.USBR.ResEOM.MONTH.
# CBT GRANBY RESERVOIR
TS Granby = NewTimeSeries(NewTSID="51460.USBR.ResEOM.Month",Units="AF",InitialValue=0)
514620.USBR.ResEOM.MONTH.
# DILLON RESERVOIR
TS Dillon = NewTimeSeries(NewTSID="364512.USBR.ResEOM.Month",Units="AF",InitialValue=0)
364512.DWB.ResEOM.MONTH.
# GREEN MOUNTAIN RESERVIOIR
TS GreenMtn = NewTimeSeries(NewTSID="363543.USBR.ResEOM.Month",Units="AF",InitialValue=0)
363543.USBR.ResEOM.MONTH.
# RIFLE GAP RESERVOIR
TS RifleGap = NewTimeSeries(NewTSID="393508.USBR.ResEOM.Month",Units="AF",InitialValue=0)
393508.USBR.ResEOM.MONTH.
WriteStateMod(TSList=AllTS,OutputFile="coloup.tar")
```

ExamplesOfUse/Reservoir_EOM/Example_Reservoir_EOM.TSTool

6.2.3 Modeling – Filling Reservoir Targets with a Pattern File (Input=HydroBase, Output=StateMod)

The following example illustrates a command file for creating a StateMod reservoir target file, using pattern filling.

```
# eom.commands.TSTool
#
# commands in this file either pull historical EOM contents from the CRDSS database
# (i.e. Rifle Gap) or from user-defined *.stm files
#
setOutputPeriod(10/1908,09/2005)
setOutputYearType(Water)
setPatternFile("../Diversions\fill2005.pat")
#
# GREEN MOUNTAIN RESERVOIR
363543...MONTH~StateMod~363543.stm
#
# UPPER BLUE RESERVOIR (ConHoosier)
# Data from HydroBase is used to better represent actual operations of the reservoir in the
cm2005 update
# rather than setting the contents to its maximum as in previous model versions.
363570.DWR.ResMeasStorage.Day~HydroBase
TS ConHoosier363570 = newEndOfMonthTSFromDayTS(363570.DWR.ResMeasStorage.Day,16)
free(TSID="363570.DWR.ResMeasStorage.Day")
fillPattern(TSList=LastMatchingTSID,TSID="ConHoosier363570",PatternID="09037500")
setConstant(TSID="ConHoosier363570",ConstantValue=0,SetEnd="03/1962")
fillInterpolate(ConHoosier363570,0,Linear)
#
# CLINTON GULCH RESERVOIR
# Data from HydroBase is used to better represent actual operations of the reservoir in the
cm2005 update
# rather than setting the contents to its maximum as in previous model versions.
363575.DWR.ResMeasStorage.Day~HydroBase
TS ClintonGulch363575 = newEndOfMonthTSFromDayTS(363575.DWR.ResMeasStorage.Day,16)
free(TSID="363575.DWR.ResMeasStorage.Day")
fillInterpolate(ClintonGulch363575,0,Linear,10/1992,9/2004)
fillPattern(TSList=LastMatchingTSID,TSID="ClintonGulch363575",PatternID="09037500")
setConstant(TSID="ClintonGulch363575",ConstantValue=0,SetEnd="03/1977")
fillInterpolate(ClintonGulch363575,0,Linear)
#
# DILLON RESERVOIR
364512...MONTH~StateMod~364512.stm
#
# WOLCOTT RESERVOIR
373639...MONTH~StateMod~zero.stm
... similar commands for other reservoirs omitted...
#
# Fill remaining missing data with historical averages
fillHistMonthAverage(TSList=AllTS)
#
writeStateMod(TSList=AllTS,OutputFile="../statemod\cm2005.eom",Precision=0)
```

From Colorado_1_2007 CDSS data set

6.2.4 Modeling – Using a List File to Automate Time Series Processing (Input=HydroBase, Output=StateMod)

It may be desirable to read a file containing a list of station/structure identifiers and process the corresponding time series. The following example illustrates a command file to use a list to read time series from HydroBase and output a StateMod data file.

```
#
# Example to illustrate how a delimited list of location identifiers can be used
# to create time series identifiers for processing. This example creates
# time series identifiers to read from the State of Colorado's HydroBase, and
# outputs to a StateMod model file.
#
CreateFromList(ListFile="structure_list.txt",IDCol=1,DataSource=DWR,
DataType="DivTotal",Interval=Month,InputType=HydroBase,
HandleMissingTShow=IgnoreMissingTS)
SetOutputYearType(Calendar)
WriteStateMod(TSList=AllTS,OutputFile="structure_list.stm")
```

ExamplesOfUse/CreateFromList/Example_CreateFromList

where the list file contains the following:

```
#
# Structures for which to process data
#
# WDID - State of Colorado Water District Identifier
# Name - Structure name (from HydroBase)
#
"WDID", "Name"
0100501,EMPIRE DITCH
0100503,RIVERSIDE CANAL
0100504,ILLINOIS DITCH
```

6.2.5 Modeling – Processing Frost Dates (Input=HydroBase, Output=StateCU)

Frost dates are special time series consisting of four dates per year. The dates correspond to:

- Last day in spring that the temperature was 28° F
- Last day in spring that the temperature was 32° F
- First day in fall that the temperature was 32° F
- First day in fall that the temperature was 28° F

These specific dates are currently consistent with the HydroBase and StateCU input types. Older versions of TSTool (before version 06.00.00) treated frost dates as a single time series, where the four components were internally manipulated as dates. The Add () command had a limited number of features supported manipulating frost date time series. However, other commands could not be used to process the time series. Consequently, display, analysis, and manipulation capabilities were limited.

As of TSTool version 06.00.00, TSTool handles each of the above data items as separate data types and time series, internally treating the values as Julian days from January 1. The StateCU input type, which is used when reading and writing frost date files, converts between Julian days and Month/Year in the file.

By handling as four separate numerical time series, all of TSTool's manipulation tools can be used to fill and analyze frost dates. This does require each time series to be specified, whereas before the four were internally handled with a single time series identifier. Because frost dates are internally treated as numerical Julian days, using the generic numerical `Add()` command functionality may result in unexpected output if the time series overlap (Julian days will be added). To avoid this situation, use the `FillFromTS()`, `SetFromTS()`, or `Blend()` commands when merging multiple time series. The following example illustrates how to process a frost dates file for StateCU:

```
SetOutputPeriod(OutputStart="1950",OutputEnd="2002")
#
# 0130 - ALAMOS SAN LUIS VALLEY RGNL
0130.NOAA.FrostDateL28S.Year~HydroBase
0130.NOAA.FrostDateL32S.Year~HydroBase
0130.NOAA.FrostDateF32F.Year~HydroBase
0130.NOAA.FrostDateF28F.Year~HydroBase
# Add Meeker Stations (5484 and 5487)
# then "free" 5487
5484.NOAA.FrostDateL28S.Year~HydroBase
5484.NOAA.FrostDateL32S.Year~HydroBase
5484.NOAA.FrostDateF32F.Year~HydroBase
5484.NOAA.FrostDateF28F.Year~HydroBase
5487.NOAA.FrostDateL28S.Year~HydroBase
5487.NOAA.FrostDateL32S.Year~HydroBase
5487.NOAA.FrostDateF32F.Year~HydroBase
5487.NOAA.FrostDateF28F.Year~HydroBase
FillFromTS(TSList=AllMatchingTSID,TSID="5484.NOAA.FrostDateL28S.Year",
IndependentTSList=AllMatchingTSID,IndependentTSID="5487.NOAA.FrostDateL28S.Year")
FillFromTS(TSList=AllMatchingTSID,TSID="5484.NOAA.FrostDateL32S.Year",
IndependentTSList=AllMatchingTSID,IndependentTSID="5487.NOAA.FrostDateL32S.Year")
FillFromTS(TSList=AllMatchingTSID,TSID="5484.NOAA.FrostDateF32F.Year",
IndependentTSList=AllMatchingTSID,IndependentTSID="5487.NOAA.FrostDateF32F.Year")
FillFromTS(TSList=AllMatchingTSID,TSID="5484.NOAA.FrostDateF28F.Year",
IndependentTSList=AllMatchingTSID,IndependentTSID="5487.NOAA.FrostDateF28F.Year")
Free(TSID="5487*")
#
#
FillHistYearAverage(TSList=AllMatchingTSID,TSID="*")
#
#
writeStateCU("../StateCU\Frost2002.stm")
```

ExamplesOfUse\FrostDates\Example_FrostDates.TSTool

6.2.6 Modeling – Filling Streamflow Using MOVE2 (Input=HydroBase)

Data filling is an important activity for modeling. TSTool provides a number of data filling commands, as described in **Section 4.4 – Fill Time Series Data**. Data filling can be accomplished using varying levels of complexity. The approach used for data filling depends on the data type and interval. For example, estimating daily precipitation may be difficult because relationships between daily precipitation time series may not exist. TSTool provides tools for data filling but it does not automatically pick the most appropriate fill methods (see the `FillMixedStation()` command and **Tools...Analysis...Mixed Station Analysis** for help automating filling). Data filling involves a number of steps:

1. Initial review of the data (e.g., using the **Results...Report - Data Coverage by Year** menu, and graphs).
2. Review of the spatial proximity of gages using the TSTool **View...Map Interface** capability or GIS software.

3. Comparison of candidate time series (e.g., using the **Results...Graph - XY-Scatter** menu).
4. Apply data filling commands.
5. Review final results visually and review time series histories (by right-clicking on a time series in the **Time Series Results** list and selecting **Time Series Properties**).

The data filling approach can be simple or complex. An example of a complex data filling technique is to use the `FillMOVE2()` command on daily streamflow data. In particular, consider the following case:

- Time series 1 (TS1) has a long period of gaged unregulated data (e.g., a headwater): 1900 to 2000
- Time series 2 (TS2) has a shorter period of gaged data with 1920 to 1950 being unregulated and 1950 to 2000 being regulated (e.g., due to the construction of a reservoir)
- The goal is to produce an estimate of unregulated flow for TS2 for the full period 1900 to 2000.

This can be accomplished using the following commands:

```
#
# Data filling example - assume daily DateValue time series as input
#
#
# Generate some sample data for the example described above:
# ts1 has observed values from 1900-2000
# ts2 has observed values from 1920 to 1950 and regulated thereafter
#
# Although data are generated below, they could be read from files or a
# database. In this case, the SetOutputPeriod() command might need to be used
# to ensure that the final result is for a required period.
#
TS ts1 = NewPatternTimeSeries(NewTSID="ts1..Streamflow.Day",Description="Time series
1, all unregulated",SetStart="1900-01-01",SetEnd="2000-12-
31",Units="CFS",PatternValues="1,2,4,7,12,6,2,1.5")
TS ts2 = NewPatternTimeSeries(NewTSID="ts2..Streamflow.Day",Description="Time series
1, unregulated from 1920 to 1950, regulated after",SetStart="1900-01-
01",SetEnd="2000-12-31",Units="CFS",PatternValues="2,3,6,10,15,3,2.5,2")
#
# Clear out the period 1919- in ts2 because it was not recorded in our example.
SetConstant(TSList=AllMatchingTSID,TSID="ts2",ConstantValue=-999,SetEnd="1919-12-31")
# Clear out the period 1951+ in ts2 because it is regulated and needs to
# be filled with the result of unregulated MOVE2 analysis.
SetConstant(TSList=AllMatchingTSID,TSID="ts2",ConstantValue=-999,SetStart="1951-01-
01")
# Analyze and fill the second time series. Transform the data to log10 and
# use monthly equations...
FillMOVE2(TSID="ts2",IndependentTSID="ts1",NumberOfEquations=MonthlyEquations,Depende
ntAnalysisStart="1920-01-01",DependentAnalysisEnd="1950-12-
31",IndependentAnalysisStart="1900-01-01",IndependentAnalysisEnd="2000-12-
31",FillFlag="M")
```

ExamplesOfUse/Filling/Example_Filling.TSTool

The above example illustrates a somewhat complicated situation where data filling is facilitated by the features of the `FillMOVE2()` command. If the `FillMOVE2()` command is not appropriate, then the `FillRegression()` or other commands can be applied. In some cases, it may be appropriate to fill different parts of the period using different independent time series. A simpler approach may involve only a single filling step (e.g., fill the entire period using a single `FillRegression()` command).

6.3 Time Series Ensemble Examples

The general term *time series ensemble* refers to groups of a time series, often shown in overlapping fashion. Common ways to create ensembles are:

- Split time series into N-year lengths and shift to overlap.
- Run a model multiple times with different input, in order to generate many possible outcomes.
- Generate synthetic data to use as input to a model.

Several TSTool commands have been implemented to create and process time series traces. Many other commands allow ensembles to be specified to provide a list of time series for processing. TSTool manages ensemble data by using an ensemble identifier and optionally using a trace (sequence) number for each time series, which when processing historical data is typically the starting historical year for the trace. For most functionality, the ensemble is simply a collection of the time series traces in the ensemble.

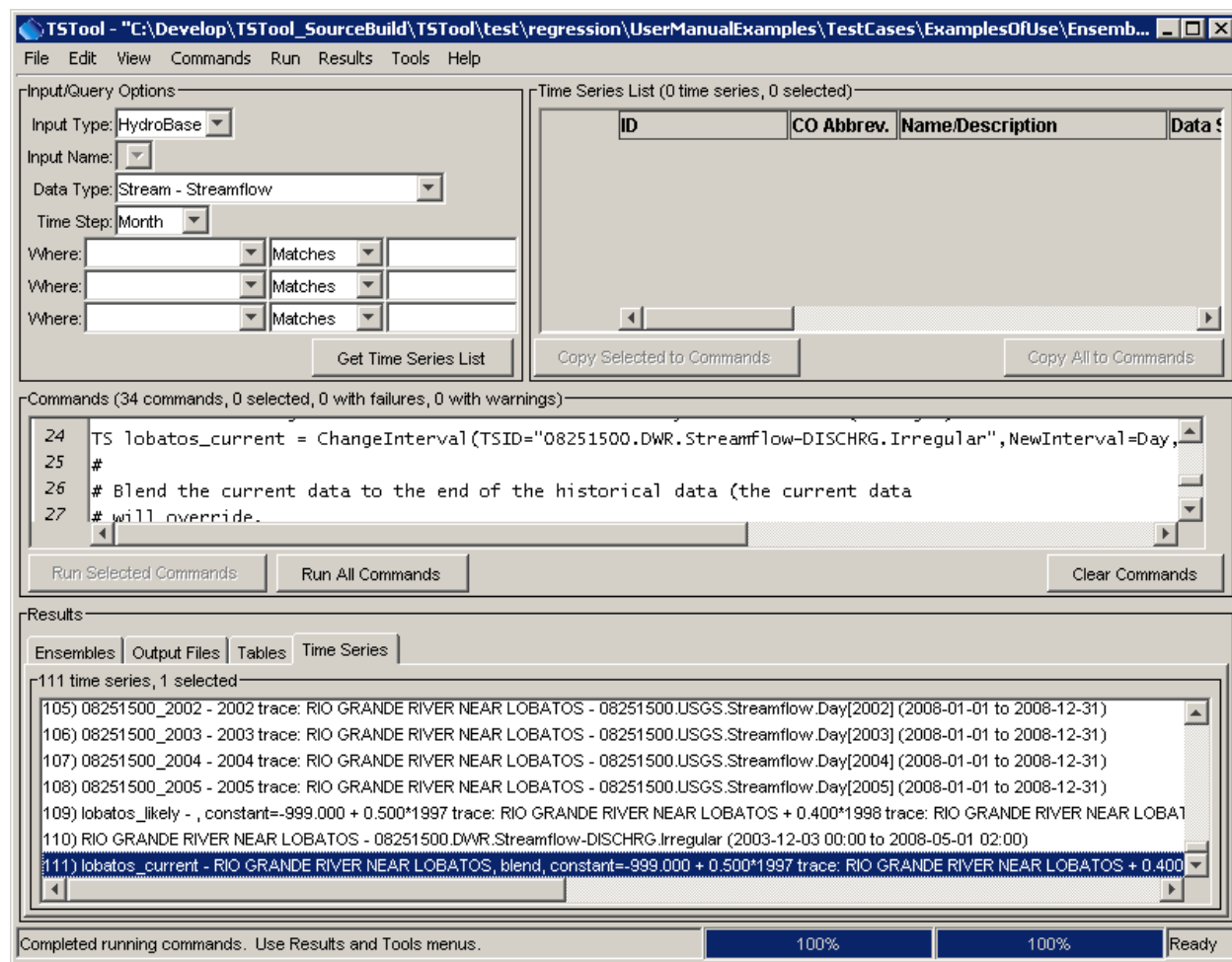
6.3.1 Time Series Traces – Comparing Historical and Current Conditions (Input=HydroBase)

The following command file illustrates how historical time series traces can be plotted on top of real-time data. The features illustrated in the example were implemented to help determine an estimate of future flow based on current conditions.

```
# StartLog(LogFile="Example_Ensemble.TSTool.log")
# These commands query historical and real-time data at the lobatos gage and
# compute a weighted "best-guess" for the flows at lobatos for the remainder
# of the current year. In other words, the current year will be complete,
# with observed at the beginning and historical "likely" at the end.
#
# First get the historic daily lobatos gage and convert to traces. Shift each trace to
# 2008-01-01 (the current year) so that the data can overlay the current year's
# values.
TS daily = readTimeSeries("08251500.USGS.Streamflow.Day~HydroBase")
CreateEnsemble(TSID="daily",TraceLength=1Year,EnsembleID="Lobatos",ReferenceDate="2008-01-
01",ShiftDataHow=ShiftToReference)
#
# Now weight the traces using representative historic years.
#
TS lobatos_likely =
WeightTraces(EnsembleID="Lobatos",SpecifyWeightsHow="AbsoluteWeights",Weights="1997,.5,1998,.4,
1999,.1",NewTSID="Lobatos..Streamflow.Day.likely")
#
# Now query the current (real-time) flows. HydroBase may only hold a few weeks or months
# of data.
#
# Uncomment the following line to see the actual real-time values
# 08251500 - RIO GRANDE RIVER NEAR LOBATOS
08251500.DWR.Streamflow-DISCHRG.Irregular~HydroBase
# Convert the irregular instantaneous values to a daily instantaneous (midnight)
TS lobatos_current = ChangeInterval(TSID="08251500.DWR.Streamflow-
DISCHRG.Irregular",NewInterval=Day,OldTimeScale=INST,NewTimeScale=INST)
#
# Blend the current data to the end of the historical data (the current data
# will override.
#
Blend(TSID="lobatos_current",IndependentTSID="lobatos_likely",BlendMethod=BlendAtEnd)
#
# After the above commands are executed, time series in memory will include the traces and
# the current time series. You can select only the time series of interest and plot
# OR select many time series and then disable/enable in the plot. You may need to use
# both approaches to find appropriate time series to weight.
```

ExamplesOfUse/Ensemble_Realtime+WeightedHistorical/Example_Realtime+WeightedHistorical

The results of processing the above commands in TSTool are a list of the traces, a weighted time series (based on three traces), and the current daily data, all at the same streamflow gage, as shown in the following figure.



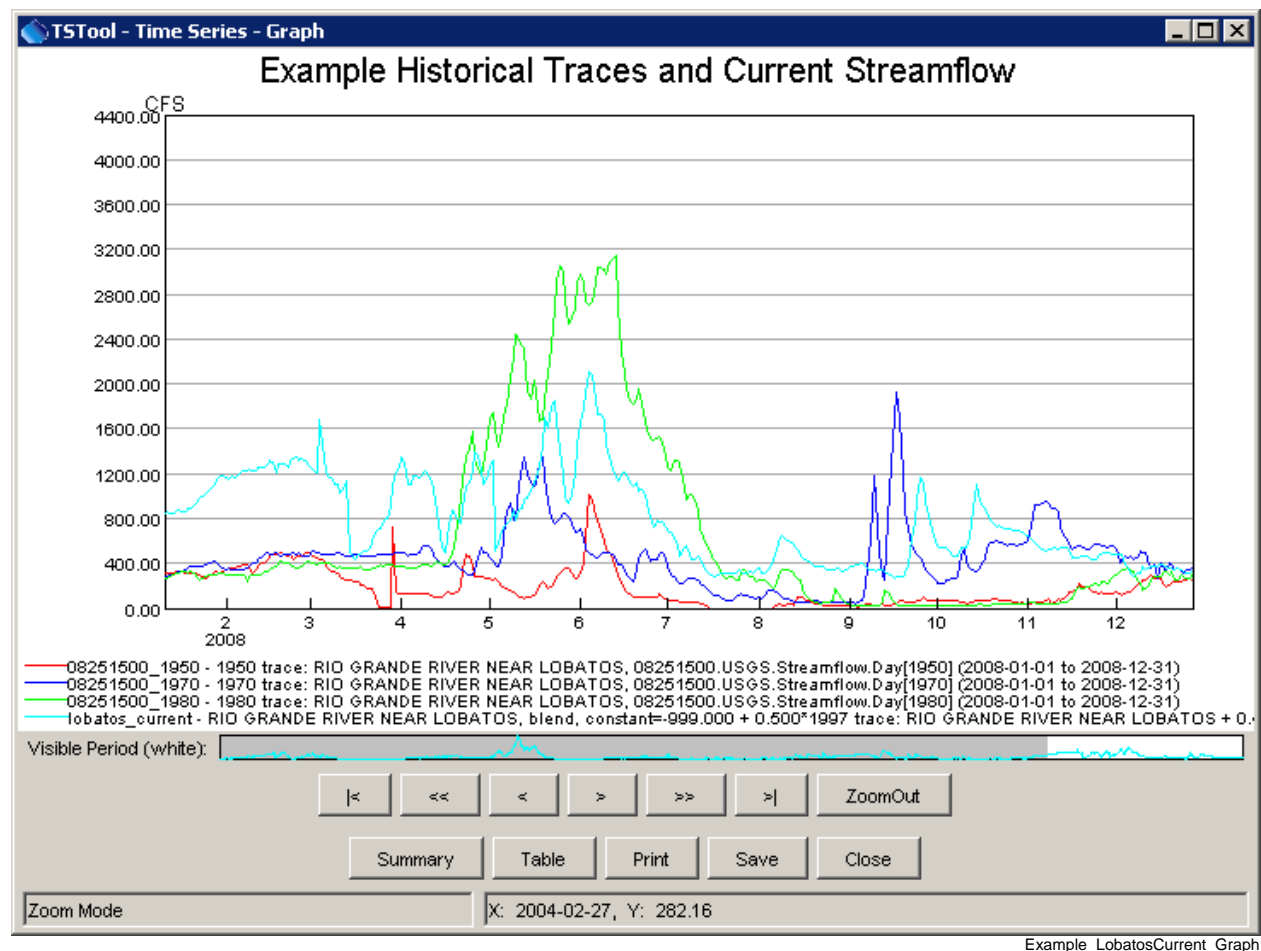
Example_LobatosCurrent_TSTool

Any of the time series can be selected and viewed.

The following features are useful for selecting appropriate traces:

1. Convert a time series to an ensemble and then plot all the traces. Use the graph properties to turn traces on/off or use symbols to identify different traces.
2. Use the tools described in **Chapter 5 - Tools** to evaluate time series and traces. For example, the **Year to Date** report can be used to determine how well different years compare volumetrically. The `NewStatisticYearTS()` and `NewStatisticTimeSeriesFromEnsemble()` commands create derived time series that are useful for evaluating ensemble time series.

Key traces and output time series can be selected and graphed, as shown below.



Example Graph of Traces and Combined Real-time/Historical Time Series

6.4 Time Series Product Examples

Time series products are described in the **TSTool Time Series Viewing Tools** appendix. In summary, a time series product file can be generated that uses time series identifiers to indicate data to be processed, and includes other properties (e.g., titles) to configure a graph. TSTool can process time series products in a number of ways, as illustrated by the following examples.

6.4.1 Time Series Product - Using TSTool to Display Graphs from an Application

TSTool is primarily used as an interactive tool or to process command files in batch mode. However, it is also possible to run TSTool with a command file, no main graphical user interface, and still display only specific graphs. For example, TSTool can be called from an application to display a graph by reading data from a recognized database or file format. This takes advantage of TSTool's features rather than adding additional features to the application. The following example illustrates how to display a graph of precipitation and streamflow data in a single graph, using data from the State of Colorado's HydroBase database. TSTool should be started using a command line similar to:

```
tstool -commands example.tstool -nomaingui
```

Additionally, the HydroBase database to be used should be configured in the TSTool configuration file (see the **Installation and Configuration Appendix**). In this run mode the main GUI is never made visible. Because the interactive main interface is disabled, the normal HydroBase login dialog is not shown; therefore, the HydroBase information must be defined in the configuration information.

Although it is possible to display several graphs at the same time, it is currently assumed that only one graph will be shown. Closing the graph will close TSTool. The command file can be complex but in many cases will be simple because an application is calling TSTool to display a single graph. The following example shows a typical command file for this run mode:

```
# Example command file to run TSTool without showing the main GUI
# but showing a plot to the screen. When the plot window closes, TSTool will
# exit without prompting. TSTool should be called using:
#
# TSTool -commands ThisFile -nomaingui
#
# This is useful for displaying plots from applications that only need to use
# TSTool in a supporting role.
#
# Process a time series product description file and display a plot window
# to the screen.
ProcessTSProduct(TSProductFile="test.tsp",RunMode=GUIAndBatch,View=True)
```


The `ProcessTSProduct()` command references a time series product file. An example of the file is as follows (see the **TSView Time Series Viewing Tools Appendix** for a full description of time series product file properties):

```
[Product]

ProductType = "Graph"
TotalHeight = "400"
TotalWidth = "600"

[SubProduct 1]

GraphType = "Bar"
MainTitleString = "Precipitation"
BarPosition = "CenteredOnDate"

[Data 1.1]

Color = "Blue"
TSID = "7337.NOAA.Precip.Month~HydroBase"

[SubProduct 2]

GraphType = "Line"
MainTitleString = "Streamflow"

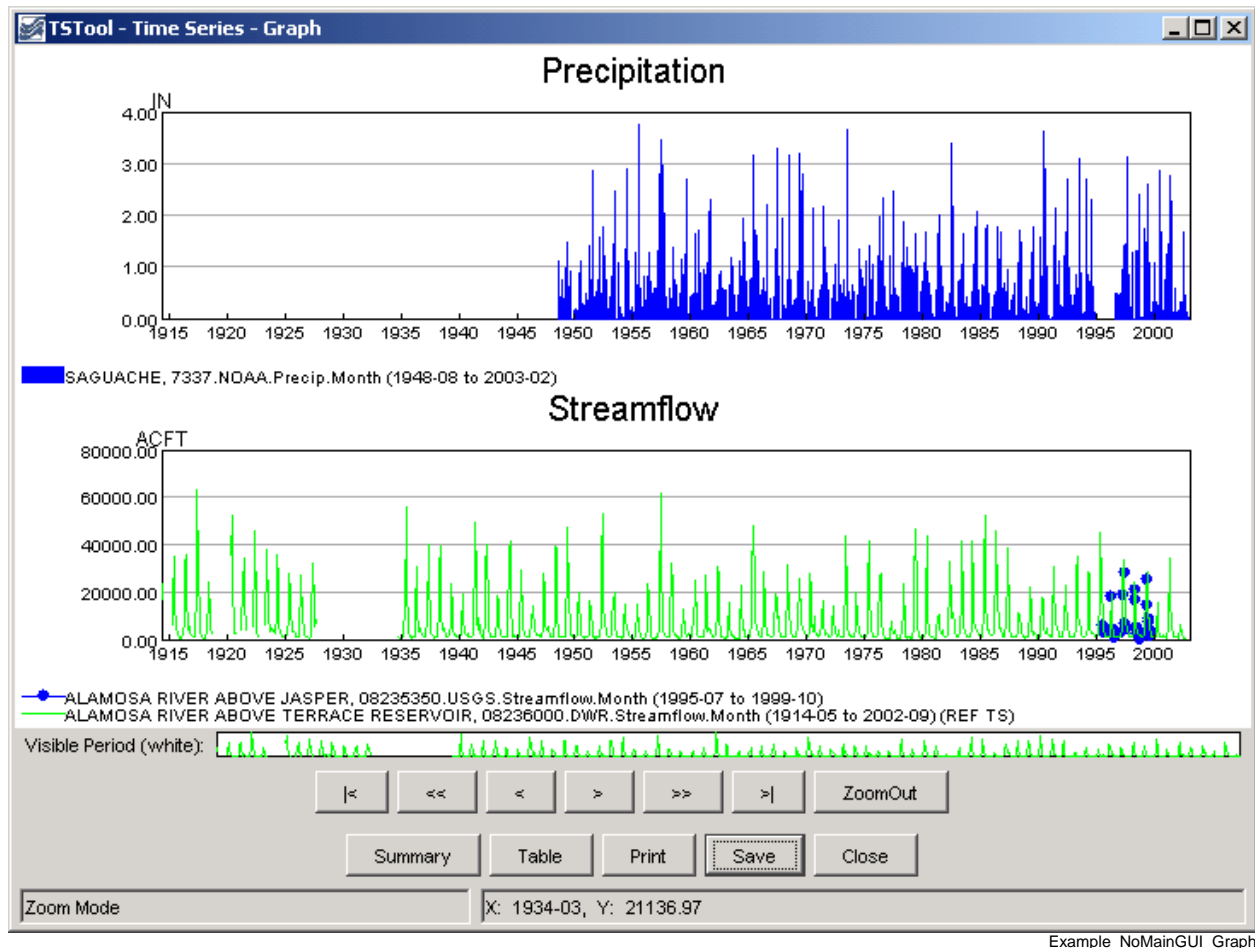
[Data 2.1]

SymbolSize = "7"
SymbolStyle = "Circle-Filled"
TSID = "08235350.USGS.Streamflow.Month~HydroBase"

[Data 2.2]

TSID = "08236000.DWR.Streamflow.Month~HydroBase"
```

The resulting graph is shown in the following figure. Pressing **Close** will exit TSTool.



Example_NoMainGUI_Graph

6.4.2 Automating Graphs to Compare Observed and Simulated Time Series

It is often useful to automate comparison of observed and simulated time series (e.g., during model calibration). For example, after making an adjustment to a model during calibration, many comparisons may be made to evaluate the changes. TSTool can help with comparisons. For example, consider the simulated and observed time series stored in a DateValue file (*results.dv*), as follows (in this example the DateValue file was created by reading StateMod model input and output, and the TSID and DataType lines were hand-edited in the DateValue file to facilitate this example).

```
# DateValueTS 1.3 file
# File generated by...
# program:   TSTool 6.08.02 (2004-07-27) Java
# user:      sam
# date:      Thu Jul 29 09:17:35 MDT 2004
# host:      host unknown
# directory: J:\CDSS\develop\Apps\TSTool\test\Commands\createTraces
# command:   TSTool -home C:\CDSS
#-----
#
# Commands used to generate output:
#
# 09152500...MONTH~StateMod~J:\CDSS\Data\SWSI_Gunnison\StateMod\gunnv.rih
# 09152500.StateMod.River_Outflow.Month~StateModB~J:\CDSS\Data\SWSI_Gunnison\StateMod\gunnv.b43
#
Delimiter   = " "
NumTS       = 2
TSID        = "09152500..StreamFlow_Observed.MONTH" "09152500..Streamflow_Simulated.Month"
Alias       = " " " "
Description = "09152500" "Gunn R. NR GrandJ" "_FLO"
DataType   = "Streamflow_Observed" "Streamflow_Simulated"
Units       = "ACFT" "ACFT"
MissingVal  = -999.0000 -999.0000
Start       = 1908-10
End         = 2001-09
#
# Time series comments/histories:
#
#
# Creation history for time series 1 TSID=09152500...MONTH Alias=):
#
# Read StateMod TS for 1908-10 to 2001-09 from "J:\CDSS\DataSets\SWSI_Gunnison\StateMod\gunnv.rih"
#
# Creation history for time series 2 TSID=09152500.StateMod.River_Outflow.Month Alias=):
#
#   Read from "J:\CDSS\DataSets\SWSI_Gunnison\StateMod\gunnv.b43 for 1908-10 to 2000-09
#
#EndHeader
Date "09152500...MONTH, ACFT" "09152500.StateMod.River_Outflow.Month, ACFT"
1908-10 -999.0000 82035.8828
. . . omitted . . .
1916-10 61488.5000 95692.6641
1916-11 56529.8000 97254.9297
1916-12 55339.6000 94700.1563
1917-01 52265.2000 47388.2305
1917-02 49984.2000 42303.5938
1917-03 79935.0000 64748.8125
. . . similar to end of file . . .
```

These time series can be read into TSTool, an XY graph produced, and the time series product saved (*results.tsp*), as shown in the following example. The original TSID properties have been inserted as comments, corresponding to the original data. The absolute paths to the time series files have also been replaced with relative paths, assuming that the command file to process the product is in the same folder as the product file.

```
[Product]

ProductType = "Graph"
TotalWidth = "600"
TotalHeight = "400"
MainTitleString = "Streamflow Gage 09152500"
SubTitleString = "Comparison of Observed and Simulated"

[SubProduct 1]

GraphType = "XY-Scatter"
XYScatterMethod = "OLSRegression"
LegendFormat = "Auto"
MainTitleString = ""

[Data 1.1]

#TSID = "09152500...MONTH~StateMod~gunnv.rih"
TSID = "09152500..Streamflow_Observed.MONTH~DateValue~results.dv"

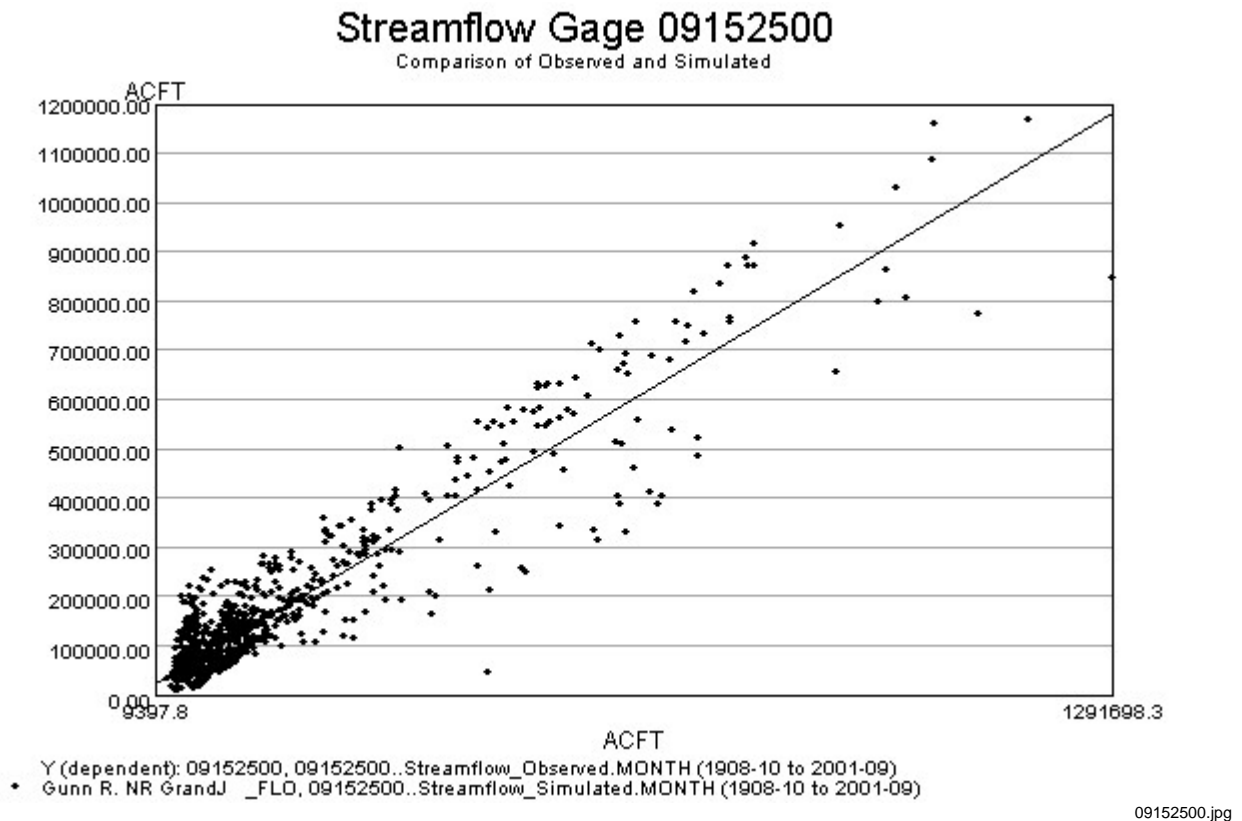
[Data 1.2]

#TSID = "09152500.StateMod.River_Outflow.Month~gunnv.b43"
TSID = "09152500..Streamflow_Simulated.MONTH~DateValue~results.dv"
```

Finally, a command file (*results.TSTool*) can be created that processes the time series and time series product file:

```
ProcessTSProduct(TSProductFile="results.tsp",RunMode=GUIAndBatch,
View=True,OutputFile="09152500.png")
```

The above commands can be run from the TSTool GUI or in batch mode to produce the following graph (note in this example that the x-axis data values are so large that the software is having difficulty finding good labels):



Because this approach relies primarily on the time series identifiers to associate the time series data with the time series product, it is important to establish a concise and clean identifier scheme. The `TSAlias` property can also be used in product files and will take precedence over `TSID` properties. Using time series aliases can improve the readability of command files.

Once a working example is established, the example can be scaled up to a larger production either by repeating the example (and changing identifiers) or by automatically changing the example to replace strings. The latter is not currently part of TSTool; however, TSTool can call external programs (see the `RunProgram()` and `RunPython()` commands), which can supplement the existing TSTool features. Additional features are being added to TSTool to facilitate product generation.

This page is intentionally blank.

Although the main focus in TSTool is time series, many time series are associated with a location such as a station, area, or sensor. This chapter discusses the relationship between time series and spatial data and provides an overview of using map-related features in TSTool. Time series concepts (such as time series identifiers) are discussed in detail in **Chapter 2 – Introduction**. Information about the map display tool used in TSTool is provided in the **GeoView Mapping Tools Appendix**.

7.1 Time Series and Map Layer Relationships

An example is useful to provide an overview of the relationship between time series and map layers.

Map layers often indicate physical features (e.g., rivers, cities, roads, data collection stations) or features that are overlaid on physical features (e.g., political boundaries, weather fronts, regions or points of interest). A layer's data consist of:

1. Features – the coordinate information that defines the shape that is drawn.
2. Attributes – a tabular list of data values associated with the features.
3. Metadata – data about the layer, including the source, coordinate system, history, etc.
4. Projection – the coordinate system for the coordinates, which is usually noted in metadata but may also be indicated by a projection file or similar.
5. Symbolology – to be visualized, a layer must be associated with a symbol (e.g., point symbol, line width, polygon fill color), labels, and other visualization information.

The features and attributes are the primary data, and the other information facilitates using the features and attributes.

Consider a data collection station, represented by a point on the map. This station may be located near a river and collect streamflow stage (water depth). The station software may convert the stage to flow or may allow this to be done by software. The station may also collect “climate” (meteorological) data such as precipitation, temperature, wind speed and direction, etc. Each measurement type requires a sensor and the cost of hardware typically controls the number and sensitivity of measuring devices. For data management, the station is typically assigned an alphanumeric station identifier and each data type that is collected is assigned an alphanumeric data type. The data are saved locally as date/value information and are then transmitted to or requested from a data collecting system. The date/value information is essentially time series. Data units and handling of missing data are considered during implementation of data collection systems. Measurements may be taken regularly (e.g., once every fifteen minutes) or may occur at irregular intervals, perhaps in response to some change in conditions. In nomenclature used with TSTool, the former are called regular time series and the latter irregular, reflected by the data interval (time step).

For the discussion purposes, consider only a meteorological station that measures precipitation and temperature. For mapping purposes, a choice may be made to focus on the physical nature of the map, in which case a single “Met Stations” (or “Climate Stations”) layer may be shown, using a single symbol. This is suitable if the measurement types for such a station are consistent throughout a system or only one data type is of interest. It is frequently the case that the real-time data that are collected are managed in a database, with data being archived over time, for example resulting in the following time series for precipitation data:

1. Real-time data (often provisional data available for a short period).
2. Real-time extended data – real-time data collected for the past year, for example, having received limited or no quality control
3. Real-time archived data – real-time data for the historical period, quality controlled
4. Hourly accumulation – for example, convert real-time precipitation data to hourly totals
5. Daily accumulation – for example, convert hourly precipitation data to daily totals
6. Monthly accumulation – etc.
7. Yearly accumulation – etc.

The first two examples are often referred to as “real-time” data whereas the last five examples are often referred to as “historical” data. In a system that is homogeneous, a map layer that shows “Precipitation Stations” will imply that all of the above time series are available at the station. However, in a system where, for various reasons, not all stations have real-time and historical data, more attention to detail may be needed on maps.

To address this case, the map could show separate layers for real-time and historical stations (two layers). However, this does not address the issue that there may be multiple categories of real-time data and multiple categories of historical data. To address this issue, additional layers might be added for each time series type, using the same or similar symbols for each layer. The limitation in this approach is that the map now has many layers and many of the points will be the same and will therefore symbols will plot on top of each other.

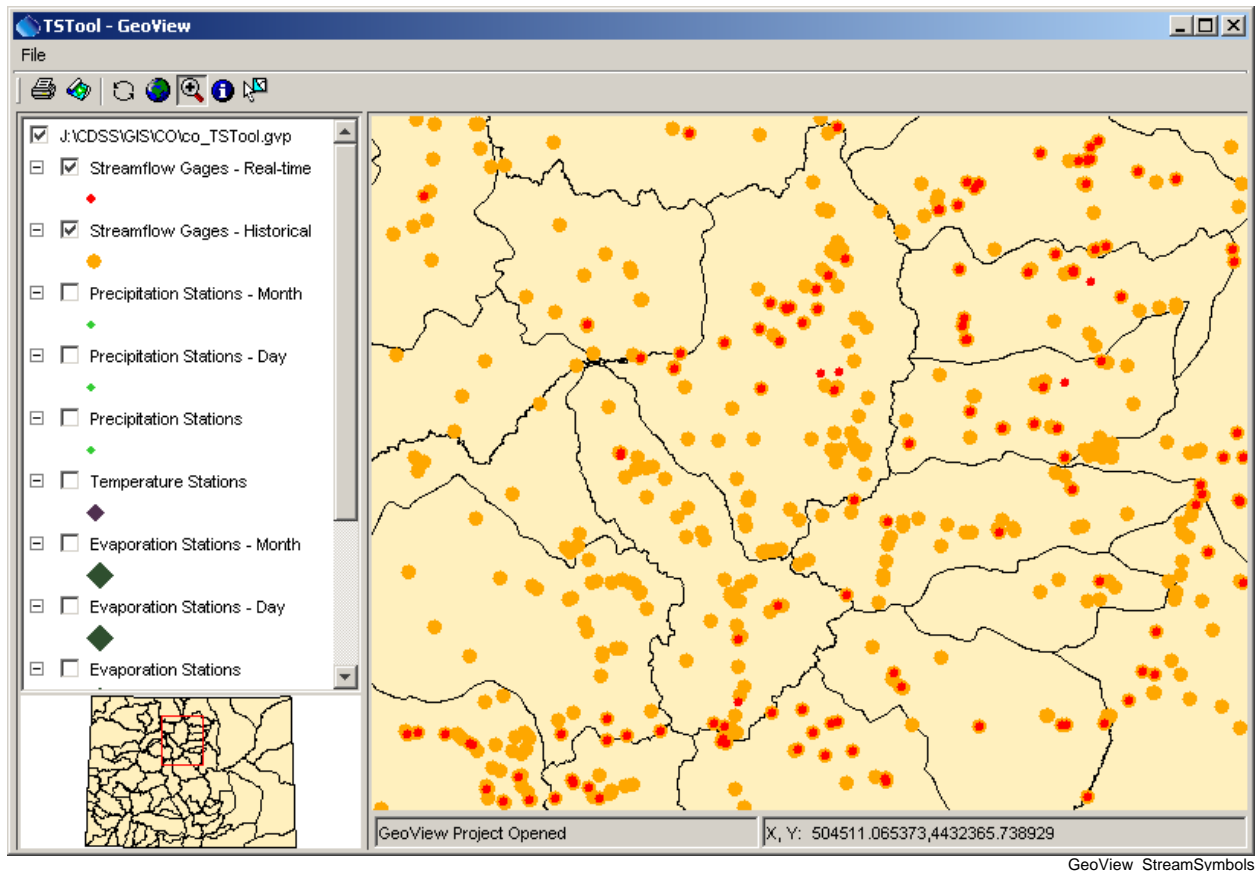
Layered symbols can be used to help with visualization. For example, use the symbol size and shape, and configure the order of layers to ensure that multiple symbols drawing on each other will still allow sufficient visibility of the symbols. This can be applied to indicate data types collected at a location, and also data type/interval information. The following example shows this approach to indicate station type type:



Example of Symbol Layering

SymbolExample

Using many layers and symbols to indicate time series data interval may not be appropriate if the symbols are to be used for classification. For example, the symbol size or color may indicate a physical characteristic of the feature. For this reason and because maps usually focus on physical features, using symbols to indicate the various data intervals is not common. More common are maps that show a layer for real-time data and a layer for historical data, as shown in the following figure:



Example Map Showing Real-time and Historical Data Layers

The map is useful because the user and software can determine where real-time time series **SHOULD** be available, and where historical data **SHOULD** be available. The mapping tools can be integrated with application software by hard-coding the data type and interval for real-time and historical data or use a configuration file (e.g., to indicate that “historical” data should always have an interval of Month).

TSTool is a generic tool and therefore configuration information is required to make the link between the map layers and time series. The approach that has been taken is to rely on a delimited lookup file, which allows users to determine at what level to categorize map layers. More specific information allows a more specific link (i.e., a time series in TSTool can be matched with a specific feature in a map layer) while less information results in a looser link (i.e., several time series in TSTool may match one station and selecting the station may result in more than one time series).

The following example illustrates the time series to map layer configuration file (use of this file in a system is described in the **Installation and Configuration Appendix**):

```
# This file allows time series in TSTool to be linked to stations in spatial
# data layers. The columns are used as appropriate, depending on the direction
# of the select (from time series list or from the map).
#
# This file has been tested with the \CDSS\GIS\CO\co_TSTool.gvp file. Not all
# possible combinations of time series and map layers have been defined - only
# enough to illustrate the configuration.
# Additional attributes need to be added to the point files to allow more
# extensive functionality. For example, if attributes for data interval (time
# step) and data source are added to the attributes, then a definition query
# can be defined on the layer for displays to use the same data file. The
# configuration below can then use the different names to configure the link
# to time series.
#
# This file is discussed in the TSTool documentation.
#
# TS_InputType - the time series input type, as used in TSTool
# TS_DataType - the data type shown in TSTool, specific to an input type
#               For example, TSTool uses "Streamflow" for HydroBase, whereas
#               for other input types a different data type string may be used.
# TS_Interval - time series interval of interest (e.g., "Month", "Day", "1Hour"
#               "Irregular")
# Layer_Name - the layer name used in the map layer list
# Layer_Location - the attribute that is used to identify a location, to be
#                  matched against the time series data location
# Layer_DataType - the attribute that is used to indicate the data type for a
#                  station's time series (CURRENTLY NOT USED - UNDER EVALUATION)
# Layer_Interval - the attribute that is used to indicate the interval for a
#                  station's time series
# Layer_DataSource - the attribute that is used to indicate the data source for
#                    a station's time series.
#
# When matching time series in the TSTool time series query list with features
# on the map, the TS_* values are matched with the time series identifier
# values and the Layer_* attributes are matched against specific time series.
#
# Data layers are listed from largest interval to smallest.
"TS_InputType", "TS_DataType", "TS_Interval", "Layer_Name", "Layer_Location", "Layer_DataSource"
HydroBase, DivTotal, Day, "Diversions", id_label_7, ""
HydroBase, DivTotal, Month, "Diversions", id_label_7, ""
HydroBase, EvapPan, Day, "Evaporation Stations", station_id, ""
HydroBase, EvapPan, Month, "Evaporation Stations", station_id, ""
HydroBase, Precip, Irregular, "Precipitation Stations", station_id, ""
HydroBase, Precip, Day, "Precipitation Stations", station_id, ""
HydroBase, Precip, Month, "Precipitation Stations", station_id, ""
HydroBase, RelTotal, Day, "Reservoirs", id_label_7, ""
HydroBase, RelTotal, Month, "Reservoirs", id_label_7, ""
HydroBase, Streamflow-DISCHRG, Irregular, "Streamflow Gages - Real-time", station_id, ""
HydroBase, Streamflow, Day, "Streamflow Gages - Historical", station_id, ""
HydroBase, Streamflow, Month, "Streamflow Gages - Historical", station_id, ""
```

Example Time Series to Map Layer Lookup File

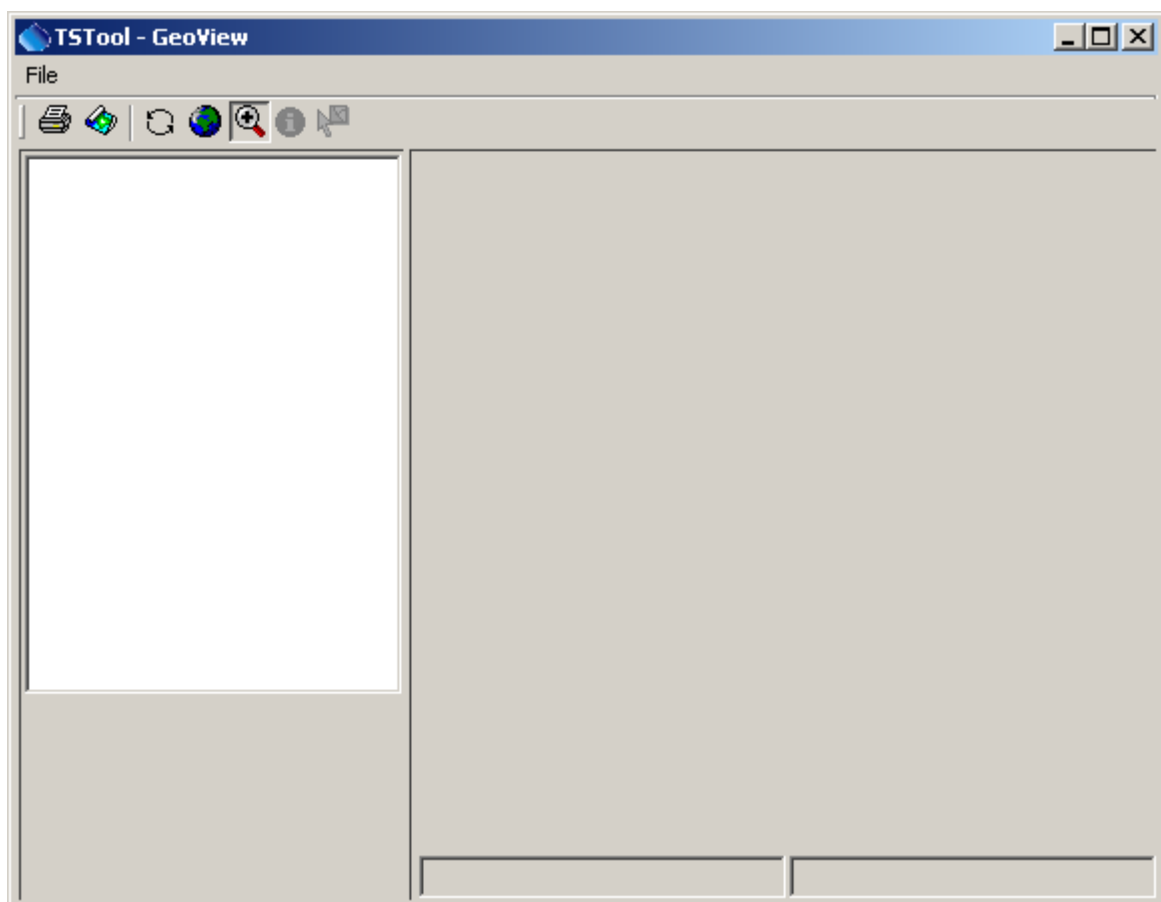
The intent of the file is to allow lookup of map layers from time series and to allow lookup of time series from map layers. The ability to perform these actions depends on the number of layers in the map and the attributes in map layers. For example, in the last two lines of the above example, historical streamflow time series are both linked to a single “Streamflow Gages – Historical” layer on the map. Consequently,

it will not be possible from the map layer to indicate to TSTool that specifically day or month interval data are requested (both daily and monthly time series would be selected). This behavior may be appropriate, or more specific layers and attributes may be needed. Multiple maps are typically needed, in order to meet all the various needs of users and therefore maps with more detail may need to be configured for use with TSTool.

The following sections describe more specifically how to utilize the links between time series and map layers.

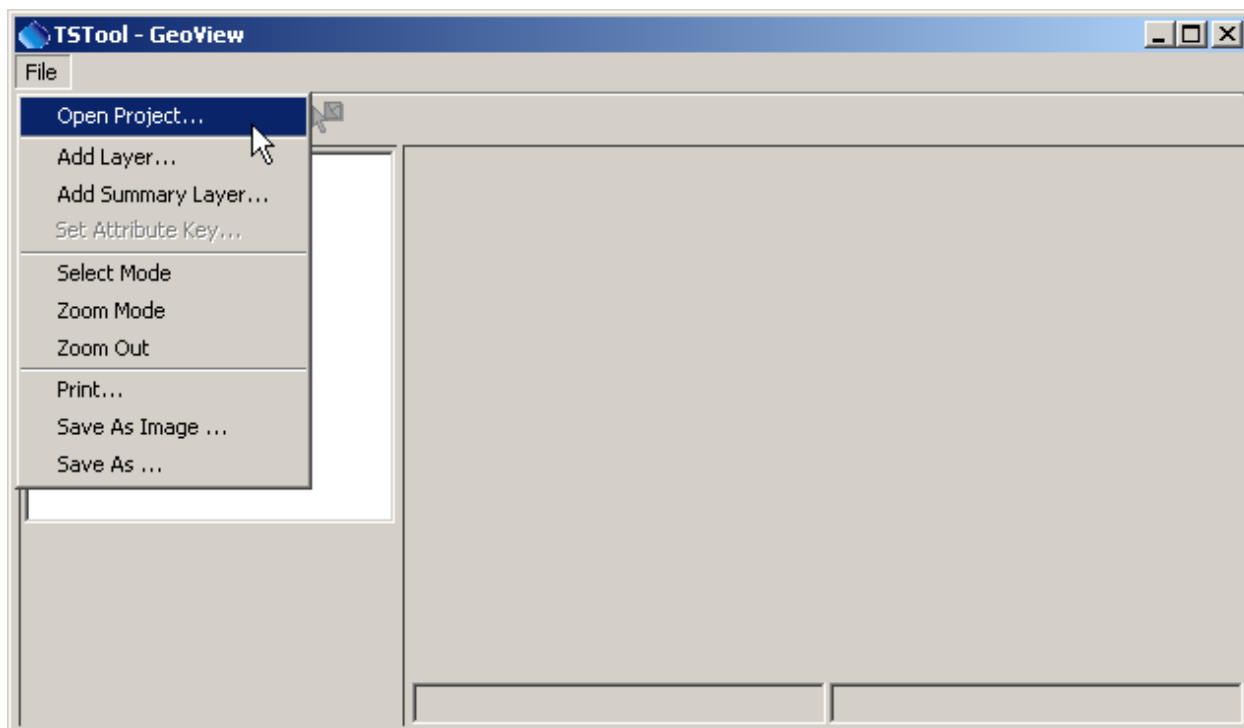
7.2 Opening a Map

To open a map in TSTool, first select the **View...Map** menu item, which will display the following window:



Map (GeoView) Window when First Opened

In this window, select **File...Open Project** and select a GeoView Project File (*.gvp), as shown below:



GeoView_Window_OpenProject

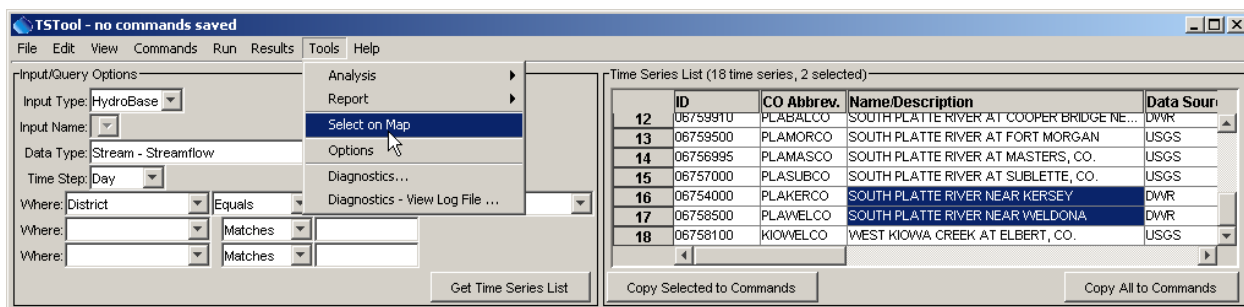
Opening a Map (GeoView Project) File

The format for a GVP file is described in the **GeoView Mapping Tools Appendix**. The file is a simple text file that can be manually edited. Although using an ESRI MXD or other file was considered, such file formats have been changing, are binary, and are proprietary in nature.

After opening the GVP file, a map will be displayed and the TSTool **Tools...Show on Map** button will be enabled when appropriate.

7.3 Using Time Series to Select Locations on the Map

To select time series on the map, first select time series in the upper part of the TSTool interface and then select the **Tools...Select on Map** menu item, as illustrated in the following figure:



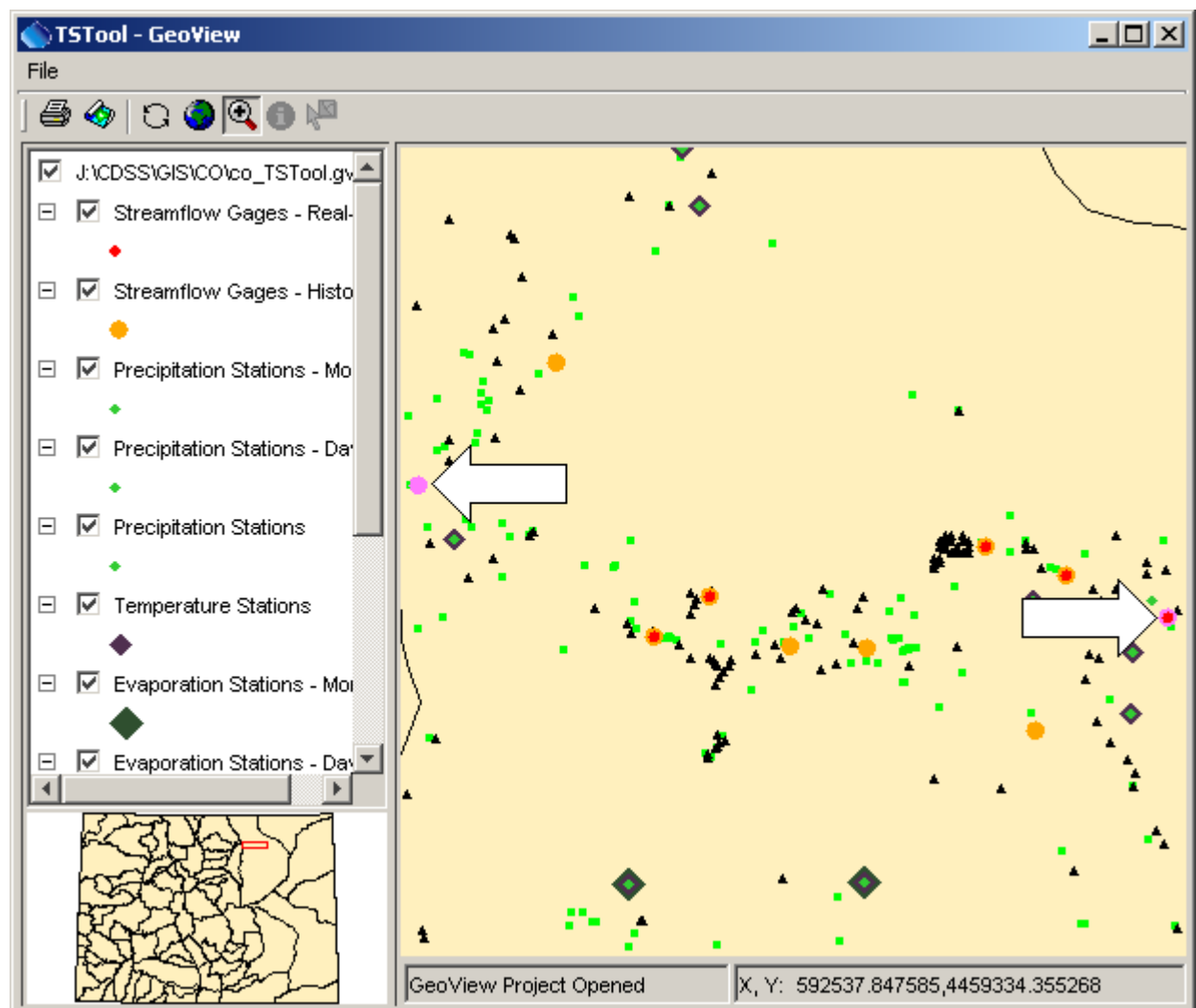
Menu_Tools_SelectOnMap

Example of Selecting Time Series on the Map

Note that in the above example, the selection is initiated from the time series input/query list (not the time series results at the bottom of the TSTool main window). The latter may be implemented in the future; however, it is faster to use the input/query list because only time series header information is processed.

When **Tools...Select on Map** is selected, features on the map are selected as follows:

1. The software tries to find appropriate map layers by matching the lookup file TS_DataType and TS_Interval column values with the **Data Type** and **Time Step** values in the input/query list.
2. The resulting layers (indicated by Layer_Name) are searched to match the **ID** (and optionally **Data Source**) values in the input/query time series list with the attributes indicated by the Layer_Location and Layer_DataSource lookup file (Layer_Interval can also optionally be used).
3. Matching features are selected and the map zooms to highlight the features, as shown in the following figure (the arrows have been added for illustration).



Menu_Tools_SelectOnMap2

Map after Using Time Series to Select Features

If the lookup file does not include a Layer_DataSource, then this value is ignored in the search. Once selected on the map, users can evaluate the significance of the distribution of stations, etc. and can use other map features.

To support this functionality, the attributes for the layer should include a column for the location identifier, and optionally the data source, where the values match the values shown in TSTool. Additionally providing an attribute for data interval allows another level of selection. In this case, the intervals in the attributes must match those shown in TSTool. Providing all information will result in record-level queries that allow a direct link between a time series and a layer.

7.4 Using Locations on the Map to Select Time Series

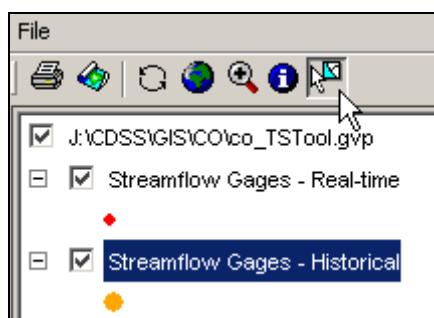
To select time series from the map:

1. Perform a time series query to create a list of time series (see the previous section for an example). **The map can be used to select time series in this list, but selecting from the map will not initiate a database query or file read. This is because the variety of input types that TSTool supports require specific information to query/read time series.**
2. Select a layer of interest on the map. This layer should correspond to time series in the list from the first step. For example, if the time series list contains daily streamflow time series, select the map layer that corresponds to such data. For example:



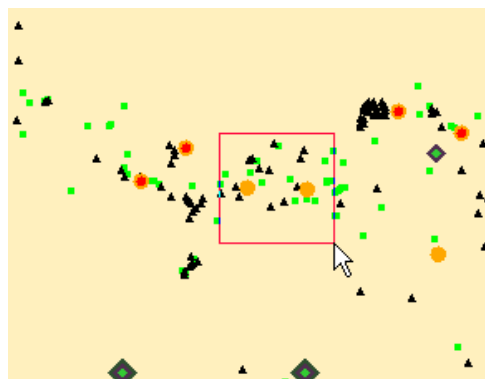
GeoView_SelectLayer

3. Activate the select tool on the map interface toolbar:



GeoView_SelectLayer2

4. Draw a box around features of interest on the map. The software will attempt to match the features in the time series list in the TSTool main window. To do so, it first matches the layer name with the similar value in the lookup file. It then tries to match the **ID** (and optionally **Data Source**) values in the input/query time series list with the attributes indicated by the `Layer_Location` and `Layer_DataSource` lookup file (`Layer_Interval` can also be used). Matched time series are selected in the list. Previous selections are not cleared – use the right click popup menu to clear selections first if appropriate.



Selecting Features on the Map

-Time Series List (18 time series, 2 selected)

	CO Abbrev.	Name/Description
12	PLABALCO	SOUTH PLATTE RIVER AT COOPER BRIDGE
13	PLAMORCO	SOUTH PLATTE RIVER AT FORT MORGAN
14	PLAMASCO	SOUTH PLATTE RIVER AT MASTERS, CO.
15	PLASUBCO	SOUTH PLATTE RIVER AT SUBLETTE, CO.
16	PLAKERCO	SOUTH PLATTE RIVER NEAR KERSEY
17	PLAWELCO	SOUTH PLATTE RIVER NEAR WELDONA
18	KIOWELCO	WEST KIOWA CREEK AT ELBERT, CO.

GeoView_SelectLayer4

Time Series List After Select from Map

- Once time series have been matched, the information can be copied to the commands area for further processing. This capability is therefore useful for identifying available data for an area.

If not all selected features in a layer correspond to time series, a warning similar to the following may be shown:



GeoView_SelectLayer_Warning

Selection Warning

This may indicate that the attributes in the spatial data are not detailed enough to do the lookup. For example, a data source attribute may exist but may only apply to one data interval (e.g., real-time instead of historical data types). A sufficient combination of layers and attributes in layers can avoid or minimize such problems.

The above procedure is not completely robust in that the user may select a layer that does not match the time series list. Additional features are being considered to minimize this possibility. However, the use of the map interface is considered an advanced feature and some reliance is made on a user's capability.

7.5 Spatial Analysis Commands

TSTool's commands provide a powerful analysis and data processing capability. The above sections provided an overview of how to link time series and spatial data. It is envisioned that in the future commands will be added to perform processing of time series, considering spatial data (e.g., weight time series based on their proximity to a point). The ability to perform spatial and temporal analysis in batch mode will be implemented as appropriate to meet user requirements.

This page is intentionally blank.

8 Troubleshooting

Version 09.03.00, 2009-04-13

This chapter discusses how to troubleshoot TSTool problems. **Section 8.1 – Obsolete Commands** lists obsolete commands, which may no longer be supported by current software and should be phased out.

TSTool can run in interactive mode with a graphical user interface (GUI) and in batch mode. In both cases a log file contains messages from the program. By default, the log file is created in the logs directory under the main installation directory. It is recommended that the `StartLog()` command be inserted as the first command in each command file, using the name of the commands file in its name. The log file can then be viewed using the **Tools...Diagnostics** features (see **Chapter 3 – Getting Started**). However, in most cases, the log file should only be used for major troubleshooting because it contains technical details that may not be understandable by the user. The error-handling features of the GUI provide a status for each command. Often, an error in an early command leads to additional errors in other commands and therefore fixing the first error can resolve multiple problems.

The most common problems are program configuration (see the **Installation and Configuration Appendix**), user input error (see the commands reference for command syntax), and data errors for various input types (see below and see also the input type appendices). Other problems should be reported to the TSTool developers (use the **Help...About** menu to list support contacts). You may need to email the log file to support to help determine the nature of a problem.

When running the TSTool GUI, major problems will be indicated with an icon next to the offending command (see **Chapter 3 – Getting Started** for a summary of command error handling features). When running in batch mode, warnings are only printed to the log file. In either case, the log file viewer can be used to pinpoint the source of problems. If the run has been successful the GUI will show no problem indicators and the log file will contain primarily status messages, which provide useful information about data processing.

The following table summarizes common errors and their fixes. If an error is occurring in batch mode, it is useful to run via the GUI to utilize error feedback features.

TSTool Errors and Possible Solutions

Error	Possible solutions
TSTool does not run on Windows (error at start-up).	<ol style="list-style-type: none"> 1. If using the TSTool executable on Windows and the following is shown (or a command line message is printed with a similar message): <div data-bbox="639 352 1234 590" data-label="Image"> </div> <p style="text-align: right; font-size: small;">Troubleshooting_LaunchError</p> <p>This error may be shown if software files have been manually moved. Reinstall using the installation program.</p> 2. If TSTool is run on Windows using the batch file... The batch file (in the <i>bin</i> directory under the main install point) uses a command shell window that may be running out of environment space (in this case you should see a message in the command shell window to that effect). To correct, change the command shell window properties so that the initial memory is 4096 or greater. This may not take effect unless the command shell window was started from the Start menu. <p>Additionally, to help diagnose errors, try running the <i>TSTool.exe</i> or <i>TSTool.bat</i> batch file from a command shell rather than from a desktop shortcut or Windows Explorer. Doing so may print useful messages to the command shell window.</p>
Time series (of any data type) are not returned from the HydroBase database.	<ol style="list-style-type: none"> 1. Verify that the database includes the water districts of interest using the File...Properties...HydroBase Information menu. HydroBase is distributed for the State of Colorado and water divisions and the version being used may be for a division. 2. Verify that the structure/station identifier is valid. For example, a USGS gage identifier may have changed. To verify, try querying by its name rather than the identifier. Also, use the CDSS StateView application to view HydroBase data details for the station/structure (it may have been renamed). 3. If a zero-filled time series file cannot be found, check its path or use the TS <code>alias = NewTimeSeries()</code> command.
A specific time series is not returned from the HydroBase database.	Time series in HydroBase are tagged with the data source (e.g., USGS). These data source abbreviations or their handling by software may have changed over time and a data source in a time series identifier may not be valid. Current software requires the data source for HydroBase time series, if a data source is used with the data type in HydroBase. Try interactively querying the time series to see if the data source has changed.
A data type combination is not available for queries.	TSTool has been implemented to support various input types as much as possible. However, it may not have features to view all time series in an input type. Refer to the input type appendix for limitations on data handling.
Time series (of any	TSTool queries time series by allocating memory for the requested period and

Error	Possible solutions
data type) have - 999 or other missing values.	then filling in values from the database. The output period (or maximum if not specified) may be such that time series values were not found in the database and were set to the missing data value of - 999. Use fill commands to fill the missing data within the requested period.
TSTool fails on large queries or displays out of memory error.	<p>TSTool may run out of memory on queries (hundreds or thousands of time series, depending on machine memory). More time series may be able to be handled if run in batch mode because GUI resources are minimized. To increase the amount of memory that TSTool will use:</p> <ol style="list-style-type: none"> 1. If running on Windows using the TSTool.exe program (the default configuration), increase the value of the -XmxNNNm option in the <i>bin\TSTool.l4j.ini</i> file under the software installation folder. 2. If running on Windows using the <i>TSTool.bat</i> file, change the -XmxNNNm option after the JRE program name to tell Java to allow more memory (increase the number of MB NNN as appropriate for the amount of memory available on the machine – use a high number to force using hard disk swap space if desired). 3. If running on Linux or Unix using the <i>tstool</i> script, change the -XmxNNNm option after the JRE program name to tell Java to allow more memory (increase the number of MB NNN as appropriate for the amount of memory available on the machine – use a high number to force using hard disk swap space if desired).
Unexpected failure.	<p>If there was an error in input that was serious, TSTool may quit processing input. See the log file for details. If the log file does not offer insight, contact support. Specific causes of failure may include:</p> <ol style="list-style-type: none"> 1. TSTool has been developed using a version of Java that is indicated in the metadata for software files. Trying to use an older Java version may cause unexpected errors. This will only be a problem in custom installations where the default Java distributed with TSTool is not used. To determine the Java version that is being used to run TSTool and that was used to create the software, use Tools...Diagnostics and select the Allow debug checkbox. Then use the Help...About TSTool menu item and press Show Software/System Details to display information that includes the Java version that is being used to run TSTool. 2. An unforeseen issue may be occurring. Contact support. You may need to provide the data and command file being used, which will allow troubleshooting and also allow developers to add additional tests to be run before software is released.
Unable to find files correctly.	The working directory is assumed to be the same as the location of the most recently opened or saved command file. The current working directory is generally displayed by editor dialogs that use a path and can also be displayed using File...Properties...TSTool Session . If files are not being found, verify that the path to the file is correct, whether specified as an absolute path or relative to the command file.
When used with the HydroBase input type for a Microsoft Access database, an error occurs	<ol style="list-style-type: none"> 1. TSTool tries to list the ODBC DSN that are available for HydroBase databases. It does so by running the <i>shellcon.exe</i> program, typically installed in <i>\cdss\bin</i> if TSTool is used with CDSS. If this directory is not in the PATH environment variable, the program will not be found and an error will occur. The PATH normally will include the directory after

Error	Possible solutions
<p>selecting the HydroBase database.</p> <p>Microsoft Access versions of HydroBase have not been used for years so this problem is unlikely.</p>	<p>installation, in order to allow TSTool to be run from directories other than the installation directory. The PATH can be checked by opening a command shell and typing path at the command prompt. If the PATH is not properly set, edit it as follows:</p> <ul style="list-style-type: none"> For Windows NT/2000/XP machines, add <i>\CDSS\bin</i> (or <i>\Program Files\RTi\RiverTrak\bin</i>) to the PATH using the Settings...Control Panel...System...Advanced...Environment Variables settings. You may need to have the administrator perform this step. For Windows 95/98 machines, add <i>\CDSS\bin</i> (or <i>\Program Files\RTi\RiverTrak\bin</i>) to the PATH in the <i>autoexec.bat</i> file. Reboot to apply for all subsequent windows. You may have done this previously. <p>A work-around is to manually type in the ODBC DSN in the entry field.</p> <p>2. Only user ODBC DSNs are listed when selecting HydroBase. If the DSN was defined as a system DSN, it will not be listed. Redefine the DSN as a user DSN.</p>
<p>TSTool is unable to load HEC-DSS DLL files and therefore HEC-DSS features are unavailable.</p>	<p>The HEC-DSS library files are distributed in the <i>TSTool-Version\bin</i> folder and by default this is where TSTool looks for the files. If the start folder for TSTool is changed from this folder, the files will not be found. Therefore, do not reconfigure TSTool to start in other than the <i>bin</i> folder.</p>

8.1 Obsolete Commands

TSTool and the commands that it supports have evolved over time. In early versions, many commands used syntax similar to the following:

```
-SomeCommand parameter
```

Later, the function notation with fixed parameter list was adopted:

```
someCommand(parameter1,parameter2)
```

Parameters for such commands were required to be in a specific order and enhancements were difficult to implement because the parameter order needed to be maintained. Recent enhancements have added new commands and converted older commands to a new free-format “named parameter” notation:

```
SomeCommand(Param1=Value1,Param2=Value2)
```

The new notation allows parameters to be omitted when using a default value, and allows new parameters to be added to commands, as necessary, to enhance existing functionality. Old commands are being transitioned to the new syntax until all commands have been updated. Support for the older notation is provided where possible. In most cases, loading an old command file and/or editing an old command with the command editor dialogs will automatically convert from old to new syntax. TSTool provides warnings for commands that are not recognized or are out of date and cannot automatically be updated.

The following table lists obsolete commands. The TSID abbreviation, when inside parentheses for a command, is interchangeable with the time series alias.

TSTool Command Summary – Obsolete Commands

Command	Description	Replacement
add(TSID,TSID1,TSID2,...)	Add the 2 nd + time series to the first time series, retaining the original identifier. This form of the command is obsolete and should be updated to use the new form described that includes a flag for handling missing data.	Current Add() .
-archive_dbhost HostName	This option is normally set during installation and is typically not specified in command files. Specify the Internet host name for the remote HydroBase database server. This is configured at installation time and will be either "localpc" (for a local Microsoft Access HydroBase database, indicating that no remote server is used) or a machine name for the Informix database server. To change the defaults from those in the <i>tstool.bat</i> file, specify this option again on the command line or edit the batch file. See also -dbhost. This option is used in addition to the -dbhost information to allow a TSTool user to switch between the local PC and the main database server.	OpenHydroBase() and configuration information.
-averageperiod MM/YYYY MM/YYYY	Specify the period to be used to compute averages when the -fillhistave option is specified.	SetAveragePeriod()
-batch	Indicates to run in batch mode. This is automatically set if -commands is specified.	None – no longer used.
-browser Path	This option is normally set during installation and is typically not specified in command files. Specify the path to the web browser to use for on-line documentation.	None – no longer used.
CreateTraces()	Create an ensemble from a time series.	CreateEnsemble()
-cy	Output in calendar year format.	SetOutputYearType()
-d#[, #]	Set the debug level. The first number is the debug level for the screen. The second is for the log file. If one level is specified, it is applied to the screen and log file output.	SetDebugLevel()
-data_interval Interval	Indicate the data interval (e.g., MONTH, DAY) to use with all structures/stations indicated by the -slist option. See the appendices for a list of intervals for different input and data types. This option is only available in batch mode.	CreateFromList()
-datasource ODBCDataSourceName	Specify an ODBC Data Source Name to use for the HydroBase database.	OpenHydroBase() and configuration information.
-data_type Type	Indicate the data type (e.g., DivTotal, DQME) to use with all structures/stations indicated by the -slist option. This option is only available in batch mode. This command is obsolete.	CreateFromList()

TSTool Command Summary – Obsolete Commands (continued)

Command	Description	Replacement
<code>day_to_month_reservoir (TSID, ndays, flag)</code>	Read a daily time series and convert to a monthly time series using the reservoir method. This is generally only applied to reservoir storage.	<code>NewEndOfMonthTS</code> <code>FromDayTS()</code> and <code>FillInterpolate()</code>
<code>-dbhost HostName</code>	This option is normally set during installation and is typically not specified in command files. Specify the Internet host name for the primary HydroBase database server. This is configured at installation time and will be either <code>localpc</code> (for a local Microsoft Access database) or a machine name for the Informix database server. To change the defaults from those in the <code>tstool.bat</code> file, specify this option again on the command line or edit the batch file.	<code>OpenHydroBase()</code> and configuration information.
<code>-detailedheader</code>	Insert time series creation information in output headers. This preserves information from the log file that may otherwise be lost. The default is not to generate detailed headers.	See output command parameters to control.
<code>fillCarryForward()</code>	Fill by repeating value.	<code>FillRepeat()</code>
<code>fillconst (TSID, Value)</code>	Fill the time series with a constant value.	<code>FillConstant()</code>
<code>-fillData File</code>	Specify a StateMod format fill pattern file to be used with the <code>fillpattern()</code> command. This command can be repeated for multiple pattern files.	<code>SetPatternFile()</code>
<code>-fillhistave</code>	Currently only enabled for frost dates and monthly data. Indicates that the time series should be filled with the historical average values from the output period where data are missing (after filling by other methods). See also the <code>-averageperiod</code> option.	<code>FillHistMonthAverage()</code> and <code>FillHistYearAverage()</code>
<code>Graph g = newGraph(GraphType, Visibility, TimeSeriesToGraph)</code>	Create a new graph window.	This command is no longer supported. See <code>ProcessTSProduct()</code> .
<code>-helpindex Path</code>	This option is normally set during installation and is typically not specified in command files. Specify the path to help index file for on-line documentation.	No longer used.
<code>-ignorelezero</code>	Treat data values ≤ 0 as missing when computing averages but do not replace when filling.	<code>SetIgnoreLEZero()</code>
<code>-include_missing_ts</code>	If a time series cannot be found, include an empty time series.	<code>SetIncludeMissingTS()</code>

TSTool Command Summary – Obsolete Commands (continued)

Command	Description	Replacement
-informix	Indicate that Informix is used for HydroBase.	Not used.
-missing Value	Use the specified value for missing data values (StateMod only). The default is -999.0.	See WriteStateMod().
-fillusingcomments	This option only applies to diversion time series and causes the diversion comments to be evaluated. Comments that indicate no diversion in an irrigation year will result in missing data for that year being replaced with zeros.	FillUseDiversionComments()
month1/year1 month2/year2	Specifies beginning and ending months for period of record - calculations are still based on the entire period of record (i.e., regression values) but the final output is according to these values, if given. Month 1 is January. Years are 4-digit.	SetOutputPeriod()
-o outputfile	Specify output file name. This is used in conjunction with other -o options.	Write*() commands.
-odatevalue	Output a DateValue format file.	WriteDateValue()
-ostatemod	Output a StateMod format file.	WriteStateMod()
-osummary	Output a time series summary.	WriteSummary()
-osummarynostats	Output a time series summary without statistics (this is used with the data extension procedure developed by Ayres for CDSS).	No longer supported.
regress(TSID1,TSID2)	Performs a linear regression analysis between the two time series, filling missing data of the first time series. Regression information is printed to the log file.	FillRegression()
regress12(TSID1,TSID2) regressMonthly(TSID1,TSID2)	Same as regress() except 12 separate monthly regressions values are calculated.	FillRegression()
regresslog(TSID1,TSID2)	Same as regress() except regressions values are calculated logarithmically.	FillRegression()
regresslog12(TSID1,TSID2) regressMonthlyLog(TSID1,TSID2)	Same as regresslog() except 12 monthly regressions values are calculated.	FillRegression()
setconst(TSID,Value)	Set the time series to the given value for all data. If the time series is not in the database, created an empty time series and then set to a constant value.	SetConstant()
setconstbefore(TSID,Value,Date)	The time series to the given value for all data on and before the specified date (YYYY-MM or MM/YYYY).	SetConstant()
setConstantBefore()	Set a value constant before a date/time.	SetConstant()
SetMissingDataValue()	Set the missing data value used in a StateMod time series.	See WriteStateMod().

TSTool Command Summary – Obsolete Commands (continued)

Command	Description	Replacement
<code>setQueryPeriod(Start,End)</code>	Set the global period to query databases and read from files.	<code>SetInputPeriod()</code>
<code>-sqlserver</code>	Specify that SQL Server is used for HydroBase.	<code>OpenHydroBase()</code> and configuration information. SQL Server is also now the default because Microsoft Access is no longer supported.
<code>-slist File</code>	Create time series from a list file.	<code>CreateFromList()</code>
<code>-units value</code>	Output using the specified units (default is to use database units).	No longer used. If necessary, units can be converted by a number of commands including <code>ConvertUnits()</code> .
<code>-w# [, #]</code>	Set the warning level. The first number is the warning level for the screen. The second is for the log file. If one level is specified, it is applied to the screen and log file output.	<code>SetWarningLevel()</code>
<code>-wy</code>	Output in water year format.	<code>SetOutputYearType()</code>

This page is intentionally blank.

9 Quality Control

Version 09.04.00, 2009-02-26

This chapter discusses how TSTool software is quality controlled and how to use TSTool for quality control.

9.1 Quality Control for TSTool Software

TSTool software provides many data processing commands. Each command typically provides multiple parameters. The combination of commands and parameters coupled with potential data changes and user errors can make it difficult to confirm that TSTool software is itself performing as expected. To address this quality control concern, several commands have been built into TSTool to facilitate using TSTool itself to test functionality. Test cases can be defined for each command, with test cases for various combinations of parameters. The suite of all the test cases can then be run to confirm that the version of TSTool does properly generate expected results. This approach performs regression testing of the software and utilizes TSTool's error-handling features to provide visual feedback during testing.

Test cases are developed by software developers as new features are implemented, according to the following documentation. However, users can also develop test cases and this is encouraged to ensure that all combinations of parameters and input data are tested. Providing verified test data and results prior to new development will facilitate the new development.

9.1.1 Writing a Single Test Case

A single test case is illustrated by the following example (indented lines indicate commands that are too long to fit on one line in the documentation).

```
# Test filling with interpolation where maximum gap interval to fill is 2.
StartLog(LogFile="Results/Test_FillInterpolate_MaxIntervals=2.TSTool.log")
RemoveFile(InputFile="Results/Test_FillInterpolate_MaxIntervals=2_out.dv",
    IfNotFound=Ignore)
TS tsl_day = NewPatternTimeSeries(NewTSID="tsl...Day",
    Description="test data 1",SetStart="2000-01-01",SetEnd="2003-05-13",
    PatternValues="1,2,3,2,1,-999,5,1,-999,-999,-999,1,3,5")
FillInterpolate(TSList=AllMatchingTSID,TSID="tsl_day",MaxIntervals=2)
# Uncomment the following command to regenerate expected results.
# WriteDateValue(OutputFile="ExpectedResults/Test_FillInterpolate_MaxIntervals=2_out.dv")
WriteDateValue(OutputFile="Results/Test_FillInterpolate_MaxIntervals=2_out.dv")
CompareFiles(InputFile1="ExpectedResults/Test_FillInterpolate_MaxIntervals=2_out.dv",
    InputFile2="Results/Test_FillInterpolate_MaxIntervals=2_out.dv",WarnIfDifferent=True)
```

Example Test Case Command File

The purpose of the test case command file is to regenerate results and then compare the results to previously generated and verified expected results. The example illustrates the basic steps that should be included in any test case:

1. **Start a log file to store the results of the specific test case.** The previous log file will be closed and the new log file will be used until it is closed. The log file is not crucial to the test but helps with troubleshooting if necessary (for example if evaluating the test case output when run in a test suite, as explained later in this chapter).

2. **Remove the results that are to be generated by the test.** This is necessary because if the software fails and old results match expected results, it may appear that the command was successful. Using the `IfNotFound=Ignore` parameter is useful because someone who is running the tests for the first time may not have previous results. Test developers should use `IfNotFound=Warn` when setting up the test to confirm that the results being removed match the name that is actually generated in a later command, and then switch to `IfNotFound=Ignore`.
3. **Generate or read time series data.** The `NewPatternTimeSeries()` command is used in the example to create a time series of repeating values. This is a useful technique because it allows full control over the initial data and minimizes the number of files associated with the test. Synthetic data are often appropriate for simple tests. If the test requires more complicated data, then time series can be read from a `DateValue` or other time series file. For example, if functionality of another software program is being implemented in TSTool, the data file from the original software may be used.
4. **Process the time series using the command being tested.** In the example, the `FillInterpolate()` command is being tested. In many cases, a single command can be used in this step. However, in some cases, it is necessary to use multiple commands. This is OK as long as each command or the sequence is sufficiently tested with appropriate test cases.
5. **Write the results.** The resulting time series are written to a standard format. The `DateValue` format is useful for general testing because it closely represents all time series properties. Note that two write commands are used in the example – one writes the expected results and the other writes the results from the current test. The expected results should only be written when the creator of the test has confirmed that it contains verified values. In the example, the command to write expected results is commented out because the results were previously generated. Some commands do not process time series; therefore, the `WriteProperty()` and `WriteTimeSeriesProperty()` commands can be used to write processor properties (e.g., global output period) and time series properties (e.g., data limits). Additional properties will be enabled as the software is enhanced.
6. **Compare the expected results and the current results.** The example uses the `CompareFiles()` command to compare the `DateValues` files generated for the expected and current results. This command omits comment lines in the comparison because file headers often change due to dynamic comments with date/time. If the software is functioning as expected, the data lines in the file will exactly match. The example illustrates that if the files are different, a warning will be generated because of the `WarnIfDifferent=True` parameter. Other options for comparing results include:
 - a. **Use the `CompareTimeSeries()` command.** This command expects to find matching time series and will compare data values to a precision. For example, read one time series from a `DateValue` file and then compare with the current time series in memory. Using this command avoids potential issues with the `DateValue` or other file formats changing over time (and requiring the expected results to be reverified); however, doing a file comparison is often easier to troubleshoot because a graphical difference program can visually illustrate differences that need to be evaluated.
 - b. **If testing a read/write command, compare the results with the original data file.** For example, if the test case is to verify that a certain file format is properly read, then there will generally also be a corresponding write command. The test case can then consist of a command to read the file, a command to write the results, and a comparison command to compare the two files. This may not work if the header of the file uses comment lines that are not recognized by the `CompareFiles()` command.

If the example command file is opened and run in TSTool, it will produce time series results, the log file, and the output file. If the expected and current results are the same, no errors will be indicated. However,

if the files are different, a warning indicator will be shown in the command list area of the main window next to the **CompareFiles()** command.

General guidelines for defining test cases are as follows. Following these conventions will allow the test cases to be incorporated into the full test suite.

- Define the test case in a folder matching the command name.
- Name the command file with prefix *Test_*, extension *.TSTool*, and use the following guidelines:
 - for the default case using the filename pattern *Test_CommandName.TSTool*
 - If there is a reason to define a test for a specific data set or input, add additional information to the filename, for example: *Test_CommandName_RiverX.TSTool* or *Test_CommandName_6Hour.TSTool*
 - If defining a test for legacy syntax, name the command file as follows (and see the @readOnly comment tag described in **Section 9.1.3**): *Test_CommandName_Legacy.TSTool*
 - If defining a test for parameter values other than the default values, use a command file name similar to the following, where the parameters are listed at the end of the file name body: *Test_CommandName_Param1=Value1,Param2=Value2.TSTool*
Although this can result in very long names, the explicit naming clarifies the purpose of the test. The name of the example command file shown above is *Test_FillInterpolate_MaxIntervals=2.TSTool*.
- Add a short comment to the top of the test case explaining the test.
- Use as little data as possible to perform the test – long time series cause tests to run longer and take up more space in the repository that is used for revision control. Even though hundreds or thousands of tests may ultimately be defined, it is important to be able to run them in a short time to facilitate testing.
- If possible, test only one command in the test – more complicated testing is described in **Section 9.1.4**.
- If an input file is needed, place it in a folder named *Data*, if necessary copying the same input from another command – this may require additional disk space but ensures that each command can stand alone.
- Write the expected results to a folder named *ExpectedResults*.
- Write the generated results and other dynamic content, including log file, to a folder named *Results*.
- (Recommended) When creating output files, use *_out* in the filename before the extension and use an extension that is appropriate for the file content – this helps identify final output products in cases where intermediate files might be produced.

9.1.2 Creating and Running a Test Suite

The previous section described how to define a single test case. However, opening and running each test case command file would be very tedious and inefficient. Therefore, TSTool provides a way to generate and run test suites, which is the approach taken to perform a full regression test prior to a software release.

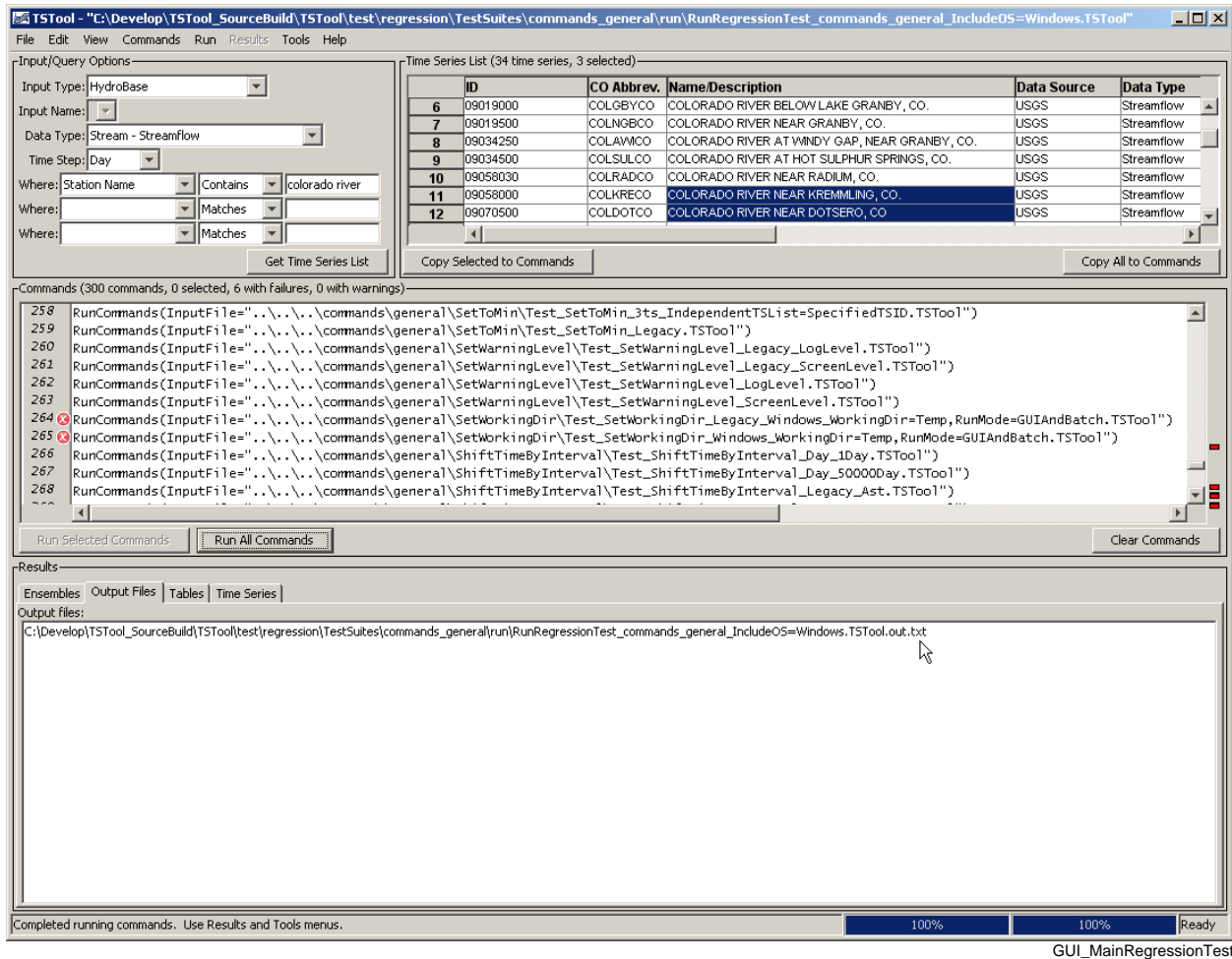
The following example command file illustrates how to create a test suite:

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

When the command file is run, it searches the indicated search folder for files matching the pattern *Test_*.TSTool*. It then uses this list to create a command file with contents similar to the following example (its contents are truncated in the following figure due to length). This file will be listed as an output file after running the above command file. The IncludeTestSuite and IncludeOS parameters are described in **Section 9.1.3**.

```
# File generated by...
# program:      TSTool 9.00.04 (2009-01-20)
# user:         sam
# date:         Tue Jan 20 22:56:17 MST 2009
# host:         SOPRIS
# directory:    C:\Develop\TSTool_SourceBuild\TSTool\test\regression\TestSuites\commands_general\create
# command line: TSTool -home test/operational/RTi
#
# The following 287 test cases will be run to compare results with expected results.
# Individual log files are generally created for each test.
# The following test suites from @testSuite comments are included: *
# Test cases for @os comments are included: Windows
StartRegressionTestResultsReport(
    OutputFile="RunRegressionTest_commands_general_IncludeOS=Windows.TSTool.out.txt")
RunCommands(InputFile="..\..\..\commands\general\Add\Test_Add_1.TSTool")
RunCommands(InputFile="..\..\..\commands\general\Add\Test_Add_Ensemble_1.TSTool")
RunCommands(InputFile="..\..\..\commands\general\AddConstant\Test_AddConstant_1.TSTool")
RunCommands(InputFile="..\..\..\commands\general\AddConstant\Test_AddConstant_Legacy_Ast.TSTool")
RunCommands(InputFile="..\..\..\commands\general\AddConstant\Test_AddConstant_Legacy_NoAst.TSTool")
...omitted...
RunCommands(InputFile="..\..\..\commands\general\WriteSummary\Test_WriteSummary_1.TSTool")
```

The above command file can then be opened and run. Each RunCommands () command will run a single test case command file. Warning and failure statuses from each test case command file are propagated to the test suite RunCommands () command. The output from running the test suite will be all of the output from individual test cases (in the appropriate *Results* folders) plus the regression test report provided in the TSTool **Results** list in the main window. An example of the TSTool main window after running the test suite is shown in the following figure. Note the warnings and errors, which should be addressed before releasing the software (in some cases commands are difficult to test and more development on the test framework is needed).



TSTool Main Interface Showing Regression Test Results

An excerpt from the output file is shown below (normally the test number would be sequential from 1 to the number of tests but only a few examples are included below).

```
# File generated by...
# program:      TSTool 9.00.04 (2009-01-20)
# user:         sam
# date:         Wed Feb 25 16:59:52 MST 2009
# host:         SOPRIS
# directory:    C:\Develop\TSTool_SourceBuild\TSTool\test\regression\TestSuites\commands_general\run
# command line: TSTool

# The test status below may be PASS or FAIL.
# A test can pass even if the command file actual status is FAILURE, if failure is expected.
#
# Test    Commands    Commands
# Pass/   Expected    Actual
# Num Fail Status      Status      Command File
#-----
1  PASS  SUCCESS  SUCCESS  C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands_general\
Add\Test_Add_1.TSTool
2  PASS  SUCCESS  SUCCESS  C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands_general\
Add\Test_Add_Ensemble_1.TSTool
34 PASS  Warning  WARNING  C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands_general\
CreateFromList\Test_CreateFromList_InputType=HydroBase,
IDCol=1,DataSource=DWR,DataType=DivTotal,
Interval=Month,IfNotFound=Ignore.TSTool
35 PASS  Failure  FAILURE  C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands_general\
CreateFromList\Test_CreateFromList_InputType=HydroBase,
IDCol=1,DataSource=DWR,DataType=DivTotal,
Interval=Month,IfNotFound=Warn.TSTool
36 PASS  Success  SUCCESS  C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands_general\
```

				CreateFromList\Test_CreateFromList_InputType=HydroBase, OutputPeriod,IDCol=1,DataSource=DWR,DataType=DivTotal, Interval=Month,IfNotFound=Default.TSTool
37	PASS	Warning	WARNING	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ CreateFromList\Test_CreateFromList_Legacy.TSTool
38	PASS	Failure	FAILURE	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ CreateTraces_Alias\Test_CreateTraces_Legacy_1.TSTool
251	*FAIL*	SUCCESS	FAILURE	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ SetWorkingDir\Test_SetWorkingDir_Legacy_Windows_WorkingDir=Temp, RunMode=GUIAndBatch.TSTool
252	*FAIL*	SUCCESS	FAILURE	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ SetWorkingDir\Test_SetWorkingDir_Windows_WorkingDir=Temp, RunMode=GUIAndBatch.TSTool
287	PASS	SUCCESS	SUCCESS	C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ WriteTimeSeriesProperty\ Test_WriteTimeSeriesProperty_PropertyName=DataLimitsOriginal.TSTool
#-----				
# FAIL count = 6				
# PASS count = 281				

A test passes if its expected status (by default SUCCESS) matches the actual status, and the test fails otherwise. Note that there are cases where a test case is actually intended to fail, in order to test that TSTool is properly detecting and handling the failure (rather than ignoring it or crashing).

The features built into TSTool can therefore be used to efficiently test the software, contributing to increased software quality and efficient software releases. See the next section for more information on controlling the test process.

9.1.3 Controlling Tests with Special Comments

The previous two sections described how to define individual test cases and how to automatically create and run a test suite comprised of test cases. However, there are special conditions that will cause the normal testing procedures to fail, in particular:

- tests depend on a database that is not available
- tests depend on a database version that is not available (data in the “default” database have changed)
- tests can only be run on a certain operating system
- tests depend on a specific environment configuration that is not easily reproduced for all users

Any of these conditions can cause a test case to fail, leading to inappropriate errors and wasted time tracking down problems that do not exist. To address this issue, TSTool recognizes special comments that can be included in test case command files. The following table lists tags that can be placed in # comments in command files to provide information for to the CreateRegressionTestCommandFile() command and command processor. The syntax of the special comments is illustrated by the following example:

```
#@expectedStatus Failure
```


Special #-comment Tags

Parameter	Description
@expectedStatus Failure @expectedStatus Warning	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 @os UNIX	Using this tag indicates that the test is designed to work only on the specified platform and will be included in the test suite by the CreateRegressionTestCommandFile() command 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.
@readOnly	Use this tag to indicate that a command file is read-only. This is useful when legacy command files are being tested because TSTool will automatically update old syntax to new. Consequently, saving the command file will overwrite the legacy syntax and void the test. If this tag is included, the TSTool interface will warn the user that the file is read-only and will only save if the user indicates to do so.
@testSuite ABC	Indicate that the command file should be considered part of the specified test suite, as specified with the IncludeTestSuite parameter of the CreateRegressionTestCommandFile() command. Do not specify a test suite tag for general tests. 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.

Using the above special comment tags, it is possible to create test suites that are appropriate for specific environments. For example, using @testSuite HydroBase indicates that a test case should be included in the HydroBase test suite, presumably run in an environment where a connection to HydroBase has been opened. Consequently, multiple test suites can be created and run as appropriate depending on the system environment.

9.1.4 Verifying TSTool Software Using a Full Dataset

The previous sections described how to test TSTool software using a suite of test cases. This approach can be utilized when performing general tests, for example prior to a normal software release. However, there may be cases where TSTool has been used to produce a large data set and it is desirable to confirm that a software release will still create the full dataset without differences. For example, for the State of Colorado's Decision Support Systems, large basin model data sets are created and are subject to significant scrutiny. Approaches previously described in this chapter can be utilized to verify that TSTool is functioning properly and creates the dataset files. The following procedure is recommended and uses CDSS as an example:

1. If not already installed, install the data set in its default location (e.g., *C:\CDSS\data\colorado_1_2007*) – these files **will not be modified** during testing.
2. Create a parallel folder with a name indicating that it is being used for verification (e.g., *C:\CDSS\data\colorado_1_2007_verify20090216*).
3. Copy the data set files from step 1 to the folder created in step 2 (e.g., copy to *C:\CDSS\data\colorado_1_2007_verify20090216\colorado_1_2007*) – these files **will be modified** during testing.
4. Create a TSTool command file in the folder created in step 2 that will run the tests (e.g., *VerifyTSTool.TSTool*). It is often easier to edit this command file with a text editor rather than with TSTool itself. The contents of the file are illustrated in the example below. Some guidelines for this step are as follows:
 - a. Organize the command file by data set folder, in the order that data need to be created.
 - b. Process every *.*TSTool* command to verify that it runs and generates the same results.
 - c. If command files do not produce the same results, copy the command file to a name with “-updated” or similar in the filename and then change the file until it creates the expected results. This may be required due to changes in the command, for example implementing stricture error handling. These command files can then be shared with maintainers of the data set so that future releases can be updated.
 - d. As tests are formalized, it may be beneficial to save a copy of this file with the original data set so future tests can simply copy the verification command file rather than recreating it (e.g., save in a *QualityControl* folder in the master data set). This effort will allow the creator of the data set to quality control their work as well as helping to quality control the software.
5. Run the command file – any warnings or failures should be evaluated to determine if they are due to software or data changes. Software differences should be evaluated by software developers. It may be necessary to use command parameters such as *Version*, available for some commands, to recreate legacy data formats.

The following example command file illustrates how TSTool software is verified using the full data set (indented lines indicate commands that are too long to fit on one line in the documentation). Note that intermediate input files that would normally be modified by other software (e.g., StateDMI for CDSS data sets) could impact TSTool verification. However, a similar quality control procedure can be implemented for StateDMI.

Guidelines for setting up the each test in the command file are as follows:

1. Remove output files that are generated from each individual command file that is run using `RemoveFile()` commands. This will ensure that test does not use old results for its output comparison.
2. Run each individual command file using the `RunCommands()` command.
3. Compare the results of the run with the original data set file using the `CompareFiles()` command.

```

StartLog(LogFile="VerifyTSTool.TSTool.log")
# This command file verifies the TSTool functionality by recreating a released
# StateMod/StateCU data set. The general process is as follows:
# 1) Copy the entire original data set to this folder (e.g., do manually).
# 2) Commands below will remove output files from product and StateMod/StateCU
# folders. This is done in case regeneration stops - don't want any confusion
# with original output and what should be created here.
# 3) Commands below will run the command files used to generate the model files.
# 4) Commands below will use CompareFile() commands to compare results. Comment
# lines are ignored so only data differences (processing output) will be
# flagged.
# If run interactively from TSTool, indicators will show where results are
# different. Differences must then be evaluated to determine if input data,
# process, or software have changed. Differences may be valid.
#
#####
# Diversions
#####
RemoveFile(InputFile="colorado_1_2007\Diversions\514634.stm")
RunCommands(InputFile="colorado_1_2007\Diversions\514634.stm.commands.TSTool")
CompareFiles(InputFile1="colorado_1_2007\Diversions\514634.stm",
             InputFile2="..\colorado_1_2007\Diversions\514634.stm",WarnIfDifferent=True)
#
RemoveFile(InputFile="colorado_1_2007\Diversions\954699.stm")
RunCommands(InputFile="colorado_1_2007\Diversions\954699.commands.TSTool")
CompareFiles(InputFile1="colorado_1_2007\Diversions\954699.stm",
             InputFile2="..\colorado_1_2007\Diversions\954699.stm",WarnIfDifferent=True)
#
RemoveFile(InputFile="colorado_1_2007\Diversions\Fraser.stm")
RunCommands(InputFile="colorado_1_2007\Diversions\Fraser.commands.TSTool")
CompareFiles(InputFile1="colorado_1_2007\Diversions\Fraser.stm",
             InputFile2="..\colorado_1_2007\Diversions\Fraser.stm",WarnIfDifferent=True)
#
#####
# instream
#####
RemoveFile(InputFile="colorado_1_2007\statemod\cm2005.ifm")
RunCommands(InputFile="colorado_1_2007\instream\ifm.commands.tstool")
CompareFiles(InputFile1="colorado_1_2007\statemod\cm2005.ifm",
             InputFile2="..\colorado_1_2007\statemod\cm2005.ifm",WarnIfDifferent=True)
#
#####
# reservoirs
#####
RemoveFile(InputFile="colorado_1_2007\reservoirs\363543-add.stm")
RemoveFile(InputFile="colorado_1_2007\reservoirs\364512-add.stm")
RemoveFile(InputFile="colorado_1_2007\reservoirs\503668-add.stm")
RemoveFile(InputFile="colorado_1_2007\reservoirs\513709-add.stm")
RemoveFile(InputFile="colorado_1_2007\reservoirs\514620-add.stm")
RemoveFile(InputFile="colorado_1_2007\reservoirs\723844-add.stm")
RemoveFile(InputFile="colorado_1_2007\reservoirs\838713-add.stm")
RunCommands(InputFile="colorado_1_2007\reservoirs\res.stm.commands.tstool")
# CompareFiles(InputFile1="colorado_1_2007\reservoirs\363543-
add.stm",InputFile2="..\colorado_1_2007\reservoirs\363543.stm",WarnIfDifferent=True)
# CompareFiles(InputFile1="colorado_1_2007\reservoirs\364512-add.stm",
             InputFile2="..\colorado_1_2007\reservoirs\364512-add.stm",WarnIfDifferent=True)
# CompareFiles(InputFile1="colorado_1_2007\reservoirs\503668-add.stm",
             InputFile2="..\colorado_1_2007\reservoirs\503668-add.stm",WarnIfDifferent=True)
# CompareFiles(InputFile1="colorado_1_2007\reservoirs\513709-add.stm",
             InputFile2="..\colorado_1_2007\reservoirs\513709-add.stm",WarnIfDifferent=True)
# CompareFiles(InputFile1="colorado_1_2007\reservoirs\514620-add.stm",
             InputFile2="..\colorado_1_2007\reservoirs\514620-add.stm",WarnIfDifferent=True)
# CompareFiles(InputFile1="colorado_1_2007\reservoirs\723844-add.stm",
             InputFile2="..\colorado_1_2007\reservoirs\723844-add.stm",WarnIfDifferent=True)
# CompareFiles(InputFile1="colorado_1_2007\reservoirs\838713-add.stm",

```

```

        InputFile2="..\colorado_1_2007\reservoirs\838713-add.stm",WarnIfDifferent=True)
#
RemoveFile(InputFile="colorado_1_2007\statemod\cm2005B.tar")
RunCommands(InputFile="colorado_1_2007\reservoirs\Btar.commands.tstool")
CompareFiles(InputFile1="colorado_1_2007\statemod\cm2005B.tar",
        InputFile2="..\colorado_1_2007\statemod\cm2005B.tar",WarnIfDifferent=True)
#
RemoveFile(InputFile="colorado_1_2007\statemod\cm2005C.tar")
RunCommands(InputFile="colorado_1_2007\reservoirs\Ctar.commands.tstool")
CompareFiles(InputFile1="colorado_1_2007\statemod\cm2005C.tar",
        InputFile2="..\colorado_1_2007\statemod\cm2005C.tar",WarnIfDifferent=True)
#
RemoveFile(InputFile="colorado_1_2007\statemod\cm2005.eom")
RunCommands(InputFile="colorado_1_2007\reservoirs\eam.commands.tstool")
CompareFiles(InputFile1="colorado_1_2007\statemod\cm2005.eom",
        InputFile2="..\colorado_1_2007\statemod\cm2005.eom",WarnIfDifferent=True)
#
RemoveFile(InputFile="colorado_1_2007\statemod\cm2005H.tar")
RunCommands(InputFile="colorado_1_2007\reservoirs\Htar.commands.tstool")
CompareFiles(InputFile1="colorado_1_2007\statemod\cm2005H.tar",
        InputFile2="..\colorado_1_2007\statemod\cm2005H.tar",WarnIfDifferent=True)
#
#####
# streamSW
#####
#
RemoveFile(InputFile="colorado_1_2007\streamSW\404657.stm")
RunCommands(InputFile="colorado_1_2007\streamSW\404657.commands.tstool")
CompareFiles(InputFile1="colorado_1_2007\streamSW\404657.stm",
        InputFile2="..\colorado_1_2007\streamSW\404657.stm",WarnIfDifferent=True)
#
RemoveFile(InputFile="colorado_1_2007\statemod\cm2005.rbd")
RunCommands(InputFile="colorado_1_2007\streamSW\rbd.commands.tstool")
CompareFiles(InputFile1="colorado_1_2007\statemod\cm2005.rbd",
        InputFile2="..\colorado_1_2007\statemod\cm2005.rbd",WarnIfDifferent=True)
#
RemoveFile(InputFile="..\colorado_1_2007\statemod\cm2005.rid")
RunCommands(InputFile="colorado_1_2007\streamSW\rbd.commands.tstool")
CompareFiles(InputFile1="colorado_1_2007\statemod\cm2005.rid",
        InputFile2="..\colorado_1_2007\statemod\cm2005.rid",WarnIfDifferent=True)
#
RemoveFile(InputFile="colorado_1_2007\statemod\cm2005.rih")
RunCommands(InputFile="colorado_1_2007\streamSW\rih.commands.tstool")
CompareFiles(InputFile1="colorado_1_2007\statemod\cm2005.rih",
        InputFile2="..\colorado_1_2007\statemod\cm2005.rih",WarnIfDifferent=True)
#
#####
# TSTool - do after others, in case differences might cascade
#####
#
RemoveFile(InputFile="colorado_1_2007\TSTool\fill2005.pat")
RunCommands(InputFile="colorado_1_2007\TSTool\fill2005.pat.commands.TSTool")
CompareFiles(InputFile1="colorado_1_2007\TSTool\fill2005.pat",
        InputFile2="..\colorado_1_2007\TSTool\fill2005.pat",WarnIfDifferent=True)
#
RemoveFile(InputFile="colorado_1_2007\TSTool\Patgage.xbg")
RunCommands(InputFile="colorado_1_2007\TSTool\pattern_gage_raw_data.commands.TSTool")
CompareFiles(InputFile1="colorado_1_2007\TSTool\patgage.xbg",
        InputFile2="..\colorado_1_2007\TSTool\patgage.xbg",WarnIfDifferent=True)

```

9.2 Using TSTool to Quality Control Data

This section will be expanded in the future. Current development is enhancing TSTool's data quality control features.

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.
- DateTime** – A specific date/time associated with time series data. See the `setDataValue()` command.
- 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.

ExtremeValue – The threshold value when adjusting extreme values. 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.

FillUsingCIU – Indicate whether missing data values should be filled using “currently in use” (CIU) data from HydroBase. Additional zeros will be included in data. See the `fillUsingDiversionComments()` command.

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.

IDCol – The column number (or name) to be read from a delimited 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.

Interval – The data interval to use when forming a `TSID`. See the `createFromList()`, `readHydroBase()`, and `shiftTimeByInterval()` commands.

- K** – 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.
- MatchLocation** – Indicate whether the location part of a TSID (Alias) 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 the main time series. See the `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.

Percentile – Percentile value(s) used when analyzing time series. See the `analyzePattern()` command.

Pos – The position in the time series list. See the `deselectTimeSeries()` and `selectTimeSeries()` commands.

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.

ScaleValue – A 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.
- Suffix** – The suffix to be automatically applied to the name of a file. See the `setLogFile()` 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 `fillDayTSFrom2MonthTSAnd1DayTS()` command.

TSID2 – Time series identifier for the first daily time series in a command. See the `fillDayTSFrom2MonthTSAnd1DayTS()` 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.

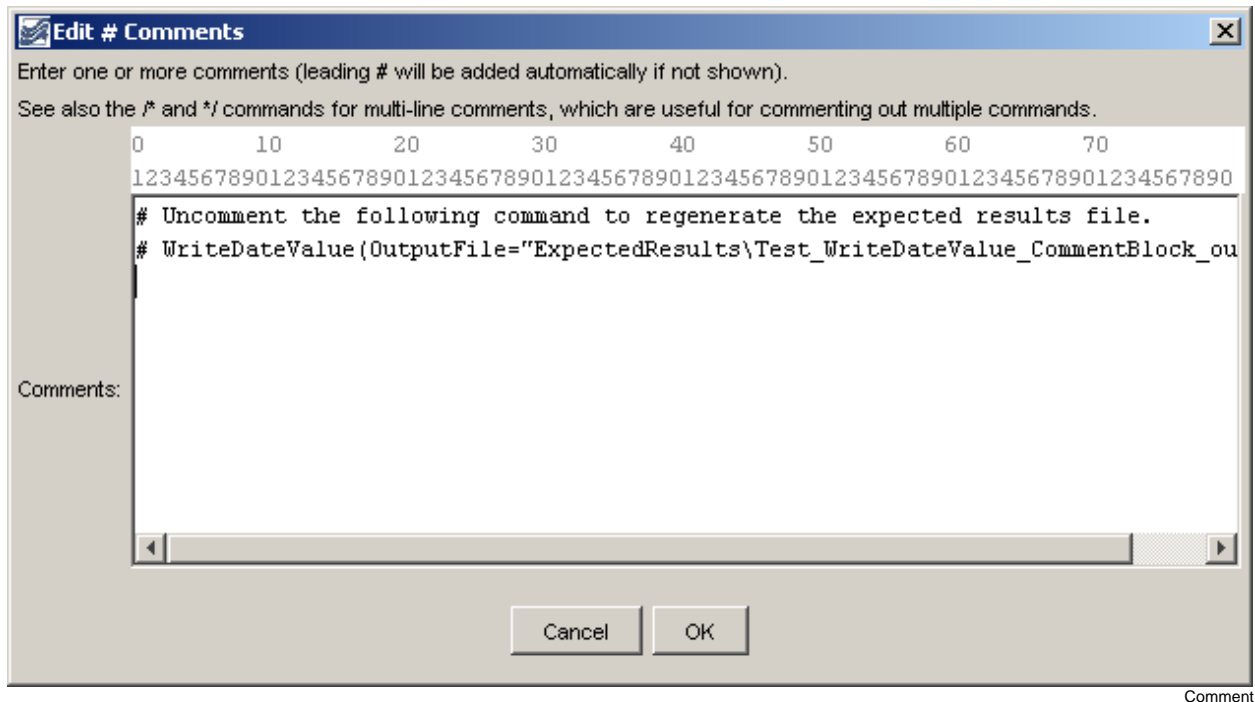
Command Reference:

Comment line

Version 08.17.00, 2008-10-01

The # command indicates single-line comments. Commands can be converted to and from # comments. See also the /* and */ comment block commands, which are to comment multiple commands.

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



Command Editor

The command syntax is as follows:

```
# Some text
```

A sample command file is as follows:

```
#  
# Some comments...  
#
```

This page is intentionally blank.

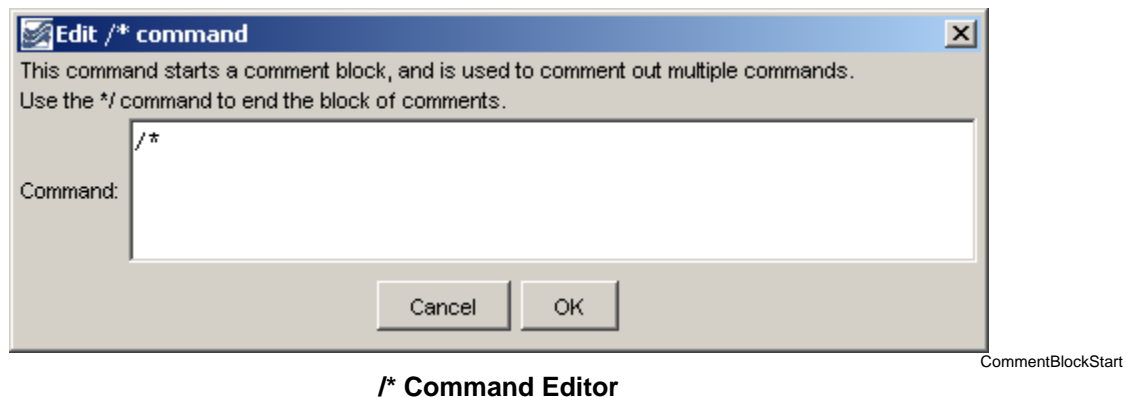
Command Reference: **/***

Comment block start

Version 08.17.00, 2008-10-01

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.

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



The command syntax is as follows:

`/*`

A sample command file is as follows:

```
/*  
SomeCommentedOutCommands()...  
*/
```

This page is intentionally blank.

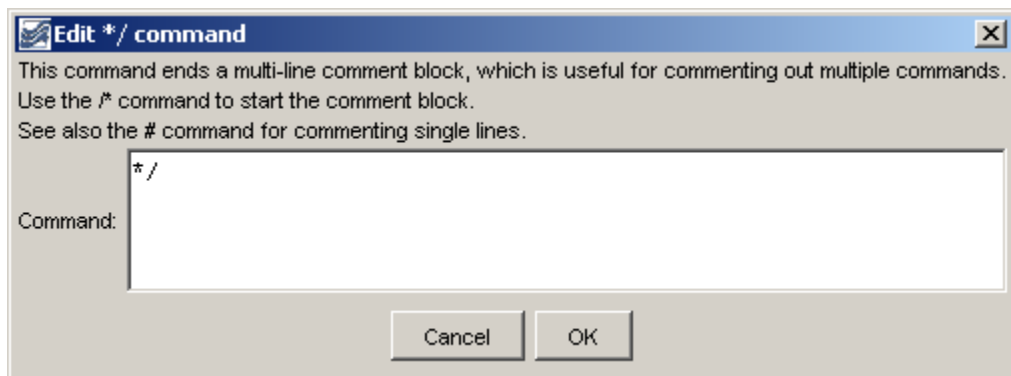
Command Reference: */

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:



CommentBlockEnd

***/ 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

Version 09.00.03, 2009-01-15

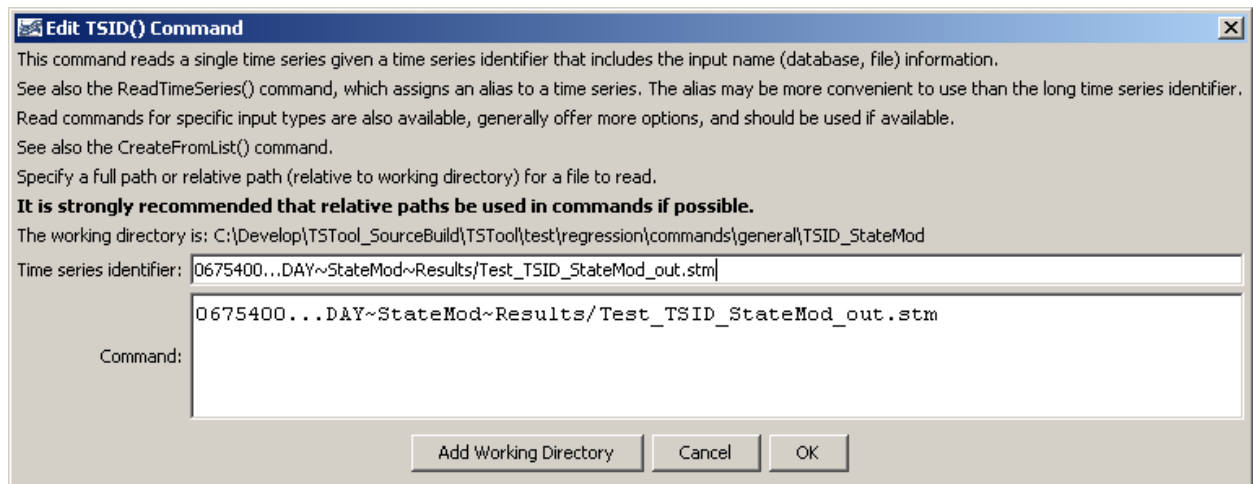
A time series identifier (TSID) command reads a single time series. In order to read the time series from a persistent format (database, file, or web site), the TSID must contain the input type, and if necessary, the input name. For example, a TSID command for the State of Colorado's StateMod model file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

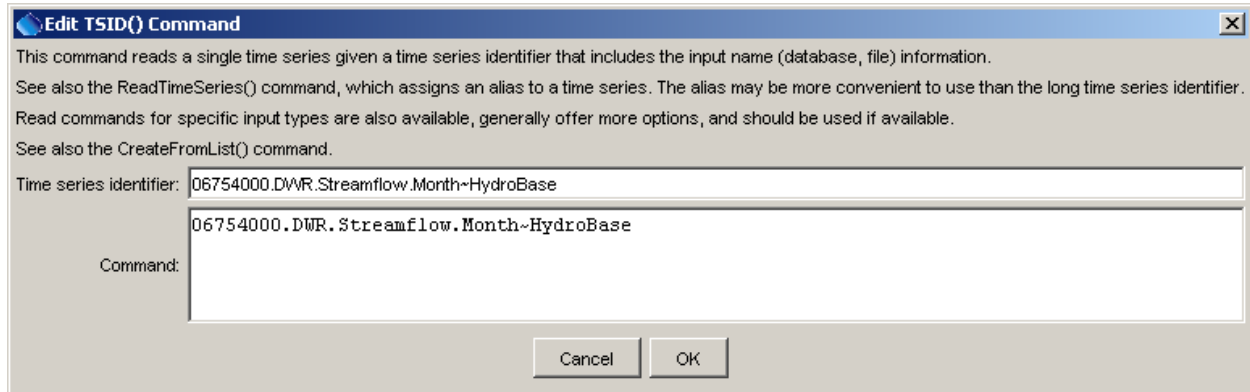
The following dialog is used to edit the command and illustrates the syntax of the command. Limited checks are done while editing the command. However, once committed, TSTool will attempt to read the time series metadata and will issue a warning if unable to read the data. Time series identifiers that 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.



TSID_StateMod

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.



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 08.15.00, 2008-05-04

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.

Edit Add() Command

Add one or more time series to a time series (or ensemble of time series). The receiving time series (or ensemble) is modified. The time series to be added are selected using the AddTSList parameter:

- AllMatchingTSID - add all previous time series with matching identifiers.
- AllTS - add all previous time series.
- EnsembleID - add all time series for the ensemble.
- LastMatchingTSID - add the last time series (before this command) with matching identifier.
- SelectedTS - add time series selected with SelectTimeSeries() commands
- SpecifiedTSID - add time series selected from the list below

Time series to receive results: 0100501.DWR.DivTotal.Month

Ensemble to receive results:

Time series to add (AddTSList): SpecifiedTSID Indicates the time series to process (default=AllTS).

Add TSID (for TSList=AllMatchingTSID): 0100503.DWR.DivTotal.Month

Add EnsembleID (for AddTSList=EnsembleID):

Add specified TSID (for AddTSList=SpecifiedTSID): 0100501.DWR.DivTotal.Month
0100503.DWR.DivTotal.Month

Handle missing data how?: IgnoreMissing

Command:
Add(TSID="0100501.DWR.DivTotal.Month",AddTSList=SpecifiedTSID,AddTSID="0100503.DWR.DivTotal.Month",HandleMissingHow="IgnoreMissing")

Cancel OK

Add

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: <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> 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

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 08.15.00, 2008-05-04

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 appropriate for the data, use blank for all available data, OutputStart, or OutputEnd.

TS list: Indicates the time series to process (default=AllTS).

TSID (for TSList=AllMatchingTSID):

EnsembleID (for TSList=EnsembleID):

Constant value to add:

Analysis Period: to

Command:

```
AddConstant(TSList=AllMatchingTSID,TSID="2003536.DWR.ResMeasStorage.Day",ConstantValue=5000)
```

AddConstant

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: <ul style="list-style-type: none"> 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 08.16.04, 2008-09-15

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() Command

Adjust extreme data values by considering values to each side of extreme values.
If the Extreme to Adjust is AdjustMinimum, values \leq Extreme Value are adjusted.
If the Extreme to Adjust is AdjustMaximum, values \geq Extreme Value are adjusted.
The Average Adjust Method replaces the extreme and values on each side of the extreme by the average of all values, preserving the total.
The maximum intervals parameter indicates how many intervals on each side of the extreme can be modified.
Specify dates with precision appropriate for the data.

TS list: Indicates the time series to process (default=AllTS).

TSID (for TSList=AllMatchingTSID):

EnsembleID (for TSList=EnsembleID):

Adjust method:

Extreme to adjust:

Extreme value: Required.

Maximum intervals: Optional - default = 0, no limit.

Analysis start: Optional - default is all.

Analysis End: Optional - default is all.

Command:
`AdjustExtremes(TSList=AllMatchingTSID,TSID="06759000.USGS.Streamflow.Day",AdjustMethod=Average,ExtremeToAdjust=AdjustMinimum,ExtremeValue=0,MaxIntervals=0)`

AdjustExtremes

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: <ul style="list-style-type: none"> 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 processing an ensemble.	Required if TSList=EnsembleID.
AdjustMethod	Only the Average adjust method is implemented, in which adjusted data values are set to the average over the adjusted period, necessary 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 all values that make up the sum are then set to the average value. Missing values remain missing and therefore this 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.	None – must be specified.
ExtremeToAdjust	Indicate whether minimum (AdjustMinimum) or maximum (AdjustMaximum) values to be adjusted.	None – must be specified.
ExtremeValue	The extreme value that is the limit of acceptable values.	None – must be specified.
MaxIntervals	Indicates how many values on each side of a point are allowed to be examined.	0, indicating no limit.
AnalysisStart	The date/time to start analyzing data.	Full period.
AnalysisEnd	The date/time to end analyzing data.	Full period.

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.
# -e-b-----eb-----eb-----eb-----eb-----eb-----eb-----eb-----eb-----eb-----eb-----e
10/1908 - 9/1996 ACFT 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
...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 `AnalyzePattern()` command and illustrates the syntax of the command.

Edit AnalyzePattern() Command

This command creates the pattern file for use with the FillPattern() command.
Only monthly time series can be processed.
Example percentiles are .25,.75, with corresponding pattern identifiers DRY,AVG,WET.
The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\AnalyzePattern

TS list: Optional - indicates the time series to process (default=AllTS).

TSID (for TSList=AllMatchingTSID):

EnsembleID (for TSList=EnsembleID):

Method:

Percentile: Required - comma-separated list of fractions (0 to 1) for cutoffs.

PatternID: Required - pattern identifiers corresponding to the fractions.

Output file:

Table ID: Optional - identifier for table to create, containing statistics.

Row(s) for data: Insert: Optional - data row name(s) for 1+ time series.

Legacy behavior: Optional - use legacy logic (error with some edge values shifted to lower percentile).

Command:

```
AnalyzePattern(TSList=AllTS,Method=Percentile,Percentile="0.25,0.75",PatternID="DRY,AVG,WET",OutputFile="Div1.pat",TableID="Statistics",DataRow="%L, %U")
```

AnalyzePattern

AnalyzePattern() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none"> 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.

	<ul style="list-style-type: none"> SelectedTS – the time series selected with the <code>SelectTimeSeries()</code> 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 <code>TSList=*TSID</code> .
EnsembleID	The ensemble to be processed, if processing an ensemble.	Required if <code>TSList=EnsembleID</code> .
Method	Method used to determine the patterns. Currently only Percentile is recognized.	Percentile
Percentile	A comma-separated list of percentiles for cutoffs, used when <code>Method=Percentile</code> . 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:

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 value > 0.75	Apr last value < 0.25	Apr first value > 0.75	May last value < 0.25
06720500, ACFT	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

AnalyzePatter_Table

Command Reference: ARMA()

Lag and attenuate a time series using AutoRegressive Moving Average

Version 08.16.04, 2008-09-15

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 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 $t - 1$ 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

Lag and attenuate a time series using the ARMA (AutoRegressive Moving Average) method.

The adjusted output time series O is computed from the original input I using:

$$O[t] = a_1 \cdot O[t-1] + a_2 \cdot O[t-2] + \dots + a_p \cdot O[t-p] + b_0 \cdot [t] + b_1 \cdot [t-1] + \dots + b_q \cdot [t-q]$$

where t = time, p = number of outflows to consider, and q = number of inflows to consider

ARMA a and b coefficients must be computed externally and should sum to 1.0.

The values for p and q will be determined from the number of coefficients.

Specify the interval used to compute ARMA coefficients as 1Day, 6Hour, 2Hour, etc.

The ARMA interval must be <= the time series interval.

The resulting value is set to missing if one or more input values are missing (typically only filled data should be used). The period will not automatically be extended.

TS list: **AllMatchingTSID** Indicates the time series to process (default=AllTS).

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=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")**

Cancel OK

ARMA

ARMA() Command Editor

The command syntax is as follows:

ARMA (Parameter=Value,...)

Command Parameters

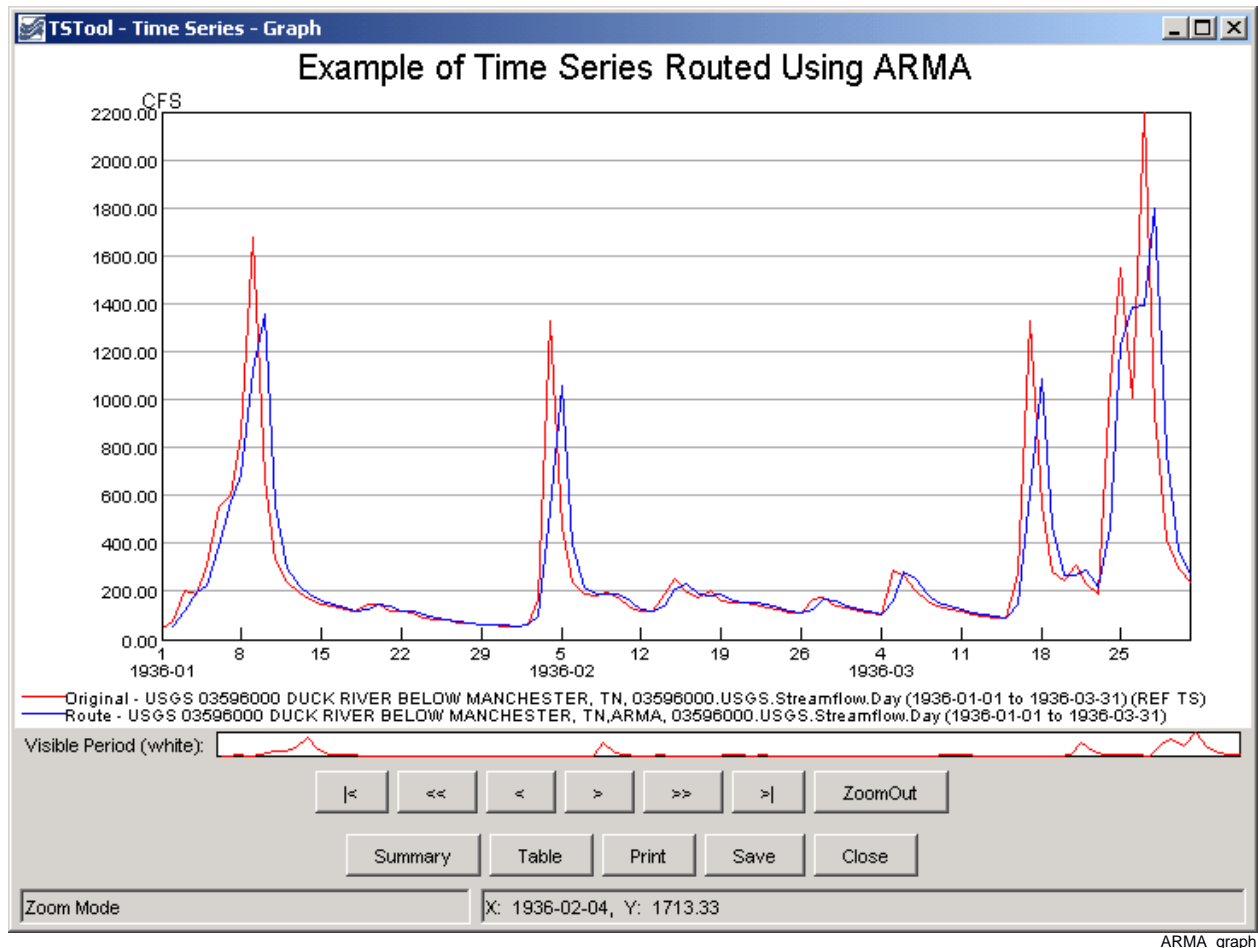
Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none"> AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will be modified. AllTS – all time series before the command. EnsembleID – all 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. 	AllTS

Parameter	Description	Default
	<ul style="list-style-type: none"> 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.
ARMA Interval	The ARMA interval to use in the analysis	None – must be 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")
TS Original = ReadUsgsNwis(InputFile="Data/G03596000.in1")
TS Routed = Copy(TSID="Original",NewTSID="03596000.USGS.Streamflow.Day.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.

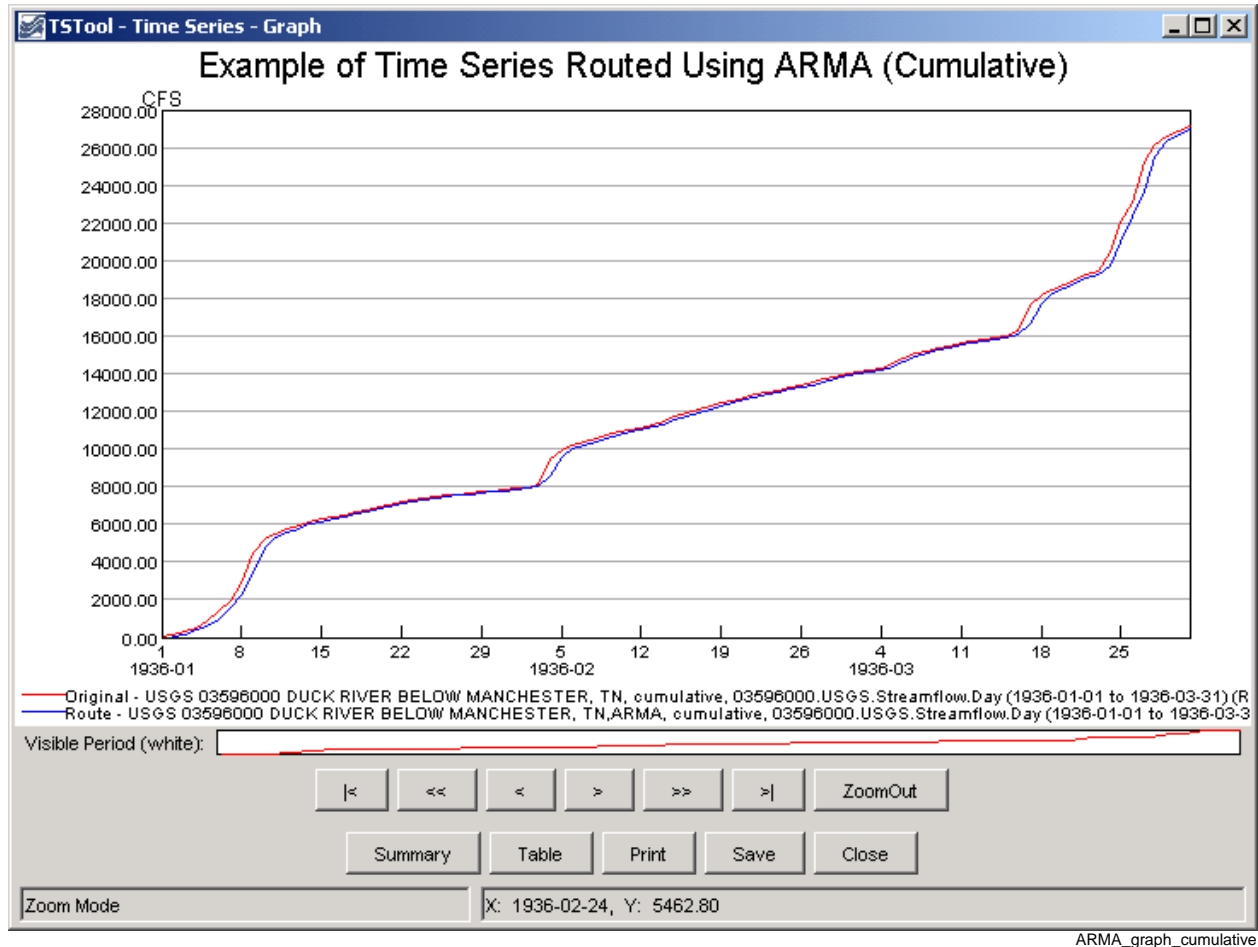


ARMA_graph

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() command

Blend one time series into the start or end of another.
The BlendAtEnd blend method will use data from the second (independent) time series at the end of the first.
The overall period will be that of both time series.
Additional blend methods (e.g., interpolating the blend) may be added in the future.
See also the SetFromTS() and Add() commands.

Time series to modify: lobatos_current

Independent time series: lobatos_likely

Blend method: BlendAtEnd

Command: Blend(TSID="lobatos_current",IndependentTSID="lobatos_likely",BlendMethod=BlendAtEnd)

Cancel OK

Blend

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 time series to be modified.	None – must be specified.
IndependentTSID	The time series identifier or alias for the time series to be blended to the first time series.	None – must be specified.
BlendMethod	The method used to blend the data, one of: <ul style="list-style-type: none"> BlendAtEnd, resulting in the main time series having the other time series attached to the end of its period. 	None – must be specified. Currently only BlendAtEnd is recognized.

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 09.08.01, 2010-09-15

The `CalculateTimeSeriesStatistic()` command calculates a statistic for a time series (typically a single value) and optionally adds the result to a table (see the `NewTable()` command). Multiple time series can be processed. The sample from each time series consists of data values for the full period or a shorter period if specified for the command. Missing values are typically 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 CalculateTimeSeriesStatistic() Command

Calculate a statistic for time series and optionally save in a table.
Statistics results may have 1+ values and may include the date/time of the result.
Use table commands to save the results.
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):

EnsembleID (for TSList=EnsembleID):

Statistic to calculate: Required - may require other parameters.

Value1: Optional - may be needed as input to calculate statistic.

Value2: Optional - may be needed as input to calculate statistic.

Value3: Optional - may be needed as input to calculate statistic.

Analysis period: to

Table ID for output: Optional - if statistic should be saved in table.

Table TSID column: Required if using table - column name for TSID.

Format of TSID: Insert: Optional - use %L for location, etc. (default=alias or TSID).

Table statistic column: Required if using table - column name for statistic.

Command:
`CalculateTimeSeriesStatistic (Statistic="NqYY", Value1=7, Value2=10, Value3=0, TableID="Table1", TableTSIDColumn="TSID", TableStatisticColumn="7q10")`

CalculateTimeSeriesStatistic

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 of: <ul style="list-style-type: none"> 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. 	AllTS
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.
Statistic	Statistic to compute, one of the following: <ul style="list-style-type: none"> Count – number of data values total, including missing and non-missing DeficitMax – the maximum deficit value (where deficit is mean minus value) DeficitMean – the mean deficit value (where deficit is mean minus value) DeficitMin – the minimum deficit value (where deficit is mean minus value) DeficitSeqLengthMax – the maximum number of sequential intervals where each value is less than the mean (for example maximum drought length) DeficitSeqLengthMean – the mean number of sequential intervals where each value is less than the mean (for example mean drought length) DeficitSeqLengthMin – the minimum number of sequential intervals where each value is less than the mean (for example minimum drought length) DeficitSeqMin – the maximum sum of sequential values where each value is less than 	None – must be specified.

Parameter	Description	Default
	<p>the mean (for example maximum drought water volume)</p> <ul style="list-style-type: none"> DeficitSeqMean – the mean of the sum of sequential values where each value is less than the mean (for example mean drought water volume) DeficitSeqMin – the minimum sum of sequential values where each value is less than the mean (for example minimum drought water volume) Lag-1AutoCorrelation – the 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 Max – maximum value Mean – mean value Min – minimum value MissingCount – number of missing values MissingPercent – percent of values that are missing NonmissingCount – number of non-missing values NonmissingPercent – percent of values that are not missing NqYY – restricted to daily data and typically used to analyze return interval of low flows, requires values of N, YY, and number of missing allowed to be specified with Value parameters (see Statistic Details table below) Skew – skew coefficient, as follows: $Cs = \frac{N \sum_{i=1}^N (Y_i - Y_{mean})^3}{(n-1)(n-2)s^3}$ <p>where s = standard deviation</p> StdDev – standard deviation SurplusMin – the maximum surplus value (where surplus is value minus mean) SurplusMean – the mean surplus value (where surplus is value minus mean) SurplusMin – the minimum surplus value (where surplus is value minus mean) SurplusSeqLengthMax – the maximum number of sequential intervals where each value is greater than the mean (for example maximum water surplus length) SurplusSeqLengthMean – the mean number of sequential intervals where each value 	

Parameter	Description	Default
	<p>is greater than the mean (for example mean water surplus length)</p> <ul style="list-style-type: none"> SurplusSeqLengthMin – the minimum number of sequential intervals where each value is greater than the mean (for example minimum water surplus length) SurplusSeqMin – the maximum sum of sequential values where each value is greater than the mean (for example maximum water surplus volume) SurplusSeqMean – the mean of the sum of sequential values where each value is greater than the mean (for example mean water surplus volume) SurplusSeqMin – the minimum sum of sequential values where each value is greater than the mean (for example minimum water surplus volume) Variance – variance 	
Value1	Input data required by the statistic. Currently the dialog does not check the value for correctness – it is checked when the statistic is computed.	See Statistic Details table below.
Value2	Input data required by the statistic. Currently the dialog does not check the value for correctness – it is checked when the statistic is computed.	See Statistic Details table below.
Value3	Input data required by the statistic. Currently the dialog does not check the value for correctness – it is checked when the statistic is computed.	See Statistic Details table below.
AnalysisStart	The date/time to start analyzing data.	Full period is analyzed.
AnalysisEnd	The date/time to end analyzing data.	Full period is analyzed.
TableID	Identifier for table that receives the statistic.	Optional – table output is not 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.

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
NqYY	<p>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:</p> <ol style="list-style-type: none"> 1. Determine the number of years to be analyzed (from analysis period command parameters or time series data). 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. 3. For the year, save the minimum 7-day average. 4. Utilize the minimum values for all years, with log-Pearson Type III distribution, to determine the value for the 10-year recurrence interval. See http://pubs.usgs.gov/sir/2008/5126/section3.html for a description of NqYY and “Hydrology for Engineers, 3rd Edition,” Linsley, Kohler, Paulhus for a description of log-Pearson Type III distribution. 	<p>Value1 – specify the number of daily values to be averaged. Currently this must be an odd number to allow bracketing the current day.</p> <p>Value2 – specify the return interval (e.g., 10).</p> <p>Value3 – specify the number of 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.</p>
All other statistics	Described above.	No additional input values are needed.

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

```
TS linsley = ReadDateValue(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")
```

This page is intentionally blank.

Command Reference: TS Alias = ChangeInterval()

Create a new time series by changing a time series data interval

Version 09.06.03, 2010-04-14

A `ChangeInterval()` command creates a new time series by changing the data interval of an existing time series. The majority of the original header data (e.g., description, units) are copied to the new time series. Time series data values have a time scale of accumulated (e.g., volume), mean, or instantaneous. Changing the interval can also 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.

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 is not currently 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, Hour 0 (A)	Day 1, Hour 1 (B)	Day 1, Hour 2 (Missing)	Day 1, Hour 3 (C)	Day 1, Hour 4 (Missing)	Day 1, Hour 5 (Missing)	Day 1, Hour 6 (Missing)
Day 1, Hour 0 =A	Day 1, Hour 3 =B+C			Day 1, Hour 6 = Missing		

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.

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

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

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

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, Hour 0 (A)	Day 1, Hour 3 (B)			Day 1, Hour 6 (Missing)			Day 1, Hour 9 (C)		
Day 1, Hour 0 =A	Day 1, Hour 1 =B	Day 1, Hour 2 =B	Day 1, Hour 3 =B	Day 1, Hour 4 =C	Day 1, Hour 5 =C	Day 1, Hour 6 =C	Day 1, Hour 7 =C	Day 1, Hour 8 =C	Day 1, Hour 9 =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, Hour 0 (A)	Day 1, Hour 1 (B)	Day 1, Hour 2 (Missing)	Day 1, Hour 3 (C)	Day 1, Hour 4 (Missing)	Day 1, Hour 5 (D)	Day 1, Hour 6 (Missing)	Day 1, Hour 7 (E)	Day 1, Hour 8 (F)
Day 1, Hour 0 =A			Day 1, Hour 3 =C			Day 1, Hour 6 =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, Hour 0 (A)	Day 1, Hour 3 (B)			Day 1, Hour 6 (Missing)			Day 1, Hour 9 (C)		
Day 1, Hour 0 =A	Day 1, Hour 1 =A+ (B-A)* (1/3)	Day 1, Hour 2 =A+ (B-A)* (2/3)	Day 1, Hour 3 =B	Day 1, Hour 4 =B+ (C-B)* (1/6)	Day 1, Hour 5 =B+ (C-B)* (2/6)	Day 1, Hour 6 =B+ (C-B)* (3/6)	Day 1, Hour 7 =B+ (C-B)* (4/6)	Day 1, Hour 8 =B+ (C-B)* (5/6)	Day 1, Hour 9 =C

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, Hour 0	Day 1, Hour <i>i</i>	Day 1, Hour 2 <i>i</i>	Day 1, Hour 3 <i>i</i>	Day 2, Hour 0
Day 1, Hour 0	Day 1, Hour 6 (A)	Day 1, Hour 12 (B)	Day 1, Hour 18 (C)	Day 2, 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, Day 1 (A1)	Month 1, Day 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 accumulate (A)			
Day 1, Hour 0	Day 1, Hour i	Day 1, Hour $2i$	Day 1, Hour $3i$
Day 1, Hour 0 = A/4	Day 1, Hour 6 = A/4	Day 1, Hour 12 = A/4	Day 1, Hour 18 = 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, Hour 0	Day 1, Hour i	Day 1, Hour $2i$	Day 1, Hour $3i$
Day 1, Hour 0 (A)	Day 1, Hour 6 (B)	Day 1, Hour 12 (C)	Day 1, Hour 18 (D)
Day 1, Hour 0 =A	Day 1, Hour 6 =B	Day 1, Hour 12 =C	Day 1, Hour 18 =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, Hour 0	Day 1, Hour i	Day 1, Hour $2i$	Day 1, Hour $3i$	Day 1, Hour 0	Day 1, Hour i	Day 1, Hour $2i$	Day 1, Hour $3i$
Day 1, Hour 0 (A)	Day 1, Hour 6 (B)	Day 1, Hour 12 (C)	Day 1, Hour 18 (D)	Missing data	Day 1, Hour 6 (E)	Day 1, Hour 12 (F)	Day 1, Hour 18 (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 instantaneous (A)				Day 2 instantaneous (B)				Day 3 ...
Day 1, Hour 0	Day 1, Hour i	Day 1, Hour $2i$	Day 1, Hour $3i$	Day 2, Hour 0	Day 2, Hour i	Day 2, Hour $2i$	Day 2, Hour $3i$	
Day 1, Hour 0 =A	Day 1, Hour 6 =A+ (B-A)* (6/24)	Day 1, Hour 12 =A+ (B-A)* (12/24)	Day 1, Hour 18 =A+ (B-A)* (18/24)	Day 2, Hour 0 =B	Day 2, Hour 6 ...	Day 2, Hour 12 ...	Day 2, Hour 18 ...	
These values are an interpolated value between the Day 1 instantaneous value and the Day 2 instantaneous value using a time of 24 hours.				These values are an interpolated value between the Day 2 instantaneous value 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, Hour 0	Day 1, Hour i	Day 1, Hour $2i$	Day 1, Hour $3i$	Day 2, Hour 0	Day 2, Hour i	Day 2, Hour $2i$	Day 2, Hour $3i$
Day 1, Hour 0	Day 1, Hour 6	Day 1, Hour 12	Day 1, Hour 18	Day 2, Hour 0	Day 2, Hour 6	Day 2, Hour 12	Day 2, Hour 18
Value A	B	C	D	E	F	G	H
Day 1 mean= (A+B+C+D)/4				Day 2 mean=(E+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):

Day 1, Hour 0	Day 1, Hour i	Day 1, Hour $2i$	Day 1, Hour $3i$	Day 2, Hour 0	Day 2, Hour i	Day 2, Hour $2i$	Day 2, Hour $3i$	
Day 1, Hour 0	Day 1, Hour 6	Day 1, Hour 12	Day 1, Hour 18	Day 2, Hour 0	Day 2, Hour 6	Day 2, Hour 12	Day 2, Hour 18	
Value A	B	C	D	E	F	G	H	I
Day 1 mean= $((A+E)/2 + B+C+D) / 4$				Day 2 mean= $((E+I)/2 + F+G+H) / 4$				

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, Hour 0	Day 1, Hour i	Day 1, Hour $2i$	Day 1, Hour $3i$	Day 2, Hour 0
Day 1, Hour 0 (A)	Day 1, Hour 6 (B)	Day 1, Hour 12 (Missing)	Day 1, Hour 18 (D)	Day 2, Hour 0 (E)
Day 1, Hour 0 =A	Day 1, Hour 6 =B	Day 1, Hour 12 =B	Day 1, Hour 18 =D	Day 2, Hour 0 =E

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, Hour 0	Day 1, Hour i	Day 1, Hour $2i$	Day 1, Hour $3i$
Day 1, Hour 0 =A	Day 1, Hour 6 =A	Day 1, Hour 12 =A	Day 1, Hour 18 =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 =A	Day 2 =A	...	Day 30 =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)

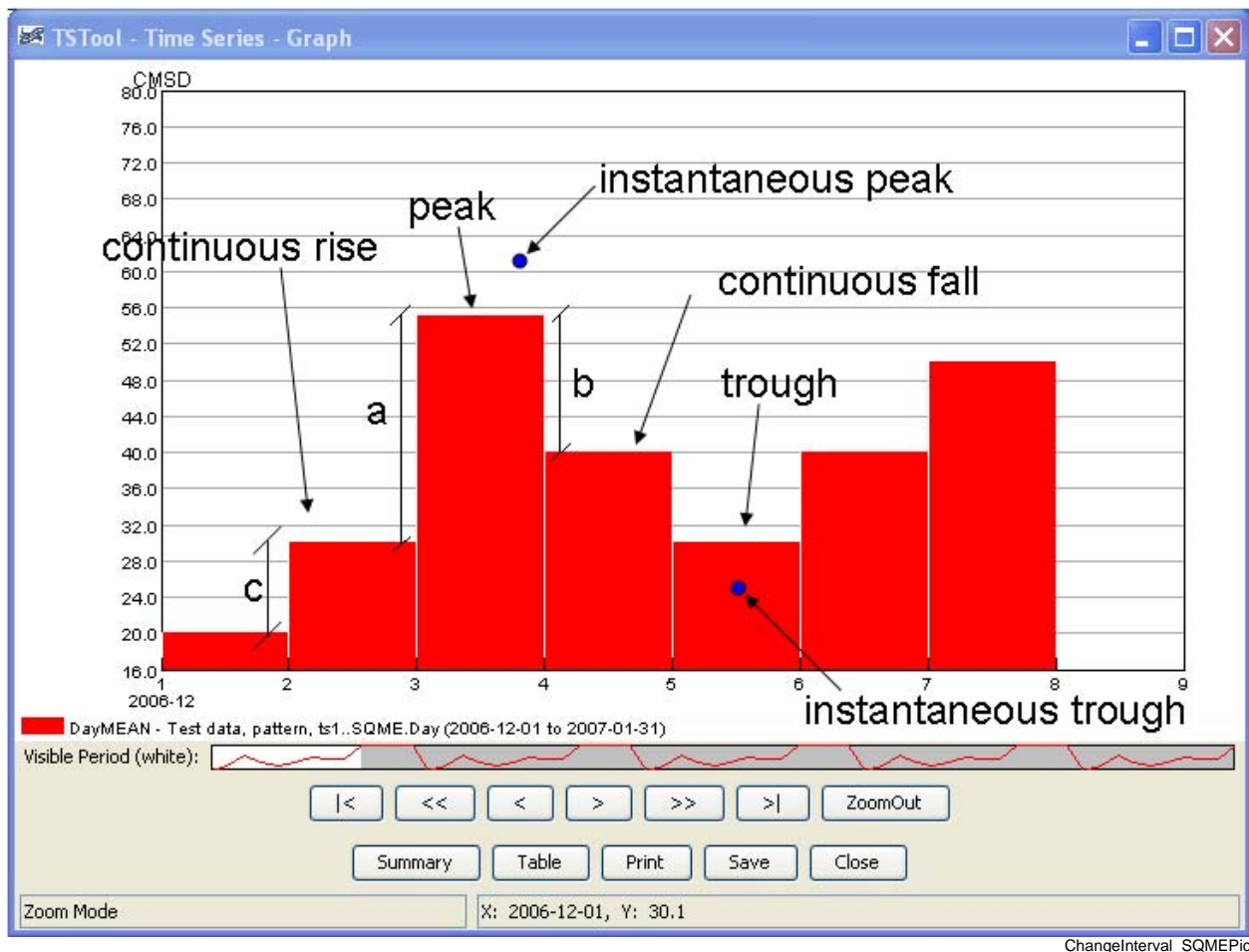
Changing the interval for large interval mean to small interval instantaneous data involves calculating a value for each new interval based on trends found in the mean data. The following example demonstrates how the data is converted from the old interval to the new interval. A general representation 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):

Day 1 mean			
Day 1, Hour 0	Day 1, Hour i	Day 1, Hour $2i$	Day 1, Hour $3i$
Day 1, Hour 0	Day 1, Hour 6	Day 1, Hour 12	Day 1, Hour 18

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. The following rules are applied when converting from large interval mean to small interval instantaneous:

- 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.



Mean data illustration

- Another 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 value for the start of the interval is set to the *current value minus $\frac{1}{4} a$* . The value for the end of the interval is set to the *current value minus $\frac{1}{4} b$* .
 3. The remaining instantaneous values for the interval are linearly interpolated between the peak (or trough) and both endpoints.
- A final 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 this case, the instantaneous value at the start of the interval is set to the *current value minus $\frac{1}{4} c$* (again using the above illustration).
 2. The instantaneous value at the end of the larger interval is set to the *current value plus $\frac{1}{4} a$* .
 3. The values c and a are calculated. If c is less than a , then the simulated 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.

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. This tolerance is specified with the *Tolerance* parameter. The volume for each interval uses the average of the first and last endpoint.

This approach has been adapted from the NWSRFS CHANGE-T operation (see http://www.nws.noaa.gov/oh/hrl/nwsrfs/users_manual/part5/_pdf/533changet.pdf).

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 ChangeInterval() Command

Create a new time series by changing the data interval of an existing time series.
 Use the alias to reference the new time series - other time series information will be copied from the original.
 The conversion process depends on whether the original and new time series contain accumulated, mean, or instantaneous data.
 The time scales must be specified (they are not automatically determined from the data type).
 Many of the advanced parameters depend on the data interval, which can only be confirmed at runtime, and are mainly for intervals less than a day - see the documentation.

Time series alias:

Time series to convert:

New interval: Required - data interval for result.

Old time scale: Required - indicates how to process data.

New time scale: 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: 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 - how to handle interval endpoint values in hourly or finer input (default=AverageEndpoints).

Allow missing count: Optional - number of missing values allowed in input interval (default=0).

Output fill method: Optional - use when converting from large to small interval (default=Repeat).

Handle missing input how?: Optional - indicate how to handle missing values in input (default=KeepMissing).

TS New =
 ChangeInterval (TSID="Original",NewInterval=Year,OldTimeScale=ACCM,NewTimeScale=MEAN,OutputYearType=Water)

Command:

ChangeInterval

ChangeInterval() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
Alias	The alias to assign to the new time series. The time series identifier for the new time series will be the same as the original time series, with the new interval and optionally a new data type (see the NewDataType parameter).	None – must be specified.
TSID	The time series identifier or alias for the original (old) time series.	None – must be specified.
NewInterval	The data interval for the new time series, from the provided choices. For example: 6Hour, Day, Month, Year.	None – must be specified.
OldTimeScale	The time scale for the original time series, one of: ACCM – accumulated data INST – instantaneous data MEAN – mean data	None – must be specified.

	In the future, this parameter may be made optional if the time scale can be determined from the data type.	
NewTimeScale	The time scale for the new time series (see OldTimeScale for possible values). In the future, this parameter may be made optional if the time scale can be determined from the data type.	None – must be specified.
Statistic	Used in the case where INST (small interval) to INST (large interval) conversion is occurring. A sample of values from the input time series, corresponding to the output interval, is determined and used to compute a statistic instead of a simple value transfer. Statistics that are currently supported are MAX and MIN. The HandleMissingInputHow parameter is initially used to adjust missing data and then the AllowMissingCount parameter is used to check whether the statistic can be computed.	The statistic is determined from the old and new time scales.
OutputYearType	The output year type if the output time series has an interval of Year. The output year type can only be specified for input time series having an interval of 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 parameter is recognized.	Calendar
NewDataType	The data type for the new time series. This will be set in the identifier of the new time series.	Use the data type from the original time series.
NewUnits	The units for the new time series. This will be set in the identifier of the new time series.	Use the units from the original time series.
Tolerance	Currently used when converting large interval MEAN 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.	0.01
HandleEndpointsHow	Indicates how endpoints should be handled when 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	Average Endpoints
AllowMissingCount	The number of missing values allowed in the source	0 – do not allow

	interval(s) in order to produce a result. For example, if converting daily data to monthly, a value of 5 would allow ≤ 5 missing daily values and still compute the result. This capability should be used with care because it may result in data that are not representative of actual conditions.	any missing data in the source data when computing a result.
OutputFill Method	Use to fill output when converting from INST to MEAN, large interval time series to small interval time series, one of: Interpolate – linearly interpolate Repeat – repeat values for the output	Repeat
HandleMissing InputHow	Indicate how to handle missing values in input, one of: 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: <ul style="list-style-type: none"> • Irregular data • Day and Month input converted to ACCM and MEAN. 	KeepMissing

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

```
0109.NOAA.Precip.Day~HydroBase
TS 0109Month =
ChangeInterval(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:

```
TS DayMEAN = NewPatternTimeSeries(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")

TS 6HourINST =
ChangeInterval(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:

```
TS 6HourInst =
NewPatternTimeSeries(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")

TS DayMean2 =
ChangeInterval(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:

```
TS IrregularINST =  
NewPatternTimeSeries(NewTSID="ts1..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")  
  
TS 3HourINST =  
ChangeInterval(TSID="IrregularINST",NewInterval=3Hour,OldTimeScale=INST,NewTimeScale=INST)
```

Command Reference: ChangePeriod()

Change period of record for time series

Version 08.16.04, 2008-09-15

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() Command

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: Indicates the time series to process (default=AllTS).

TSID (for TSList=AllMatchingTSID):

EnsembleID (for TSList=EnsembleID):

New start date/time: Blank if only end is being adjusted.

New end date/time: Blank if only start is being adjusted.

Command:

ChangePeriod

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: <ul style="list-style-type: none"> 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 processing an ensemble.	Required if TSList=EnsembleID.
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 - ALAMOS 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

Version 09.07.01, 2010-08-18

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. Matched values can also be removed (if irregular interval), or set to missing. The `WriteCheckFile()` command can be used to write a summary of the warnings. 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 reset , 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 values and statistics for critical values.
A warning will be generated for each case where a value matches the specified condition(s).
Use the `WriteCheckFile()` command to save the results of all checks.
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):

EnsembleID (for TSList=EnsembleID):

Value to check: Optional - check data values or statistic? (default=DataValue). **Statistic is not enabled.**

Check criteria: 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 period).

Analysis end: Optional - analysis end date/time (default=full time series period).

Problem type: Optional - problem type to use in output (default=check criteria).

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: Optional - action for matched values (default=no action).

Command:
`CheckTimeSeries(TSList=AllMatchingTSID,TSID="ts2",CheckCriteria="Missing",Action=Remove)`

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: <ul style="list-style-type: none"> 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. 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 processed, 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.
ValueToCheck	One of the following: <ul style="list-style-type: none"> DataValue to indicate that raw data values should be checked. Statistic to indicate that a statistic computed from data values should be checked (future enhancement). 	DataValue.
CheckCriteria	The criteria that is checked, one of: <ul style="list-style-type: none"> 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 > Value1. 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 values that are the same as the previous value. < – check for values < Value1. <= – check for values <= Value1. > – check for values > Value1. >= – check for values >= Value1. == – check for values equal to Value1. 	None – must be specified.

Parameter	Description	Default
Value1	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 messages.	CheckCriteria
MaxWarnings	The maximum number of warnings to list for each time series, useful if analysis results in many warnings.	List all warnings.
Flag	A string to use for a flag on values that are detected during the check, which will be shown in the HTML summary report.	No flag.
FlagDesc	Description for the flag.	No description.
Action	Action to take for matched values, in addition to generating warnings: <ul style="list-style-type: none"> 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. 	No action is taken.

This page is intentionally blank.

Command Reference: CompareFiles()

Compare text files to determine whether they are different

Version 09.07.00, 2010-06-14

The `CompareFiles()` command compares text files to determine data differences. For example, the command can be used to compare old and new files produced by a software process.

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. 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()` command).

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

Edit CompareFiles() command

This command compares text files. Comment lines starting with # are ignored.
A line by line comparison is made.
It is recommended that file names be relative to the working directory, which is:
C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\CompareFiles

First file to compare:

Second file to compare:

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

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

Action if different: 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", IfDifferent=Warn)`

CompareFiles

CompareFiles() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
InputFile1	The name of the first file to read. Enclose the name in double quotes to protect whitespace and special characters.	None – the file name is required.
InputFile2	The name of the second file to read. Enclose the name in double quotes to protect whitespace and special characters.	None – the file name is required.
CommentLineChar	The character(s) that if found at the start of a line indicate comment lines. Comment lines are ignored in the comparison because they typically may include information such as date/time that changes even if the remainder of the file contents are the same.	#
AllowedDiff	The number of lines allowed to be different, when checking for differences. This is useful, for example, when a non-comment line contains the date/time when the file was generated.	0
WarnIfDifferent	If True and at least one difference is detected, a warning will be generated by the command, which will result in software like TSTool displaying a warning. If False , only status messages are written to the log file. The warning is useful if it is critical to detect any difference in the files.	Do not generate a warning if the files are different. Differences are printed to the log file.
WarnIfSame	If True and no differences are detected, a warning will be generated by the command, which will result in software like TSTool displaying a warning. If False , only status messages are written to the log file. The warning is useful if it is critical to detect files that are the same.	Do not generate a warning if the files are the same.

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

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

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 `RunCommands()` commands).

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

Edit CompareTimeSeries() command

This command compares time series. Currently all available time series are evaluated, comparing time series that have the same time series identifier location and/or data type.

Specify one or more tolerances, separated by commas. Differences greater than these values will be noted.

Match location:	<input type="button" value="v"/>	Match location to find time series pair? (default=true)
Match data type:	<input type="button" value="v"/>	Match data type to find time series pair? (default=false)
Precision:	<input type="text" value="2"/>	Precision for comparison (digits after decimal).
Tolerance:	<input type="text"/>	Tolerance(s) to indicate difference (e.g., .01, .1).
Analysis period:	<input type="text"/>	to <input type="text"/>
Difference flag:	<input type="text"/>	1-character flag to use for values that are different.
Create difference time series?:	<input type="button" value="v"/>	Create a time series TS1 - TS2? (default=false)
Warn if different?:	<input checked="checked" type="button" value="v"/>	Generate a warning if different? (default=false)
Warn if same?:	<input type="button" value="v"/>	Generate a warning if same? (default=false)

Command:

```
CompareTimeSeries(Precision=2,WarnIfDifferent=True)
```

CompareTimeSeries

CompareTimeSeries() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
MatchLocation	Match the location part of time series identifiers when matching time series to compare.	True
MatchDataType	Match the data type part of time series identifiers when matching time series to compare.	False
Precision	When comparing data values, round the values to the given precision. For example, a precision of 2 will round to the hundredths place. This can be used to do comparisons on the lowest precision of the available time series.	Compare the available values without rounding.
Tolerance	Specify a comma-separated list of values. The difference in the time series values will be compared to the tolerances and messages printed to the log file.	A tolerance of zero will be used to detect differences.
AnalysisStart	The starting date/time to analyze for differences. Specify a date/time of appropriate precision for the time series or OutputStart to use the output start.	Analyze all available data.
AnalysisEnd	The ending date/time to analyze for differences. Specify a date/time of appropriate precision for the time series or OutputEnd to use the output end.	Analyze all available data.
DiffFlag	Specify as a single character to append a flag to the 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 .	Do not flag data.
CreateDiffTS	Indicate whether a time series should be created containing the differences between time series. This is useful to visually evaluate the differences and process the results with other commands.	False
WarnIfDifferent	If True and at least one difference is detected, a warning will be generated by the command, which will result in software like TSTool displaying a warning. If False, only status messages are written to the log file. The warning is useful if it is critical to detect any change in the time series.	Do not generate a warning if time series are different. Differences are printed to the log file.
WarnIfSame	If True and no differences are detected, a warning will be generated by the command, which will result in software like TSTool displaying a warning. If False, 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.	Do not generate a warning if 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 08.15.00, 2008-05-12

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.

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: Indicates the time series to process (default=AllTS).

Observed TSID (for ObservedTSList=AllMatchingTSID):

Observed ensembleID (for ObservedTSList=EnsembleID):

Simulated TS List: Indicates the time series to process (default=AllTS).

Simulated TSID (for Simulated TSList=AllMatchingTSID):

Simulated EnsembleID (for Simulated TSList=EnsembleID):

Error measure:

Alias to assign: Default is no alias is assigned.

Command:
`ComputeErrorTimeSeries(ObservedTSList=AllMatchingTSID,ObservedTSID="ts1",SimulatedTSList=AllMatchingTSID,SimulatedTSID="ts2",SimulatedEnsembleID="PercentError",ErrorMeasure=PercentError)`

ComputeErrorTimeSeries

ComputeErrorTimeSeries() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
Observed TSList	Indicates the list of observed time series to be processed, one of: <ul style="list-style-type: none">AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will be modified.	AllTS

Parameter	Description	Default
	<ul style="list-style-type: none"> AllTS – all time series before the command. EnsembleID – all time series in the ensemble will be compared. LastMatchingTSID – the last time series that matches the TSID (single TSID or TSID with wildcards) will be compared. 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 SimulatedddTSList=EnsembleID
ErrorMeasure	The error measure to compute, one of: <ul style="list-style-type: none"> PercentError – Simulated minus observed, divided by observed. AbsoluteError – not yet implemented. 	
Alias	The alias to be assigned to each trace in the ensemble. The string can include: <ul style="list-style-type: none"> % specifiers from the LegendFormat property (see the TSView Time Series Viewing Tools appendix). \${Property} strings, where Property is a value set internally by the command processor (more documentation will be provided in the future) or with the SetProperty() command. This approach is useful if the TSTool command file is dynamically created with a script. Any literal characters. 	None.

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

```
RemoveFile(InputFile="Results\Test_ComputeErrorTimeSeries_1_out.dv",WarnIfMissing=False)
TS ts1 = NewPatternTimeSeries(NewTSID="ts1..test.Day",Description="Test data",
    SetStart="1950-01-01",SetEnd="1951-03-12",Units="CFS",PatternValues="5,10,12,13,75")
TS ts2 = NewPatternTimeSeries(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="ExpectedResults\Test_ComputeErrorTimeSeries_1_out.dv")
WriteDateValue(OutputFile="Results\Test_ComputeErrorTimeSeries_1_out.dv")
CompareFiles(InputFile1="Results\Test_ComputeErrorTimeSeries_1_out.dv",
    InputFile2="ExpectedResults\Test_ComputeErrorTimeSeries_1_out.dv",WarnIfDifferent=True)
```

Command Reference: ConvertDataUnits()

Convert time series data units

Version 08.15.00, 2008-05-04

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.

Edit ConvertDataUnits() Command

The units of the selected time series will be converted to the 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: Indicates the time series to process (default=AllTS).

TSID (for TSList=AllMatchingTSID):

EnsembleID (for TSList=EnsembleID):

Dimension: Select the dimension first, to list corresponding units.

New data units:

Command:
`ConvertDataUnits(TSList=AllMatchingTSID,TSID="08236000.DWR.Streamflow.Month",NewUnits="CFSD")`

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.

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: <ul style="list-style-type: none"> 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: TS Alias = Copy()

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

Version 09.08.01, 2010-09-14

The `TS Alias = 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 should also be specified 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 the original time series needs to be maintained for graphing or other purpose.

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

Edit TS Alias = Copy() Command

Make a copy of a time series, giving the copy an alias.
The copy is exactly the same and can be referenced by its alias in other commands.
Specify new time series identifier (TSID) information for the copy to avoid errors with the copy being mistaken for the original.

Time series alias: Required - often the location from the TSID or a short string.

Time series to copy:

New time series ID: Required - specify to avoid confusion with TSID from original time series.

Command:
`TS Filled =
Copy(TSID="08223000.DWR.Streamflow.Month",NewTSID="08223000.DWR.Streamflow.Month.Filled")`

Copy

TS Alias = Copy() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
Alias	The alias to assign to the new copy, which can be used instead of the TSID by other commands.	None – must be specified.
TSID	The time series identifier or alias of the time series to copy. The time series will be found by searching backwards from the copy command.	None – must be specified.
NewTSID	A new time series identifier to assign to the copy. This is useful to avoid confusion with the original time series. Use the Edit button to edit the time series identifier parts. The data interval must match that of the original time series.	Copy the original time series TSID. If NewTSID is specified but does not have a valid interval, copy the interval from TSID. The default cannot be determined if an alias is used for the input time series.

A sample commands file is as follows:

```
# 08223000 - RIO GRANDE RIVER AT ALAMOSA
08223000.DWR.Streamflow.Month~HydroBase
TS Filled = Copy(TSID="08223000.DWR.Streamflow.Month",
NewTSID="08223000.DWR.Streamflow.Month.Filled")
```

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 be 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 series ensemble, giving the copy a new identifier.
The copy is exactly the same and can be referenced by its identifier in other commands.
Because time series in the ensemble are copies of time series from the original ensemble, the Ensemble ID should be used for processing time series.
Optionally, specify new time series identifier (TSID) information for the time series in the copy: location, source, data type, and/or scenario.
This is highly recommended if there is any chance that the copy will be mistaken for the original.

New ensemble identifier:

New ensemble name: Optional name for copy.

Ensemble to copy:

New time series ID parts: Specify to avoid confusion with TSID from original TS.

Command:

```
CopyEnsemble(NewEnsembleID="Ensemble_2",NewEnsembleName="Test ensemble 2",NewTSID="09019500.USGS.Streamflow..copy",EnsembleID="Ensemble_1")
```

CopyEnsemble

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

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",
  NewTSID="09019500.USGS.Streamflow..copy",EnsembleID="Ensemble_1")
```

Command Reference: CopyTable()

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

Version 09.09.00, 2010-09-23

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.

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).

Edit CopyTable() Command

This command creates a new table by copying another table.
For example, single column tables can be created from a larger table to use as a list.

Table ID: Required - original table.

New table ID: Required - unique identifier for the new table.

Column names to copy: Optional - names of columns to copy (default=copy all).

Command:

```
CopyTable (TableID="Table1", NewTableID="Table1Copy", IncludeColumns="LocationID")
```

CopyTable

CopyTable() Command Editor

The command syntax is as follows:

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

Command Parameters

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	Specify the names of columns to copy, separated by commas.	Copy all of the columns.

Command Reference: CreateEnsembleFromOneTimeSeries()

Create a new ensemble from a single time series

Version 09.05.00, 2009-10-06

This command was previously named `CreateEnsemble()`.

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 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 from a single time series, for example to split the time series into overlapping historical traces.
Each trace will start on the reference date and will be as long as specified.
Each trace will have the properties of the original time series with unique sequence numbers.
Specify the period to limit the number of traces generated from the original time series.
Specify the reference date using standard date formats to a precision appropriate for the data.
If shifted, each trace will start on the reference date (use to align time series for display and analysis).
If NOT shifted, each trace will start on the reference date, but year will vary with the data.

Time series from which to create traces: 09019500.USGS.Streamflow.Day

Period to read: [] to []

Ensemble ID: Ensemble_1 Required identifier for ensemble.

Ensemble name: Test Ensemble Optional name for output.

Trace length: 1Year Default=1Year.

Reference date: 2008-01-01 Default=Jan 1 of first year.

Shift data how?: ShiftToReference 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

CreateEnsembleFromOneTimeSeries() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TSID	The time series identifier or alias for the time series used to create the ensemble.	None – must 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.
EnsembleName	The name for the new ensemble.	Blank.
TraceLength	An interval for the trace length (e.g., 1Year, #Month or, #Day).	1Year
ReferenceDate	The reference date indicates the starting date for each trace and should be left blank (resulting in a default of January 1 of the current year), or set to January 1 of a year of interest (use the format 01/01/YYYY or YYYY-MM-DD). Each trace can optionally be shifted (see ShiftDataHow).	January 1 of the first year in the source time series.
ShiftDataHow	Indicates whether the traces should be shifted. Possible values are: <ul style="list-style-type: none"> ShiftToReference – each trace will be shifted to the reference date, resulting in overlapping time series. NoShift – plotting the traces will result in a total line that matches the original time series, except that each trace can be manipulated individually. 	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)
```

Command Reference: CreateFromList()

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

Version 08.16.04, 2008-09-24

A `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 CreateFromList() Command

Create a list of time series from a list of location identifiers in a file.
 The information specified below is used with the identifiers to create time series identifiers, which are then used to read the time series. The identifiers are of the form:
 ID.DataSource.DataType.Interval.Scenario~InputType~InputName
 This command is useful for automating time series creation where lists of identifiers are being processed.
 The list file can contain comment lines starting with #.
 It is recommended that the path to the file be specified using a relative path.
 The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\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.

Command:
 CreateFromList(ListFile="Data\Diversions.txt",IDCol=1,DataSource="DWR",DataType="DivTotal",Interval="Month",InputType="HydroBase",IfNotFound=Default)

Remove Working Directory Cancel OK

CreateFromList

CreateFromList() Command Editor

The command syntax is as follows:

CreateFromList (Parameter=Value, ...)

Command Parameters

Parameter	Description	Default
ListFile	The name of the list file to read, surrounded by double quotes.	None – must be specified.
IDCol	The column (1+) in the list file containing the location identifiers to use in time series identifiers.	1
Delim	The delimiter characters that separate 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., "~ ~").	Comma
ID	Indicate a pattern to filter the identifiers in the list file. For example, use A* to only process identifiers in the list file that start	Process all identifiers.

Parameter	Description	Default
	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 available data types.	May or may not be required, depending on the input type. Refer to the input type appendices.
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: <ul style="list-style-type: none"> 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

Version 09.03.00, 2009-04-12

The `CreateRegressionTestCommandFile()` command is used for software testing (or certification of processes used in operations) and 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 that can be placed in `#` comments in command files to provide information for testing, for example:

```
#@expectedStatus Failure
```

Command # Comment Tags

Parameter	Description
@expectedStatus Failure @expectedStatus Warning	The <code>RunCommands()</code> command <code>ExpectedStatus</code> parameter is by default <code>Success</code> . However, a different status can be specified if it is expected that a command file will result in <code>Warning</code> or <code>Failure</code> 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 <code>Failure</code> and the test will pass. Another example is to test that the software properly treats a missing file as a failure.
@os Windows @os UNIX	The test is designed to work only on the specified platform and will be included in the test suite only if the <code>IncludeOS</code> 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).
@testSuite ABC	Indicate that the command file should be considered part of the specified test suite, as specified with the <code>IncludeTestSuite</code> 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 CreateRegressionTestCommandFile() command

This command creates a regression test command file, for use in software testing.
 Test command files should follow documented standards.
 A top-level folder is specified and will be searched for command files matching the specified pattern.
 The resulting output command file will include RunCommands() commands for each matched file, and can be independently loaded and run.
 A "setup" command file can also be included at the top of the output, for example to initialize database connections.
 It is recommended that file names be relative to the working directory, which is:
 C:\Develop\TSTool_SourceBuild\TSTool\test\regression\TestSuites\commands_general\create

Folder to search for TSTool command files:

Command file to create:

Setup command file:

Command file name pattern: Optional - file pattern to match (default is "Test_*.TSTool").

Append to output?: 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: 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,IncludeTestSuite="*",IncludeOS="Windows")
```

CreateRegressionTestCommandFile

CreateRegressionTestCommandFile() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
SearchFolder	The folder to search for regression test command files. All subfolders will also be searched.	None – must be specified.
OutputFile	The name of the command 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.
SetupCommandFile	The name of a TSTool command file that supplies setup commands, and which will be prepended to output. Use such a file to open database connections and set other global settings that apply to the entire test run.	Do not include setup commands.
FilenamePattern	Pattern for TSTool command files, using wildcards.	Test_*.TSTool
Append	Indicate whether to append to the output file (True) or overwrite (False). This allows multiple directory trees to be searched for tests, where the first command typically specifies False and additional commands specify True.	True
IncludeTestSuite	If *, all tests that match FilenamePattern and IncludeOS are included. If a test suite is specified, only include tests that have @testSuite tag values that match a value in IncludeTestSuite. One or more tags can be specified, separated by commas.	* – include all test cases.

IncludeOS	If *, all tests that match FilenamePattern and IncludeTestSuite are included. If an OS is 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.	* – include all test cases.
-----------	--	-----------------------------

See the RunCommands () 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:

```
# The test status below may be PASS or FAIL.
# A test can pass even if the commands file actual status is FAILURE, if failure is expected.
#      Test      Commands      Commands
#      Pass/     Expected     Actual
#      Num Fail   Status      Status      Command File
#-----
1  PASS  SUCCESS  SUCCESS  C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\add\Test_Add_1.TSTool
2  PASS  SUCCESS  SUCCESS  C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\add\Test_Add_Ensemble_1.TSTool
3  PASS  SUCCESS  SUCCESS  C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\addConstant\Test_AddConstant_1.TSTool
4  PASS  SUCCESS  SUCCESS
C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\adjustExtremes\Test_AdjustExtremes_1.TSTool
...
11 PASS  SUCCESS  SUCCESS
C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ChangeInterval\Test_ChangeInterval_IrregINST_To_3HourINST.TSTool
12 PASS  SUCCESS  SUCCESS  C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ChangePeriod\Test_ChangePeriod_1.TSTool
13 PASS  SUCCESS  SUCCESS
C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\compareTimeSeries\Test_AllDifferent.TSTool
14 PASS  SUCCESS  SUCCESS  C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\compareTimeSeries\Test_AllSame.TSTool
15 PASS  SUCCESS  SUCCESS
C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ComputeErrorTimeSeries\Test_ComputeErrorTimeSeries_1.TSTool
16 PASS  SUCCESS  SUCCESS
C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\convertDataUnits\Test_ConvertDataUnits_1.TSTool
17 PASS  SUCCESS  SUCCESS  C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\Copy\Test_Copy_1.TSTool
18 PASS  SUCCESS  SUCCESS  C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\CopyEnsemble\Test_CopyEnsemble_1.TSTool
19 PASS  SUCCESS  SUCCESS
C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\CreateEnsemble\Test_CreateEnsemble_1.TSTool
20 *FAIL* SUCCESS  WARNING
C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\CreateFromList\Test_CreateFromList_1.TSTool
21 PASS  Failure  FAILURE
C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\CreateTraces_Alias\Test_CreateTraces_Legacy_1.TSTool
22 PASS  SUCCESS  SUCCESS  C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\cumulate\Test_Cumulate_1.TSTool
```

This page is intentionally blank.

Command Reference: Cumulate()

Convert time series data values to cumulative values

Version 08.15.00, 2008-05-04

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. It is also useful to check the mass balance when routing time series (the cumulative values before and after routine will track closely). The following dialog is used to edit the command and illustrates the syntax of the command.

Cumulate() Command Editor

The command syntax is as follows:

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

Command Parameters

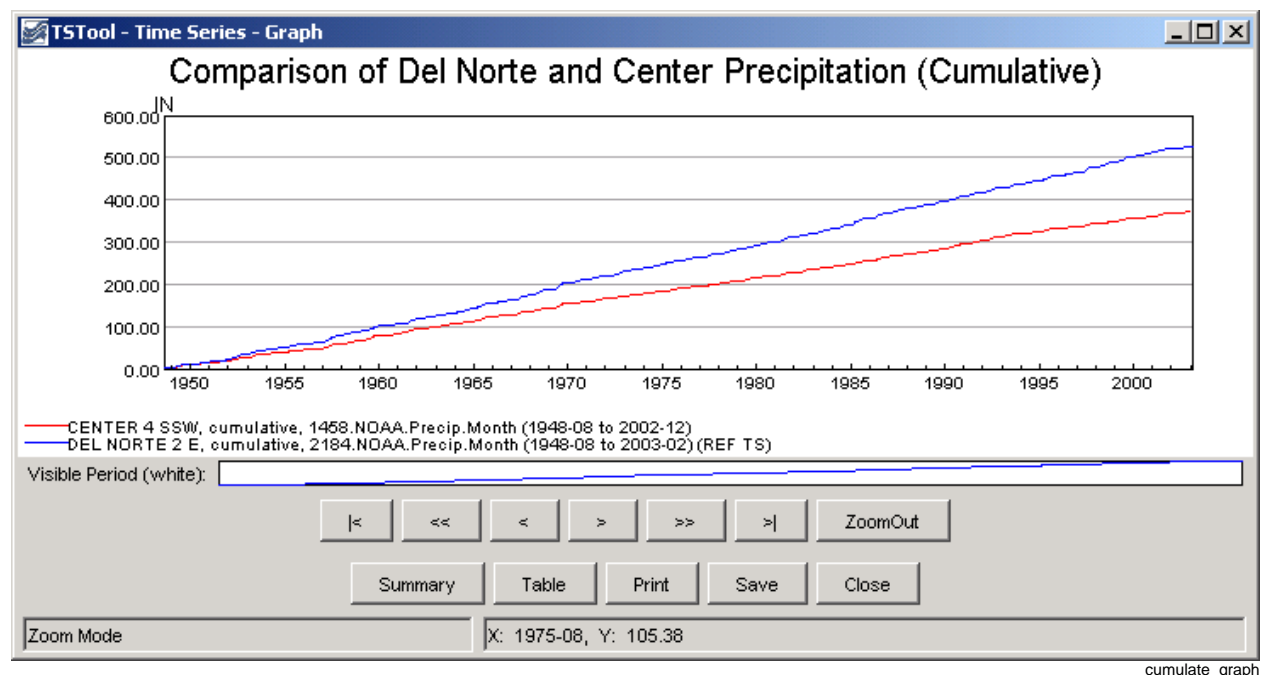
Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none"> AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will be modified. AllTS – all time series before the command. EnsembleID – all 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 <code>SelectTimeSeries()</code> 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.

Parameter	Description	Default
EnsembleID	The ensemble to be modified, if processing an ensemble.	TSID or EnsembleID must be specified.
Handle Missing How	Indicate how to handle missing data, one of: <ul style="list-style-type: none"> CarryForwardIfMissing – carry forward the last non-missing value SetMissingIfMissing – set the result to missing if the original value is missing. <p>The only difference in output is that the period of missing data will either be blank or a horizontal line in graphs.</p>	SetMissingIfMissing
Reset	A MM-DD date, day (1-31), or month (1-12) indicating when to reset the cumulative value to zero, before beginning to cumulate again. The features of this parameter are under development.	Do not reset.

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.



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 been 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.

Delta() Command Editor

The command syntax is as follows:

Delta (Parameter=Value, ...)

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none"> 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 expected to increase (or decrease) to a threshold and then reset, for example raw precipitation values that reset to zero when a container fills.	Data are not expected to reset.
ResetMax	The maximum expected data value, used when data are expected to increase (or decrease) to a threshold and then reset, for example raw precipitation values that reset to zero when a container fills.	Data are not expected to reset.
ExpectedTrend	Indicates trend of data, used when values can reset: <ul style="list-style-type: none"> Decreasing – values should decrease and then reset Increasing – values should increase and then reset 	Data are variable and don't reset at 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: <ul style="list-style-type: none"> R – indicates reset transition out of range > ResetMax r – indicates reset transition out of range < ResetMin V – indicates value out of range > ResetMax v – indicates value out of range < ResetMin 	Do not flag problem values.
Alias	Alias to assign to created time series. A literal string can be specified or use %-specifiers to set the alias dynamically (e.g., %L) to use the location part of the identifier.	None (but is highly recommended).

This page is intentionally blank.

Command Reference: DeselectTimeSeries()

Deselect time series

Version 09.04.03, 2009-08-24

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) pattern:

- The dot-delimited time series identifier parts are 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 location starting with A.
- Use `*,*.XXXXX,*,*` to match all time series with a data type XXXXX.

When specifying time series positions:

- The first time series created is position 1.
- Separate numbers by a comma. Specify a range, for example, as 1-3.

TS list: Optional - indicates the time series to process (default=AllTS).

TSID (for TSList=AllMatchingTSID):

EnsembleID (for TSList=EnsembleID):

Time series position(s) (for TSList=TSPosition): For example, 1,2,7-8 (positions are 1+).

Select all first?: ☒ Optional - eliminates need for separate select (default=False).

Command:

```
DeselectTimeSeries(TSList=AllMatchingTSID, TSID="40*", SelectAllFirst=True)
```

Cancel OK

DeselectTimeSeries

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: <ul style="list-style-type: none"> 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. TSPosition – time series specified by position in the results list (see TSPosition parameter below). 	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.
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:

```
TS 401234 = NewPatternTimeSeries(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: TS Alias = Disaggregate()

Create a new time series with shorter interval

Version 08.16.04, 2008-09-22

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 TS Alias = Disaggregate() Command

Disaggregation converts a longer-interval time series to a shorter interval.
Specify the new interval as Nbase, where base is Minute, Hour, Day (omitting N assumes a value of 1).
The new interval must be evenly divisible into the old interval.
The Ormsbee method is only enabled to disaggregate 1Day to 6Hour time series.
The SameValue method can disaggregate Year->Month, Month->Day, Day->NHour, Hour->NMinute.

Time series alias:

Original time series:

Method:

New interval: Required - e.g., Day, 6Hour.

New data type: Optional - default is same as original.

New data units: Optional - default is same as original.

Command:

```
TS HourTS =  
Disaggregate(TSID="DayTS",Method=Ormsbee,NewInterval=6Hour)
```

Disaggregate_Alias

Disaggregate() Command Editor

The command syntax is as follows:

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

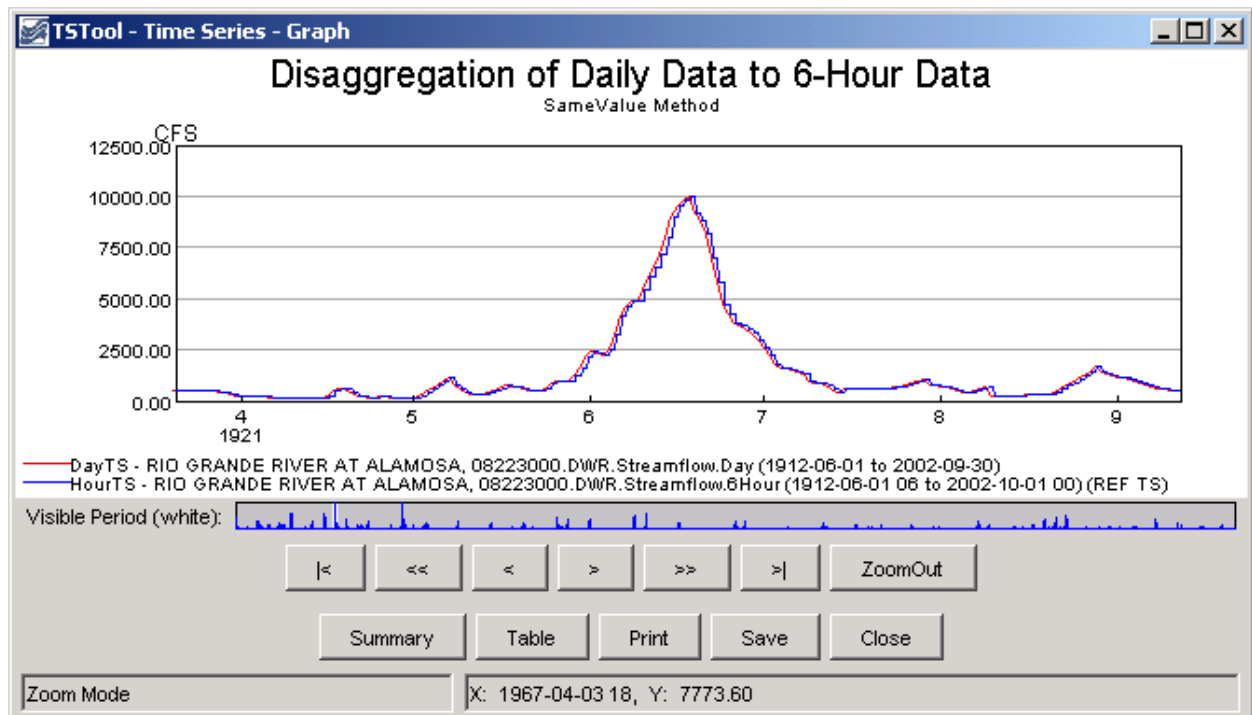
Command Parameters

Parameter	Description	Default
Alias	Alias for the disaggregated time series. The resulting time series will have the same header information and identifier as the original time series, with a different data interval and alias.	None – must be specified.
TSID	The time series identifier or alias for the time series to be disaggregated.	None – must be specified.
Method	<p>The method used to perform the disaggregation, one of the following:</p> <p>Ormsbee – 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.</p> <p>SameValue – this simple method causes the resulting time series to have the same value as the original. For example, a monthly time series that is disaggregated to a daily time series will result in each daily value being the same as for the corresponding value in the original monthly time series. Currently the following disaggregations are supported:</p> <ul style="list-style-type: none"> • Year to Month • Month to Day • Day to NHour (including 24Hour) • Hour to NMinute (including 60Minute) 	None – must be specified.
NewInterval	The data interval for the disaggregated time series (NHour, NDay, etc.).	None – must be specified.
NewDataType	The data type for the disaggregated time series, if different from the original.	Same data type as the original time series.
NewUnits	The units for the disaggregated time series, if different from the original.	Same units as the original time series.

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

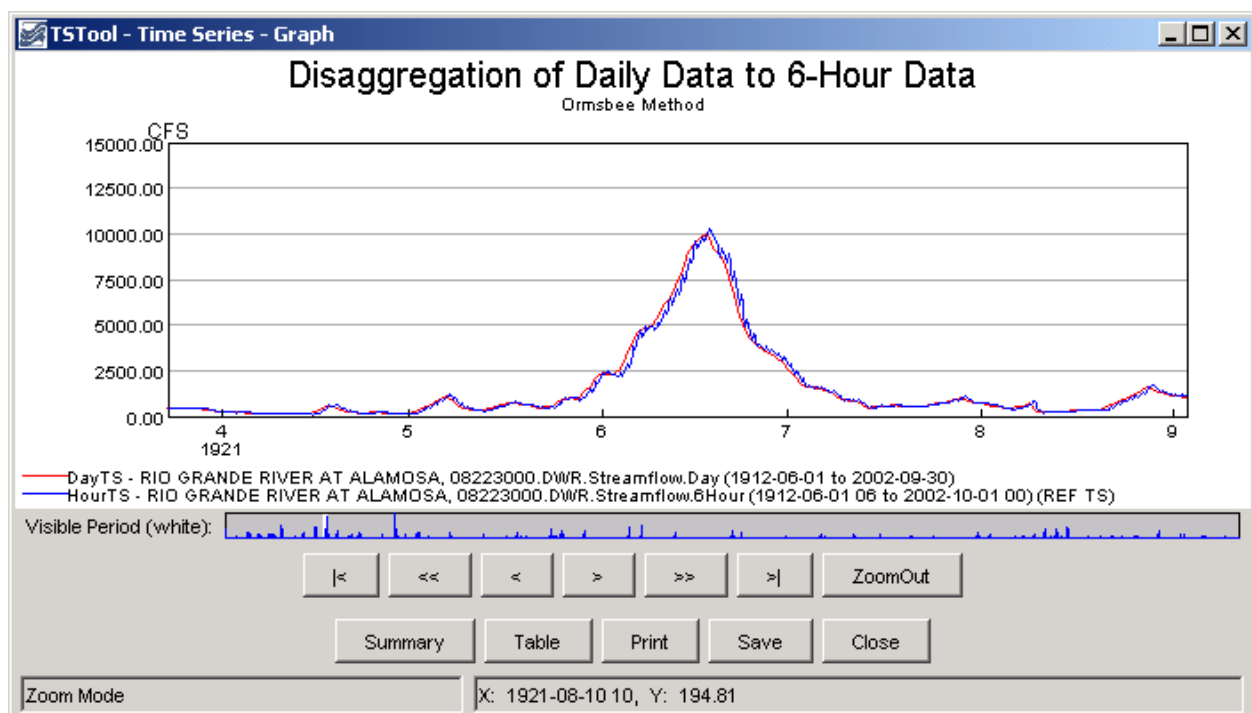
```
# 08223000 - RIO GRANDE RIVER AT ALAMOSA
TS DayTS = ReadTimeSeries("08223000.DWR.Streamflow.Day~HydroBase")
TS HourTS = Disaggregate(TSID="DayTS",Method=Ormsbee,NewInterval=6Hour)
```

Examples of graphs for the original and disaggregated data are shown below, for the two disaggregation methods:



disaggregate_SameValue_Graph

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



disaggregate_SameValue_Graph

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

This page is intentionally blank.

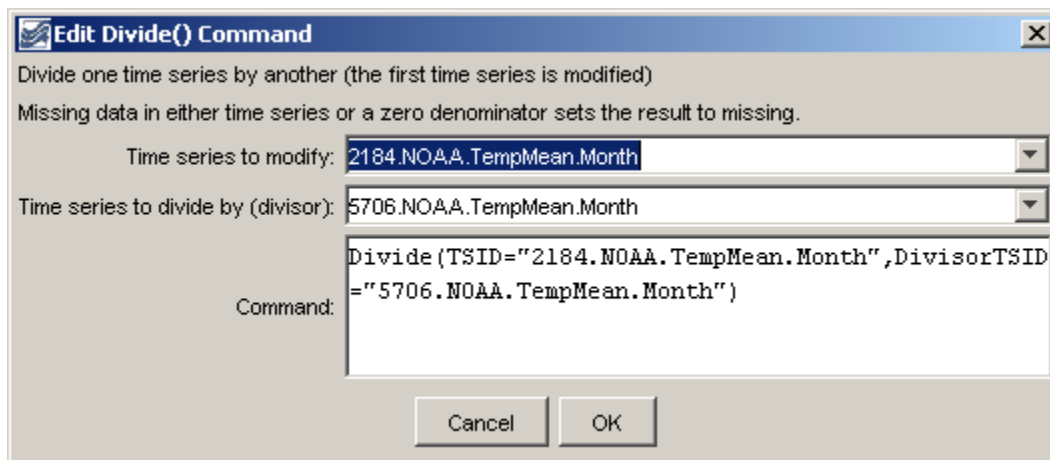
Command Reference: Divide()

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

Version 08.16.04, 2008-09-24

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.



Divide

Divide() Command Editor

The command syntax is as follows:

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

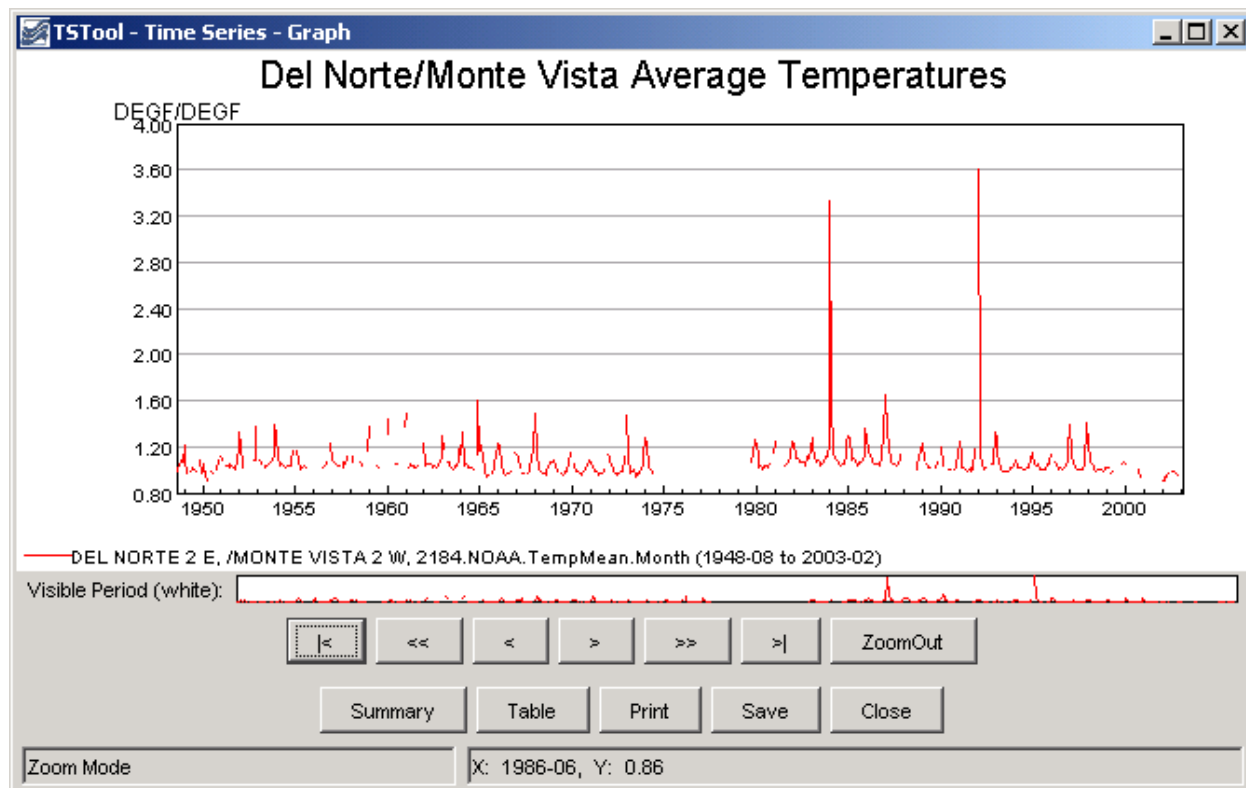
Command Parameters

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



divide_graph

Results from Divide() Command

Command Reference: Exit()

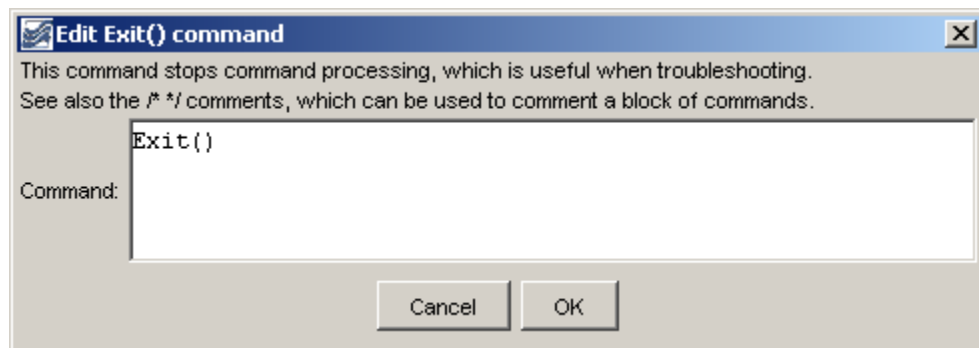
Stop processing commands

Version 08.16.04, 2008-09-25

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 be 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.

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



Exit

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 the fully-expanded file

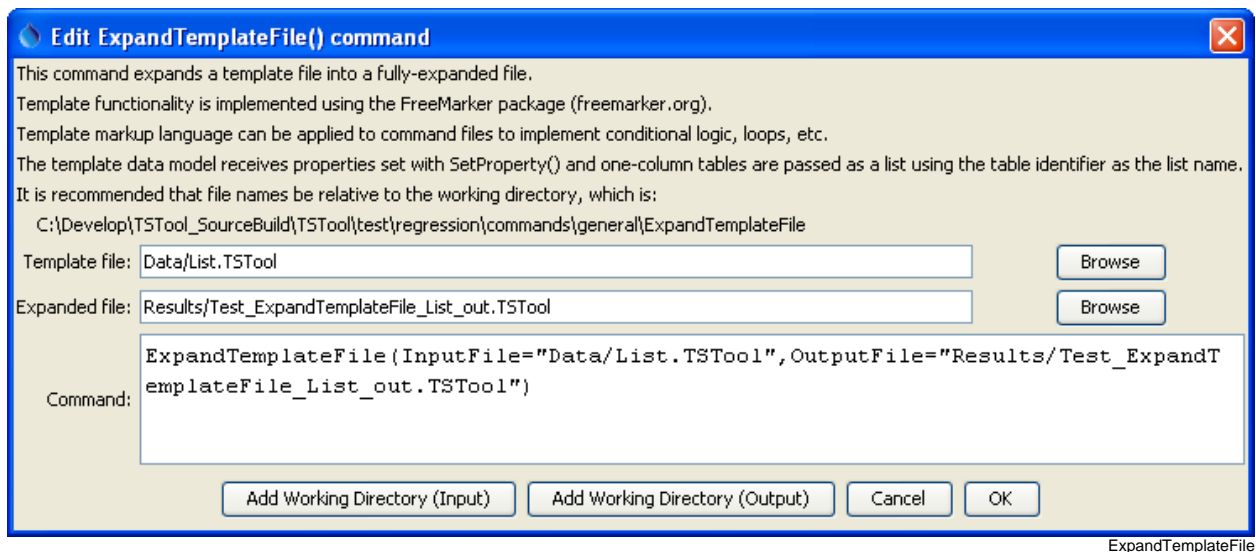
Version 09.09.00, 2010-09-23

The `ExpandTemplateFile()` command processes a template file (typically a command file or time series product file but can be any text file) to create a fully-expanded file. Templates facilitate utilizing conditional logic, loops, and other dynamic processing functionality. For example, a template can be used to repeat commands for multiple location identifiers. Templates can also be applied to other text files.

The FreeMarker software (<http://freemarker.org>) is used to implement templates. Refer to the online documentation for information about the markup language used to create templates. The built-in `normalizeNewlines` user directive is automatically used to ensure that expanded files use newline characters appropriate for the operating system – this leads to extra first and last lines in the template during processing.

Properties set with the `SetProperty()` command are passed to the template tool. One-column tables are also passed as lists, using the table identifier as the property name. For example, use the `CopyTable()` command to create a one-column table that can be used as a list for template expansion.

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



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.	None – must be specified.
OutputFile	The name of the expanded output file.	None – must be specified.

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

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

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

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

```
# 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 setEnd = "2000-03-15">
<#assign units = "CFS">
<#assign locList = ["loc1", "loc2", "loc3", "loc4"]>
<#list locList as loc>
TS ${loc} =
NewPatternTimeSeries(NewTSID="${loc}..Streamflow.Day",SetStart="${setStart}",
    SetEnd="${setEnd}",Units="${units}",PatternValues="${loc_index + 1},0")
</#list>
```

```
# 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
TS loc1 = NewPatternTimeSeries(NewTSID="loc1..Streamflow.Day",
    SetStart="2000-01-01",SetEnd="2000-03-15",
    Units="CFS",PatternValues="1,0")
TS loc2 = NewPatternTimeSeries(NewTSID="loc2..Streamflow.Day",
    SetStart="2000-01-01",SetEnd="2000-03-15",
    Units="CFS",PatternValues="2,0")
TS loc3 = NewPatternTimeSeries(NewTSID="loc3..Streamflow.Day",
    SetStart="2000-01-01",SetEnd="2000-03-15",
    Units="CFS",PatternValues="3,0")
TS loc4 = NewPatternTimeSeries(NewTSID="loc4..Streamflow.Day",
    SetStart="2000-01-01",SetEnd="2000-03-15",
    Units="CFS",PatternValues="4,0")
```

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() Command

Fill time series data missing values with a constant value.
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: Optional - indicates the time series to process (default=AllTS).

TSID (for TSList=AllMatchingTSID):

EnsembleID (for TSList=EnsembleID):

Constant value: Required - constant value to fill missing data.

Fill start date/time: Optional - fill start (default=fill all).

Fill end date/time: Optional - fill end (default=fill all).

Fill flag: Optional - string to flag filled values.

Command:
`FillConstant (TSList=AllMatchingTSID, TSID="08236500.DWR.Streamflow.Month", ConstantValue=500, FillStart="1970-02", FillEnd="1970-10", FillFlag="C")`

FillConstant

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 of: <ul style="list-style-type: none"> 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.
ConstantValue	Constant value to use when filling missing data.	None – must be specified.
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.	No flag is assigned.

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

Version 08.16.04, 2008-09-19

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

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

where:

i = day

D1 is the daily data at location 1

M1 is the monthly data at location 1 (for the month corresponding to the day)

D2 is the daily data at location 2

M2 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 FillDayTSFrom2MonthTSAnd1DayTS() Command

Fill a daily time series using the relationship: $D1[i] = D2[i] * M1[i] / M2[i]$.
 The monthly values M1/M2 give an average estimate of the volume ratio and D2 provides an estimate of the daily pattern in the month.
Note that TSTool cannot verify whether 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.DWR.Streamflow.Month

Daily time series for distribution (D2): 08236000.DWR.Streamflow.Day

Fill start date/time: Optional - default=fill all.

Fill end date/time: Optional - default=fill all.

Command: FillDayTSFrom2MonthTSAnd1DayTS(TSID_D1="08235350.USGS.Streamflow.Day",TSID_M1="08235350.USGS.Streamflow.Month",TSID_M2="08236000.DWR.Streamflow.Month")

Cancel OK

FillDayTSFrom2MonthTSAnd1DayTS

FillDayTSFrom2MonthTSAnd1DayTS() Command Editor

The command syntax is as follows:

FillDayTSFrom2MonthTSAnd1DayTS (Parameter=Value,...)

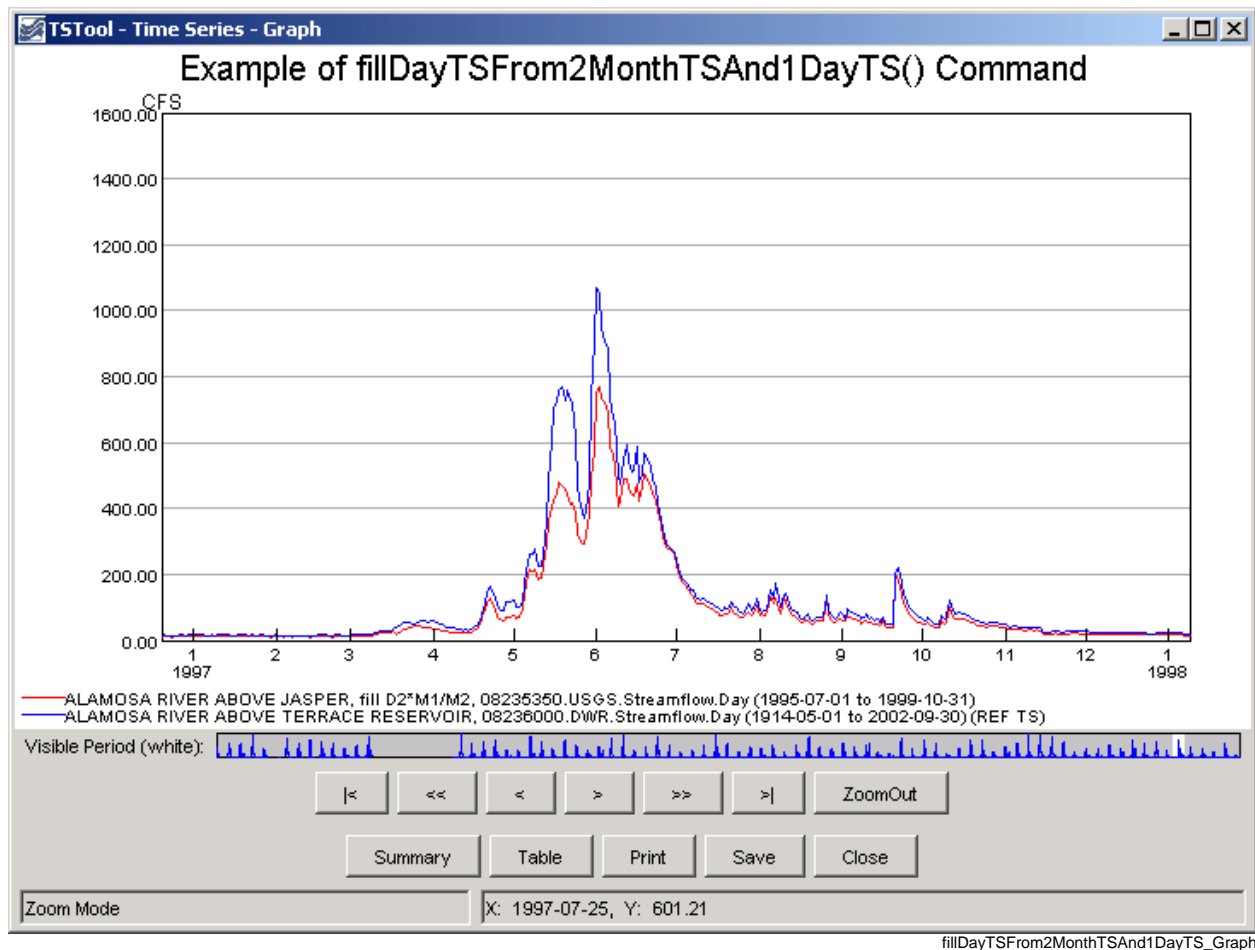
Command Parameters

Parameter	Description	Default
TSID_D1	The time series identifier or alias for the daily time series to be filled.	None – must be specified.
TSID_M1	The time series identifier or alias for the monthly time series, corresponding to TSID_D1, to supply the monthly values to be distributed to daily.	None – must be specified.
TSID_M2	The time series identifier or alias for the independent monthly time series.	None – must be specified.
TSID_D2	The time series identifier or alias for the independent daily time series, corresponding to TSID_M2.	None – must be specified.
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.

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 09.08.01, 2010-09-14

The `FillFromTS()` command fills missing data in a time series (or ensemble) by transferring non-missing 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.

Edit FillFromTS() Command

Copy data values from the independent time series to replace missing 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 dependent time series. If multiple independent time series are specified (e.g., for ensembles), the same number of dependent time series must be specified. Use a `SetOutputPeriod()` command if the dependent time series period will be extended. Specify dates with precision appropriate for the data, use blank for all available data, `OutputStart`, or `OutputEnd`.

Dependent TS List: Optional - indicates the time series to process (default=AllTS).

TSID (for TSList=*MatchingTSID):

EnsembleID (for TSList=EnsembleID):

Independent TS List: Optional - indicates the time series to process (default=AllTS).

Independent TSID (for Independent TSList=*MatchingTSID):

Independent EnsembleID (for Independent 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).

Recalculate limits: Optional - recalculate original data limits after fill (default=False).

Command: `FillFromTS (TSList=AllMatchingTSID, TSID="08241000.DWR.Streamflow.Month", IndependentTSList=AllMatchingTSID, IndependentTSID="08240500.DWR.Streamflow.Month")`

FillFromTS

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

Parameter	Description	Default
	<ul style="list-style-type: none"> 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 <code>SelectTimeSeries()</code> 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 a <code>TSList=*TSID</code>
EnsembleID	The ensemble to be modified, if processing an ensemble.	Required when <code>TSList=EnsembleID.</code>
Independent TSList	Indicates how to determine the list of independent time series (see the explanation of TSList).	AllTS
Independent TSID	The time series identifier or alias for the independent time series (see the explanation of TSID).	Required when a <code>IndependentTSList=*TSID</code>
Independent EnsembleID	The ensemble identifier for the independent time series (see the explanation of EnsembleID).	Required when <code>IndependentTSList=EnsembleID.</code>
FillStart	The date/time to start filling, if other than the full time series period.	Fill the entire period.
FillEnd	The date/time to end filling, if other than the full time series period.	Fill the entire period.
RecalcLimits	Available only for monthly time series. Indicate whether the original data limits for the time series should be recalculated after the filling the time series. Setting to True is appropriate if the independent time series provides observations consistent with the original data.	False (only the values in the initial time series will be used for historical data).

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.

Edit FillHistMonthAverage() Command

Fill monthly time series with historical monthly averages.
Historical averages are computed immediately after reading the data and therefore do not include filled values.
Only monthly time series can be processed.

TS list: Optional - indicates the time series to process (default=AllTS).

TSID (for TSList=AllMatchingTSID):

Fill start: Optional - default is full period.

Fill end: Optional - default is full period.

Fill flag: Optional - string to indicate filled values (use "auto" for MonAvg).

Fill flag description: Optional - description for fill flag.

Command:

```
FillHistMonthAverage (TSList=AllMatchingTSID, TSID="1179.NOAA.Precip.Month", FillFlag="H")
```

FillHistMonthAverage

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: <ul style="list-style-type: none">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.	None – must be specified.

Parameter	Description	Default
	<ul style="list-style-type: none"> LastMatchingTSID – process the last time series that has an identifier matching the TSID parameter. SelectedTS – process the time series that are selected (see <code>SelectTimeSeries()</code>). 	
TSID	Used if <code>TSList=AllMatchingTSID</code> 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 <code>TSList=AllMatchingTSID</code> .
FillStart	Date/time indicating the start of filling, using a precision appropriate for the time series, or <code>OutputStart</code> .	Fill the entire time series.
FillEnd	Date/time indicating the end of filling, using a precision appropriate for the time series, or <code>OutputEnd</code> .	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 <code>Auto</code> to use a flag with the month abbreviation + <code>Avg</code> .	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 time series follow...											
Time series: 0125.NOAA.Precip.Month (IN)											
Monthly limits for period 1948-08 to 1949-12 are:											
Month	Min	MinDate	Max	MaxDate	Sum	# Miss.	% Miss.	# Not Miss.	% Not Miss.	Mean	
Jan	0.2	1949-01	0.2	1949-01	0.2	0	0.00	1	100.00	0.2	
Feb	0.1	1949-02	0.1	1949-02	0.1	0	0.00	1	100.00	0.1	
Mar	0.1	1949-03	0.1	1949-03	0.1	0	0.00	1	100.00	0.1	
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	0.7	
Jul	1.5	1949-07	1.5	1949-07	1.5	0	0.00	1	100.00	1.5	
Aug	0.7	1949-08	0.8	1948-08	1.5	0	0.00	2	100.00	0.8	
Sep	0.1	1948-09	1.1	1949-09	1.2	0	0.00	2	100.00	0.6	
Oct	0.1	1949-10	0.5	1948-10	0.7	0	0.00	2	100.00	0.3	
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	0.5	

Command Reference: FillHistYearAverage()

Fill missing time series data using historical yearly average data

Version 08.15.00, 2008-05-04

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: How to get the time series to fill.

Identifier (TSID) to match:

Fill period: to

Fill flag: 1-character flag to indicate fill.

Command:

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

FillHistYearAverage

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: <ul style="list-style-type: none">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	None – must be specified.

Parameter	Description	Default
	<code>selectTimeSeries()</code> .	
TSID	Used if <code>TSList=AllMatchingTSID</code> 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 <code>TSList=AllMatchingTSID</code> .
FillStart	Date/time indicating the start of filling, using a precision appropriate for the time series, or <code>OutputStart</code> .	Fill the entire time series.
FillEnd	Date/time indicating the end of filling, using a precision appropriate for the time series, or <code>OutputEnd</code> .	Fill the entire time series.
FillFlag	If specified as a single character, data 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.	No flag is assigned.

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:          95.0000 ACRE on 1954
Max:          2684.0000 ACRE on 1959
Sum:          11090.0000 ACRE
Mean:          1008.1818 ACRE
Number Missing:  42 (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: Optional - indicates the time series to process (default=AllTS).

TSID (for TSList=AllMatchingTSID):

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: Optional (default=0 to fill all).

Transformation for interpolation: Optional - currently only default value (None) is recognized.

Fill flag: Optional - string to flag filled values (default=no flag).

Command:

```
FillInterpolate (TSList=AllMatchingTSID, TSID="06707500.DWR.Streamflow.Month", MaxIntervals=3, Transformation=None)
```

FillInterpolate

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: <ul style="list-style-type: none">AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will be modified.AllTS – all time series before the	AllTS

Parameter	Description	Default
	command. <ul style="list-style-type: none"> 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 <code>SelectTimeSeries()</code> 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 <code>TSList=*TSID</code>
EnsembleID	The ensemble to be modified, if processing an ensemble.	Required when <code>TSList=EnsembleID</code> .
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

Version 09.06.03, 2009-09-01

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. Typically, the time series involved in the analysis are related, such as being from nearby locations in a region.

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 ($N1$) can be specified to indicate a valid relationship.
- A minimum correlation coefficient r can be specified to indicate a valid relationship.
- The best fit indicator can be the correlation coefficient (R) or the standard error of prediction (SEP, described below).

The interactive MSA tool is available to facilitate configuration of the `FillMixedStation()` command. The primary uses of the MSA tool and `FillMixedStation()` command are:

- Use the MSA tool to evaluate filling options before finalizing the fill commands. For example, run the MSA tool and review the output report to confirm the filling methodology and corresponding command parameters. A `FillMixedStation()` command can then be created and passed to the TSTool command list.
- If no evaluation is needed, use the or `FillMixedStation()` command editor to create a single `FillMixedStation()` analysis command, which when run evaluates the best filling parameters for filling and performs the filling. A single command can be used to fill many time series.

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. Performance can be increased by using the Mixed Station Analysis tool to determine time series that result in the best fit, and excluding all other time series in the fill command. The number of combinations can also be limited by reducing the number of parameter options and using stricter limitations on the number of points and correlation coefficient 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 ranked according to the best fit indicator (e.g., R or standard error of prediction SEP, as described below). If a monthly analysis is performed, the results for each month are ranked.
4. Missing data in the dependent time series are filled using the regression results. 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.

Best Fit Indicators

Best fit indicators that are available include:

1. Correlation coefficient, R , defined as:

$$R = \frac{N \sum XY - \sum X \sum Y}{\sqrt{[N \sum X^2 - (\sum X)^2] \cdot [N \sum Y^2 - (\sum Y)^2]}}$$

where X is the independent and Y is the dependent for all overlapping data points.

2. Standard Error of Prediction (SEP), defined as:

$$SEP = \sqrt{\frac{\sum (Y - Y')^2}{N}}$$

where Y is the observed value and Y' and is the estimated value using the relationship determined for filling, for all original points in the independent time series.

In the future, additional indicators may be added, such as the Nash-Sutcliffe efficiency coefficient.

Mixed Station Analysis Tool

The Mixed Station Analysis tool is started after time series results are generated in TSTool, for example as the result of reading a list of time series. The following dialog illustrates the parameters of the tool, which are essentially the same as for the `FillMixedStation()` command that is described in the next section.

Mixed Station Analysis

This tool finds the best fit to fill the dependent time series with data from the independent time series, and generates commands to perform the filling.
The dependent and independent time series can be selected using the TS list parameters:
The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\FillMixedStation

— Edit Mixed Station Analysis Parameters —

Dependent TS list: Optional - indicates the time series to process (default=AllTS).

Dependent TSID (for TSList=AllMatchingTSID):

Independent TS list: Optional - time series used to fill (default=AllTS).

Independent TSID (for TSList=AllMatchingTSID):

Best Fit: Required - best fit indicator, for ranking output.

Analysis method(s): ☐ MOVE2 ☐ OLSRegression Optional - methods to use in analysis (default=OLSRegression).

Number of equations: Optional - number of equations to use in the analysis (default=OneEquation).

Transformation(s): ☐ Log ☐ None Optional - transformations to use in analysis (default=None).

Intercept: Optional - 0.0 is allowed with Transformation=None (default=no fixed intercept).

Analysis period: to

Fill period: to

Minimum data count: Optional - minimum number of overlapping points required for analysis (default=10).

Minimum R: Optional - minimum correlation coefficient R required for a best fit (default = 0.5).

Output file:

— Perform Mixed Station Analysis and Review Results —

— Copy Command(s) to TSTool —

Fill command(s):

Ready

tool_MixedStationAnalysis

Mixed Station Analysis Tool after Initial Display

The MSA tool is used as follows:

1. The tool will initially display default parameter values, but these values can be changed using the interface. Select parameters as appropriate for the analysis. The corresponding `FillMixedStation()` command is shown at the bottom of the window.
2. Press the **Analyze** button to perform the analysis. The analysis may run for several minutes.
3. When the analysis is complete, press the **View Output File** button to view the analysis results.
4. If satisfied with the results, meaning that reasonable relationships have been determined, go to the following step. Otherwise adjust the analysis parameters and run the analysis again.
5. Use the **Copy Command to TSTool** button to copy the `FillMixedStation()` command to TSTool. The command will be inserted as if it were edited from the **Commands** menu.

6. The MSA tool can then be closed and commands run as usual to automate data processing.

Subsequent edits of the command can occur using the normal command editor. The following section provides an example of the standard command editor and a description of all of the parameters.

Command Editing

The following dialog is used to edit the `FillMixedStation()` command and illustrates the syntax of the command:

Edit FillMixedStation() Command

This command finds the best fit to fill the dependent time series with data from the dependent time series, and performs the filling.
 The dependent and independent time series can be selected using the TS list parameters:
 The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\FillMixedStation

Dependent TS list: Optional - indicates the time series to process (default=AllTS).

Dependent TSID (for TSList=AllMatchingTSID):

Independent TS list: Optional - time series used to fill (default=AllTS).

Independent TSID (for TSList=AllMatchingTSID):

Best Fit: Required - best fit indicator, for ranking output.

Analysis method(s): ☒ MOVE2 ☒ OLSRegression Optional - methods 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 - transformations to use in analysis (default=None).

Intercept: Optional - 0.0 is allowed with Transformation=None (default=no fixed intercept).

Analysis period: to

Fill period: to

Minimum data count: 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).

Output file: Results/Test_FillMixedStation_gunnv.xbg_MonthlyEquations_out.txt

Fill command(s):

```
FillMixedStation(BestFitIndicator=SEP,AnalysisMethod="MOVE2,OLSRegression",NumberOfEquations=MonthlyEquations,Transformation="Log,None",MinimumR=.7,OutputFile="Results/Test_FillMixedStation_gunnv.xbg_MonthlyEquations_out.txt")
```

FillMixedStation

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: <ul style="list-style-type: none"> • 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. 	None – must be specified.
DependentTSID	The time series identifier or alias for the dependent time series to be processed, using the * wildcard character to match multiple time series.	Required if DependentTSList=*TSID.
IndependentTSList	Indicates the list of independent time series to be considered for each dependent time series, one of: <ul style="list-style-type: none"> • 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 	None – must be specified.

Parameter	Description	Default
	with the <code>SelectTimeSeries()</code> command will be processed.	
<code>IndependentTSID</code>	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 <code>IndependentTSList=</code> *TSID.
<code>BestFitIndicator</code>	Specifies the indicator to use when determining the best fit, one of: <ul style="list-style-type: none"> ▪ <code>R</code> (correlation coefficient). ▪ <code>SEP</code> (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. ▪ <code>SEPTotal</code>, when used with one equation, it is the same as <code>SEP</code>. When used with monthly equations, it is the average error considering all months. 	<code>SEP</code> .
<code>AnalysisMethod</code>	Specify the method(s) to analyze the data, in order to determine the best fit, including <code>OLSRegression</code> and/or <code>MOVE2</code> . If multiple methods are specified, separate with commas and surround with double quotes.	<code>OLSRegression</code>
<code>NumberOfEquations</code>	The number of equations to use for the analysis: <code>OneEquation</code> or <code>MonthlyEquations</code> . Only one may be chosen. If necessary, use more than one command to use different parameter combinations for different groups of time series.	None – must be specified.
<code>Transformation</code>	Indicates how to transform the data before analyzing. Specify as <code>None</code> (no transformation) or <code>Log</code> (for Log_{10}). If the <code>Log</code> option is used, zero and negative values in data are set to <code>.001</code> . Missing data are ignored. If multiple values are selected, separate with a comma and surround with double quotes.	None (no transformation)
<code>Intercept</code>	Specify as <code>0</code> to force the intercept of the best-fit line through the origin. This is made available only for <code>OLS</code> regression analysis on untransformed data, to be consistent with the <code>FillRegression()</code> command.	Do not force the intercept through zero.
<code>AnalysisStart</code>	The date/time to start the analysis, to focus on a period appropriate for analysis. For example, specify the unregulated period for streamflow.	If blank, analyze the full period.
<code>AnalysisEnd</code>	The date/time to end the analysis.	If blank, analyze the full period.

Parameter	Description	Default
FillStart	The date/time to start filling, if other than the full time series period.	If blank, fill the full period.
FillEnd	The date/time to end filling, if other than the full time series period.	If blank, fill the full period.
MinimumDataCount	The minimum number of overlapping data 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.	10
MinimumR	The minimum correlation coefficient required for a best fit. If the minimum is not met, then the results are not considered in the best fit ranking or filling.	0.5
OutputFile	Output file for the results, either as a file name to be written to the working directory, or a full path.	If not specified, partial results of the analysis may 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)
```

This page is intentionally blank.

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.

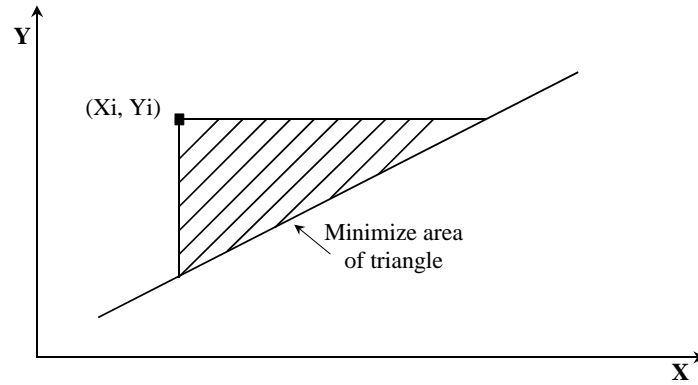
The fillMOVE1 () command is more sophisticated than the fillRegression () command.

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:

$$Y_i = \bar{Y}_1 + \frac{S_{y1}}{S_{x1}} \left[X_i - \bar{X}_1 \right]$$

or

$$Y_i = a + bX_i$$

where

N_1 = concurrent or overlapping period of record

\bar{X}_1 = mean for independent variable for N_1 years

\bar{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

$$b = \frac{S_{y1}}{S_{x1}}$$

$$a = \bar{Y}_1 - b\bar{X}_1$$

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

Version 08.15.00, 2008-05-04

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:

$$Y_i = \bar{Y} + \frac{S_y}{S_x} \left[X_i - \bar{X} \right]$$

where

Y_i = discharge for dependent time series

X_i = discharge for independent time series

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

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

$$\bar{Y} = \bar{Y}_1 + \frac{N_2}{N_1 + N_2} \left[b(\bar{X}_2 - \bar{X}_1) \right] \quad \text{(Equation 7-5a for Two-Station Comparison in Appendix 7 of Bulletin 17B)}$$

$$S_y^2 = \frac{1}{(N_1 + N_2 - 1)} \left[(N_1 - 1)S_{y1}^2 + (N_2 - 1)b^2S_{x2}^2 + \frac{N_2(N_1 - 4)(N_1 - 1)}{(N_1 - 3)(N_1 - 2)} (1 - r^2)S_{y1}^2 + \frac{N_1N_2}{N_1 + N_2} b^2(\bar{X}_2 - \bar{X}_1)^2 \right]$$

(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

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

\bar{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 for a description of the MOVE2 procedure.
 The analysis period(s) will be used to determine the relationships used for filling.
 Use a setOutputPeriod() command if the dependent time series period will be extended.
 Specify dates with precision appropriate for the data, use * for all available data, OutputStart, or OutputEnd.

Time series to fill (dependent): 06758500.DWR.Streamflow.Month

Independent time series: 06754000.DWR.Streamflow.Month

Number of equations: MonthlyEquations Number of equations to use (blank=one equation).

Transformation: Transformation box 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: 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")

Cancel OK

FillMOVE2

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 time series to be filled (dependent time series).	None – must be specified.
IndependentTSID	The time series identifier or alias for the independent time series, to supply data.	None – must be specified.
NumberOfEquations	OneEquation or MonthlyEquations, indicating how many relationships are to be determined.	OneEquation
Transformation	Log or None, indicating the type of data 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.	None
Dependent Analysis Start/End	The period for N_1 (overlapping data) that is used to analyze the dependent time series. For example, this may be the unregulated period for streamflow data. Typically, this is longer than the independent analysis period.	Analyze the full period.
Independent Analysis Start/End	The period for N_2 (non-overlapping data) that is used to analyze the independent time series. For example, this may be the unregulated period for streamflow data.	Analyze the full period.
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 filled points on graphs and other output.	Do not flag filled data.

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.

```
# 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.
# -e-b-----eb-----eb-----eb-----eb-----eb-----eb-----eb-----eb-----eb-----e
10/1908 - 9/1996 ACFT WYR
1909 09034500 AVG AVG WET WET AVG AVG AVG WET WET WET WET
1910 09034500 WET WET WET WET WET WET AVG AVG AVG WET WET WET
1911 09034500 AVG AVG WET AVG AVG AVG AVG WET WET WET WET
1912 09034500 WET WET WET WET WET AVG AVG WET WET WET WET
...omitted...
```

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

FillPattern() Command Editor

FillPattern

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: <ul style="list-style-type: none"> 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: FillPrincipalComponentAnalysis()

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 end-points 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

This command fills missing data in time series by prorating values from an independent time series.

The proration factor is calculated in one of the following ways (indicated by "FactorMethod"):

- 1) Recompute the factor at each point where a non-missing value is found in both time series (NearestPoint).
The initial value in the filled time series is used to compute the initial factor, for missing data at the end of the fill period.
- 2) Recompute the factor as the dependent time series value divided by the average of independent values (AnalysisAverage).
The factor (ratio) is $TS/IndependentTS$ and the filled value is calculated as $factor * IndependentTS$.

If the independent time series data value is zero, the factor remains as previously calculated to avoid division by zero.

The independent time series is not itself filled if matched for the TSLIST.

The fill start and end, if specified, will limit the period that is filled.

Use standard date/time formats appropriate for the date/time precision of the time series.

TS List: **AllMatchingTSID** Indicates the time series to process (default=AllTS).

TSID (for TSLIST=AllMatchingTSID): 06754000.DWR.Streamflow.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",FillDirection=Forward,InitialValue=0)`

Cancel OK

FillProrate

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 processed, one of: <ul style="list-style-type: none"> 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. Use 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.
IndependentTSID	The time series identifier or alias for the independent time series.	None – must be specified.
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 that are filled.	None – do not flag filled data.
FillDirection	Specify the direction of the fill as Forward or Backward.	Forward
FactorMethod	Specify how to calculate the factor to use in proration, one of: <ul style="list-style-type: none"> AnalyzeAverage – calculate the factor of the average of the time series divided by the independent time series, using the analysis period. NearestPoint – calculate the factor at the nearest point where both 	NearestPoint

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: <ul style="list-style-type: none"> • 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
```

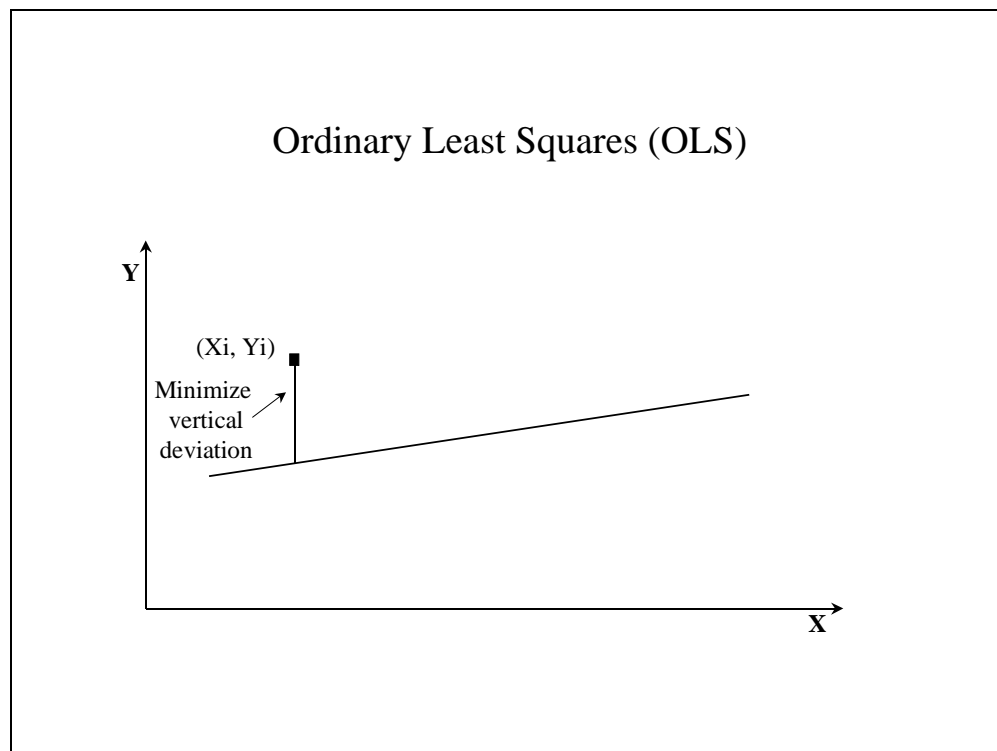
Command Reference: FillRegression()

Fill missing time series data using ordinary least squares regression

Version 08.15.00, 2008-05-04

The `FillRegression()` command fills missing data in a time series using ordinary least squares (OLS) regression (see also the `FillMOVE2()` command). Regression can be applied only to regular interval time series. The first time series selected (dependent time series) will be filled using the other selected time series (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. The **Results...Graph - XY-Scatter** is a useful tool for reviewing data. Regression relationships are developed using the analysis period for the time series and are applied to the fill period. Refer to the log file and time series properties for analysis details.

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.



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

$$Y_i = \bar{Y}_1 + r \frac{S_{y1}}{S_{x1}} \left[X_i - \bar{X}_1 \right]$$

or

$$Y_i = a + bX_i$$

where

N_1 = concurrent or overlapping period of record

\bar{X}_1 = mean for independent variable for N_1 years

\bar{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

$b = r \frac{S_{y1}}{S_{x1}}$, r = correlation coefficient

$a = \bar{Y}_1 - b\bar{X}_1$

Note that the correlation coefficient, r , is used to compute the slope, b , of the line.

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

Edit FillRegression() command

Fill missing data using ordinary least squares (OLS) regression.
 The analysis period will be used to determine relationships used for filling.
 Use a setOutputPeriod() command before reading to extend the dependent time series, if necessary.
 Specify dates with precision appropriate for the data, use blank for all available data, OutputStart, or OutputEnd.

Time series to fill (dependent): 06753400.USGS.Streamflow.Month

Independent time series: 06753500.USGS.Streamflow.Month

Number of equations: Number of equations to use (blank=one equation).

Analysis month: Can be used with monthly equations (blank=all months).

Transformation: How to transform data before analysis (blank=None).

Intercept: Blank or 0.0 are allowed with no transformation.

Analysis period: to

Fill Period: to

Fill flag: 1-character flag to indicate fill.

Command:

```
FillRegression(TSID="06753400.USGS.Streamflow.Month",IndependentTSID="06753500.USGS.Streamflow.Month")
```

Cancel OK

FillRegression

FillRegression() Command Editor

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 to be filled.	None – must be specified.
Independent TSID	The time series identifier or alias for the independent time series. Right-clicking on a time series allows it to be converted to a temporary time series (TEMPTS).	None – must be specified.
NumberOfEquations	The number of equations to use for the analysis: OneEquation or MonthlyEquations.	OneEquation
AnalysisMonth	Indicate the month to process when using monthly equations. Currently only a single month can be specified.	Process all months.
Transformation	Indicates how to transform the data before analyzing. Specify as None (previously Linear) or Log (for Log ₁₀). If the Log option is used, zero and negative values are set to .001 (missing data values are ignored in the analysis).	None (no transformation).
Intercept	Specify as 0 to force the intercept of the best-fit line through the origin (not available for log transformation).	Parameter is optional and if specified the default is to not force the intercept through zero.
AnalysisStart	The date/time to start the analysis – use to focus on only a period appropriate from analysis. For example specify the unregulated period for streamflow.	Analyze the full period.
AnalysisEnd	The date/time to end the analysis – use to focus on only a period appropriate from analysis.	Analyze the full period.
FillStart	The date/time to start filling, if other than the full time series period.	Fill the full period.
FillEnd	The date/time to end filling, if other than the full time series period.	Fill the full period.
FillFlag	A single character that will be used to flag filled data.	Filled values will not be flagged.

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

```
# 06753400 - LONETREE CREEK AT CARR, CO.
06753400.USGS.Streamflow.Month~HydroBase
# 06753500 - LONETREE CREEK NEAR NUNN, CO.
06753500.USGS.Streamflow.Month~HydroBase
FillRegression(TSID="06753400.USGS.Streamflow.Month", IndependentTSID="0675
3500.USGS.Streamflow.Month")
```

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 step-wise 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.

Edit FillRepeat() Command

This command fills missing data in the time series by repeating non-missing values.
The fill direction can be forward or backward.
Specify the maximum intervals as blank or zero to fill any gap.
The fill start and end, if specified, will limit the period that is filled.
Use standard date/time formats appropriate for the date/time precision of the time series.

TS list: Optional - indicates the time series to process (default=AllTS).

TSID (for TSList=AllMatchingTSID):

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: 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", FillDirection=Forward)
```

FillRepeat

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: <ul style="list-style-type: none"> 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
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: FillUsingDiversionsComments()

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

Version 09.07.02, 2010-08-20

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 FillUsingDiversionsComments() 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):

Diversion Comment not_used Flag Resulting in Additional Zero Values

not_used	Meaning (reason why diversion is zero)
A	Structure is not usable
B	No water is available
C	Water available, but not taken
D	Water taken in another structure

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
A	Active structure with contemporary diversion records
B	Structure abandoned by the court
C	Conditional structure
D	Duplicate; ID no longer used
F	Structure used as FROM number; located in another water district
H	Historical structure only-no longer exists or has records, but has historical data
I	Inactive structure which physically exists but no diversion records are kept
N	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 FillUsingDiversionsComments() Command

This command can be used to fill monthly, daily, and yearly diversions and reservoir releases for the HydroBase input type.
 The diversion comments in HydroBase indicate years when no water was carried for an entire irrigation year.
 Consequently, missing values in diversion time series can be set to zero for the period November to October.
 If a yearly time series is filled, the zero 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 limits flag, if set to True, will cause the average to be recalculated, for use in other fill commands (see CIU note below).
 For example, use True with a fillUsingDiversionsComments() command immediately after reading diversions.
 If the "currently in use" (CIU) flag is used for filling, additional zeros will be added and limits are recalculated a specific way (see documentation).

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 flag).

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 flag).

Recalculate limits: False Optional - recalculate original data limits after fill? (default=False).

Command: `FillUsingDiversionsComments (TSID="", FillFlag="Auto", FillUsingCIU=True, RecalcLimits=False)`

Cancel OK

FillUsingDiversionsComments

FillUsingDiversionsComments() Command Editor

The command syntax is as follows:

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

Command Parameters

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

Parameter	Description	Default
FillFlag	<p>For each value that is filled using the diversion comment <i>not_used</i> information, tag the filled value as follows:</p> <ul style="list-style-type: none"> • If FillFlag is specified as a single character, tag filled values with the 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. <p>The flag can then be used later to label graphs, etc. The flag will be appended to existing flags if necessary.</p>	No flag is assigned.
FillUsingCIU	<p>Indicates whether the “currently in use” (CIU) information is used to fill missing data. This will result in additional zeros at the beginning or end of the time series, 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 RecalcLimits parameter.</p>	False (CIU information is not used to fill missing data).
FillUsingCIUFlag	<p>For each missing data value that is filled using the CIU information, tag the filled value as follows:</p> <ul style="list-style-type: none"> • If FillUsingCIUFlag=Auto is specified, the CIU value (H, I, or N) from HydroBase is used for the flag. • Else if FillUsingCIUFlag is specified, tag filled values with the specified character. <p>The flag can then be used later to label graphs, etc. The flag will be appended to existing flags if necessary.</p>	No flag is assigned.
RecalcLimits	<p>Indicate whether the original data limits for the time series should be recalculated after the zero values are set. Zero values are included in the monthly and annual averages.</p> <p>See the discussion above related to CIU – time series that are impacted by CIU always have their limits recalculated.</p>	False (additional zeros are not considered in the original data averages).

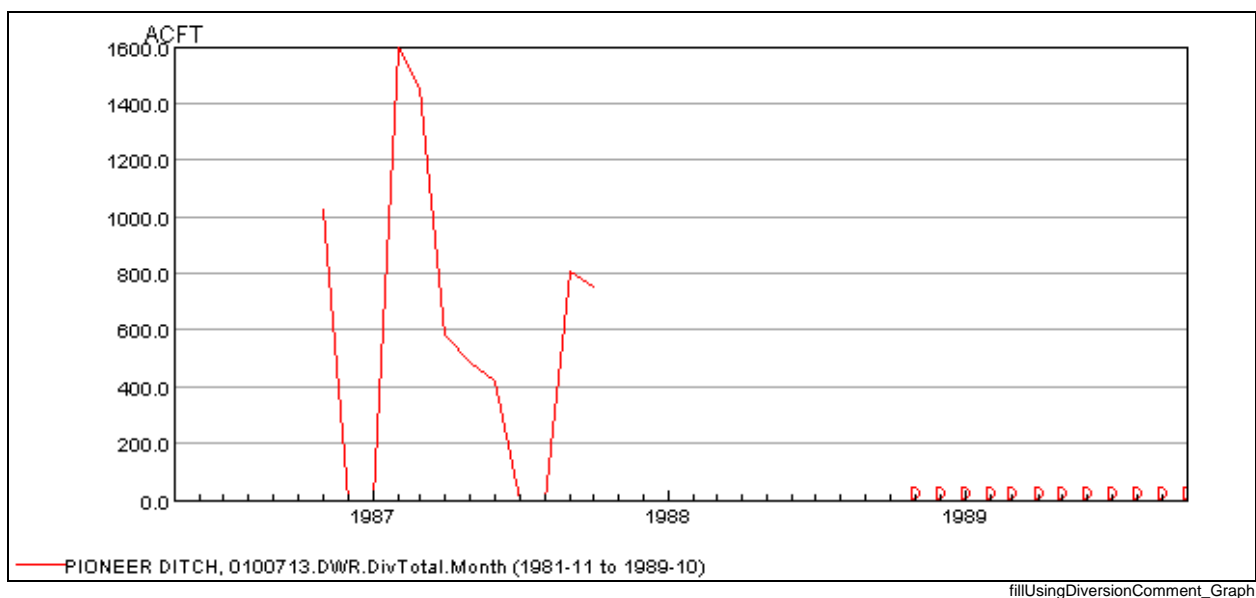
A sample commands file to fill diversion time series from the State of Colorado's HydroBase is as follows:

```
# 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: Free()

Free (remove) time series from memory

Version 08.16.02, 2008-07-28

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.

Edit Free() Command

This command frees (removes) time series, which is useful to remove unneeded or temporary time series.

The list of time series to be removed can be indicated in several ways.

Time series identifiers follow the pattern:
Location.Source.DataType.Interval.Scenario

Examples of wildcard use when `TSLIST=AllMatchingTSID` are shown below:

- * - matches all time series
- ABC* - matches locations starting with ABC
- ABC*.Type.Month - matches locations starting with ABC, with data type Type and interval Month.

Time series that are in an ensemble will be removed from the ensemble.

TS list: Indicates the time series to process (default=AllTS).

TSID (for TSLIST=matching TSID):

EnsembleID (for TSLIST=EnsembleID):

Time series position(s) (for TSLIST=TSPosition): For example, 1,2,7-8 (positions are 1+).

Free ensemble if empty? ☒ True Default (blank) = True.

Command:
`Free(TSLIST=AllMatchingTSID,TSID="40*",FreeEnsembleIfEmpty=True)`

Cancel OK

Free

Free() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none"> 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 list (see TSPosition parameter below). 	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=AllMatchingTSID or TSList=FirstMatchingTSID
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
FreeEnsembleIfEmpty	Indicate whether to free the ensemble from which time series were removed, if the ensemble is empty (has no time series).	True

A sample command file is as follows:

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

Command Reference: FTPGet()

Retrieve file(s) from a remote system using file transfer protocol (FTP)

Version 08.16.00, 2008-16-00

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.

Edit FTPGet() command

This command uses file transfer protocol (FTP) to retrieve files from a remote site and save on the local file system. It is recommended that the local folder name be relative to the working directory, which is:
C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\FTPGet

Remote site:

Login: Case-sensitive (default=anonymous).

Password: Case-sensitive (default=anonymous).

Remote folder: Default is / on FTP server.

File pattern: Default is *.

Destination folder:

Transfer mode: Default is Binary

Retry count: Default is 3.

Retry wait: Default is 3 seconds.

Command:

```
FTPGet(RemoteSite="ftp.somebody.com",Login="anonymous",Password="mypassword",RemoteFolder="/outgoing",FilePattern="data.txt",DestinationFolder="Results",TransferMode=ASCII)
```

FTPGet

FTPGet() Command Editor

The command syntax is as follows:

FTPGet (Parameter=Value, ...)

Command Parameters

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

Command Reference: InsertTimeSeriesIntoEnsemble ()

Insert 1+ time series into an existing ensemble

Version 09.05.00, 2009-10-12

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 InsertTimeSeriesIntoEnsemble() Command

Insert 1+ time series into an ensemble.
The time series must have the same data interval and units as existing time series in the ensemble.
The original time series will remain available and can be accessed directly in the ensemble.

TS list: Optional - indicates the time series to process (default=AllTS).

TSID (for TSList=AllMatchingTSID):

EnsembleID (for TSList=EnsembleID):

Receiving ensemble ID: Required - identifier for ensemble to receive time series.

Command:

```
InsertTimeSeriesIntoEnsemble (EnsembleID2="TestEnsemble")
```

InsertTimeSeriesIntoEnsemble

InsertTimeSeriesIntoEnsemble () Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be added to the ensemble, one of: <ul style="list-style-type: none"> 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. 	AllTS
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 from which to retrieve time series, if inserting time series from an ensemble.	Required when 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
TS tsl = NewPatternTimeSeries(NewTSID="tsl..Flow.Year",
    SetStart="1960",SetEnd="2000",Units="ACFT",
    PatternValues="1,2,5,8,,20")
TS ts2 = NewPatternTimeSeries(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

Version 08.17.00, 2008-10-06

The `LagK()` 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**).

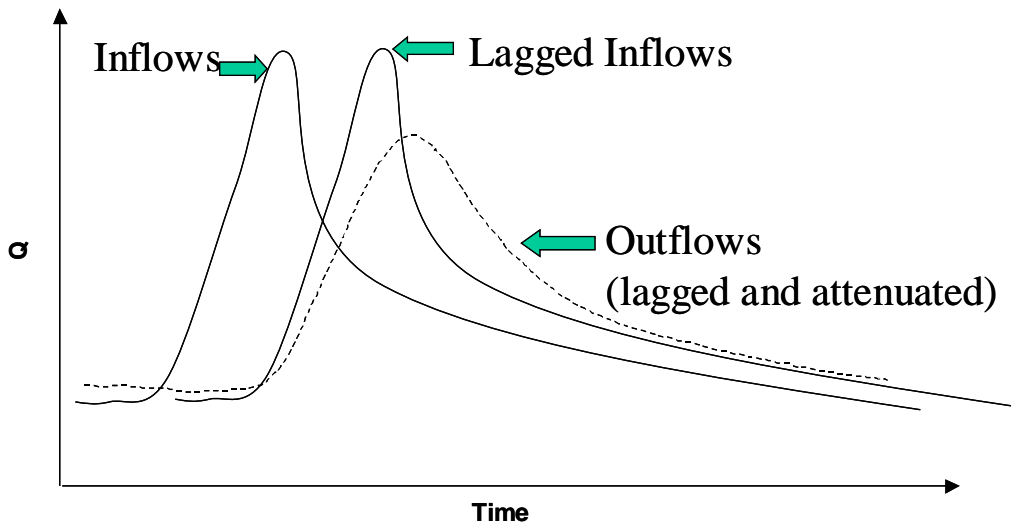


Figure 1: Lag and K Routing

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:

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

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:

Edit TS Alias = LagK() Command

Lag and attenuate a time series, creating a new time series.
 The time series to be routed cannot contain missing values.
 The observed time series is used for filling, then FillNearest, and finally DefaultFlow.
 See the documentation for a complete description of the algorithm.

Time series alias: Often the location from the TSID, or a short string.

Time series to lag (TSID):

Observed time series for filling:

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: Required - lag in time series base interval time units.

K: 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:

```
TS ts1Routed = LagK(TSID="ts1",FillNearest=True,Lag=3,K=2)
```

LagK

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:

```
TS Alias = LagK(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. See also the ObsTSID time series, and the FillNearest and DefaultFlow parameters.	None – must be specified.
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.	
FillNearest	<p>If set to True, then when a missing data value is found anywhere in 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:</p> <ul style="list-style-type: none"> • \leq Seconds: 1000 intervals • Minute, Hour: 1 day • Day: 1 Week • $>$ Day: 1 interval only <p>The assumption is that a flow value close in time will be representative of the missing value and will not result in significant errors.</p> <p>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.</p>	False
DefaultFlow	A flow value in the units of the input time series that is substituted for 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.	0
Lag	Lag time for the modeled reach in the units of the TSID time series 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.	Required
K	Attenuation factor to be applied to the wave. The units of K are time, and like the Lag value, it is assumed to have the same units as the input time series.	Required
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. <i>Note: Specifying values that are not consistent with the Lag and K parameters will result in oscillation!</i>	0 for each value
OutflowStates	Comma-delimited list of default outflow values prior to the start of the time series. See InflowStates for details.	0 for each value

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

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

Command Reference: ManipulateTableString()

Manipulate string a string column in a table

Version 09.09.00, 2010-09-23

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`) and either a second input column name (`InputColumn2`) or a constant string value (`InputValue2`), with the result being placed in the output column (`OutputColumn`). Missing/blank input will be considered as empty strings when formatting the output.

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 ManipulateTableString() Command

Perform simple manipulation on columns of string 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 enhancements may provide more cell range addressing - currently full columns are processed.

Table ID: Required - table to process.

Input column 1: Required - first input column name.

String operator: Required - string manipulation to perform on input.

Input column 2: Required if no input value 2 - second input column name.

Input value 2: Required if no input column 2 - constant string.

Output column: Required - output column name.

Command:

```
ManipulateTableString(TableID="Table1", InputColumn1="String1", Operator="Prepend", InputColumn2="String2", OutputColumn="String3")
```

ManipulateTableString

ManipulateTableString() Command Editor

The command syntax is as follows:

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

Command Parameters

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 input.	None – must be specified.
Operator	The operation to perform on the input strings. For example, Append will append the second input to the first input.	None – must be specified.
InputColumn2	The name of a column containing strings, as the second input.	Required if InputValue2 is not specified.
InputValue2	A string constant, as the second input.	Required if InputColumn2 is not specified.
OutputColumn	The name of a column to receive the output.	None – must be specified.

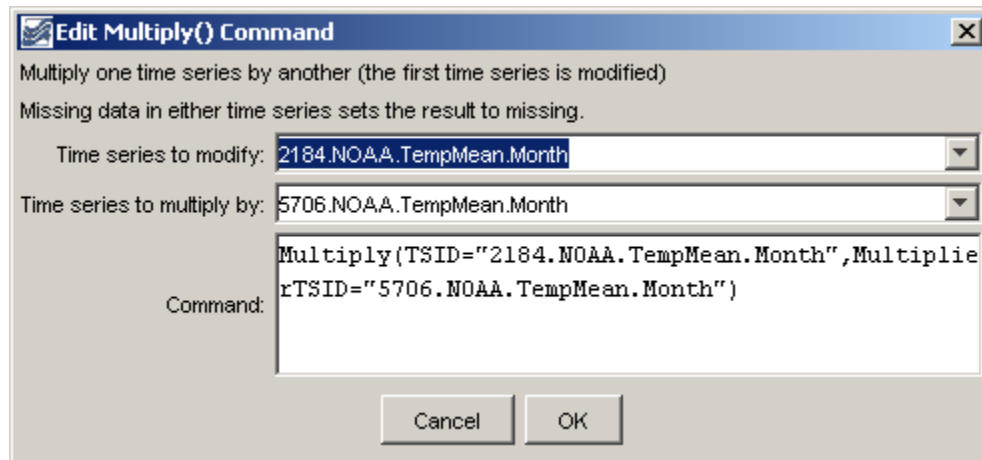
Command Reference: Multiply()

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

Version 08.16.04, 2008-09-24

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.



Multiply() Command Editor

Multiply

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TSID	The time series identifier or alias for the time series to be modified.	None – must be specified.
MultiplierTSID	The time series identifier or alias for the time series that is the multiplier.	None – must be 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: TS Alias = NewDayTSFromMonthAndDayTS()

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

Version 08.16.04, 2008-09-23

The `NewDayTSFromMonthAndDayTS ()` command creates a new daily time series by distributing a monthly time series 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:

$$DayTS2_i = MonthTS2 \frac{ACFT}{NDAYS} * \left(\frac{1DAY}{86400s} \right) \left(\frac{43560FT^2}{1AC} \right) * \left(\frac{DayTS1_i}{\sum_{i=1}^{i=Ndays \sin Month} DayTS1_i} \right)$$

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
ΣDayTS1_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:

$$1001.7 * (1/1.9835) * (14/505) = 14 \text{ CFS}$$

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.

Edit TS Alias = NewDayTSFromMonthAndDayTS() 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.

Time series alias:

New time series ID: Specify to avoid confusion with TSID from original TS.

Monthly time series for total:

Daily time series for distribution:

Command:

```
TS DayTS =
NewDayTSFromMonthAndDayTS(NewTSID="08236000.DWR.Streamflow.Day",
MonthTSID="08236000.DWR.Streamflow.Month",DayTSID="08236500.DWR.Streamflow.Day")
```

NewDayTSFromMonthAndDayTS_Alias

TS Alias = NewDayTSFromMonthAndDayTS() Command Editor

The command syntax is as follows:

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

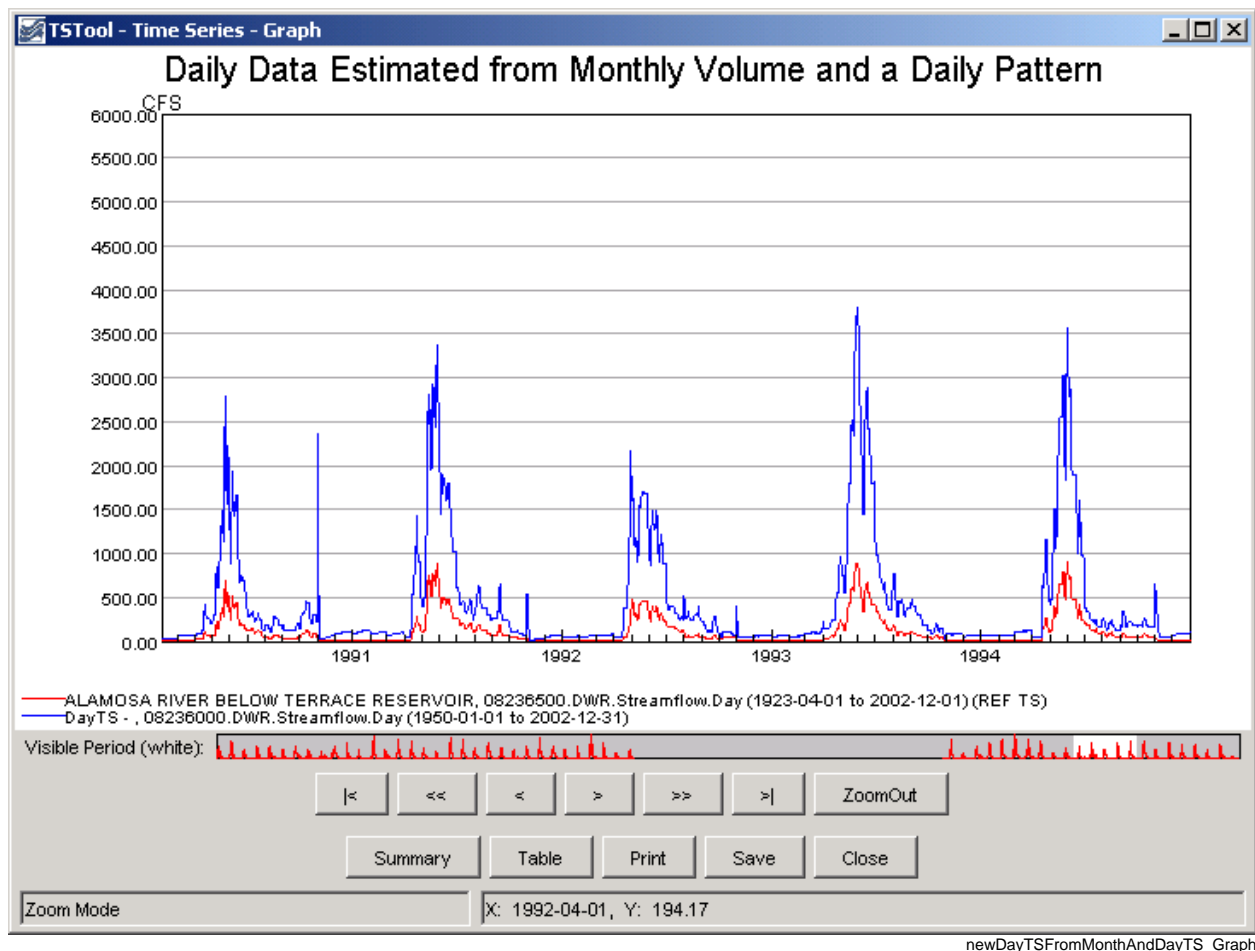
Command Parameters

Parameter	Description	Default
Alias	The alias of the new time series.	None – must be specified.
NewTSID	The time series identifier of the new time series. The interval must be Day.	None – must be specified.
MonthTSID	The time series identifier or alias for a monthly time series supplying monthly ACFT values.	None – must be specified.
DayTSID	The time series identifier or alias for a daily time series supplying daily flow values (only the pattern is used).	None – must be specified.

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

```
SetOutputPeriod(OutputStart="1950-01",OutputEnd="2002-12")
# 08236500 - ALAMOSA RIVER BELOW TERRACE RESERVOIR
08236500.DWR.Streamflow.Day~HydroBase
# 08236000 - ALAMOSA RIVER ABOVE TERRACE RESERVOIR
08236000.DWR.Streamflow.Month~HydroBase
TS DayTS = NewDayTSFromMonthAndDayTS(NewTSID="08236000.DWR.Streamflow.Day",
MonthTSID="08236000.DWR.Streamflow.Month",
DayTSID="08236500.DWR.Streamflow.Day")
```

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: TS Alias = NewEndOfMonthTSFromDayTS()

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

Version 08.16.04, 2008-09-23

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.

Edit TS Alias = 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 (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.

Time series alias:

Daily time series:

Number of days to search: Required - must be greater than zero.

Command:

```
TS Continental =  
NewEndOfMonthTSFromDayTS(DayTSID="2003536.DWR.ResMeasStorage.Day",Bracket="15")
```

NewEndOfMonthTSFromDayTS_Alias

TS Alias = NewEndOfMonthTSFromDayTS() Command Editor

The command syntax is as follows:

```
TS Alias = NewEndOfMonthTSFromDayTS (Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
Alias	The alias for the new time series.	None – must be specified.
DayTSID	The time series identifier or alias of the daily time series to be searched for data.	None – must be specified.
Bracket	The number of days to search from the end of the month, in order to find a daily value to transfer to the end of the month.	None – must be specified.

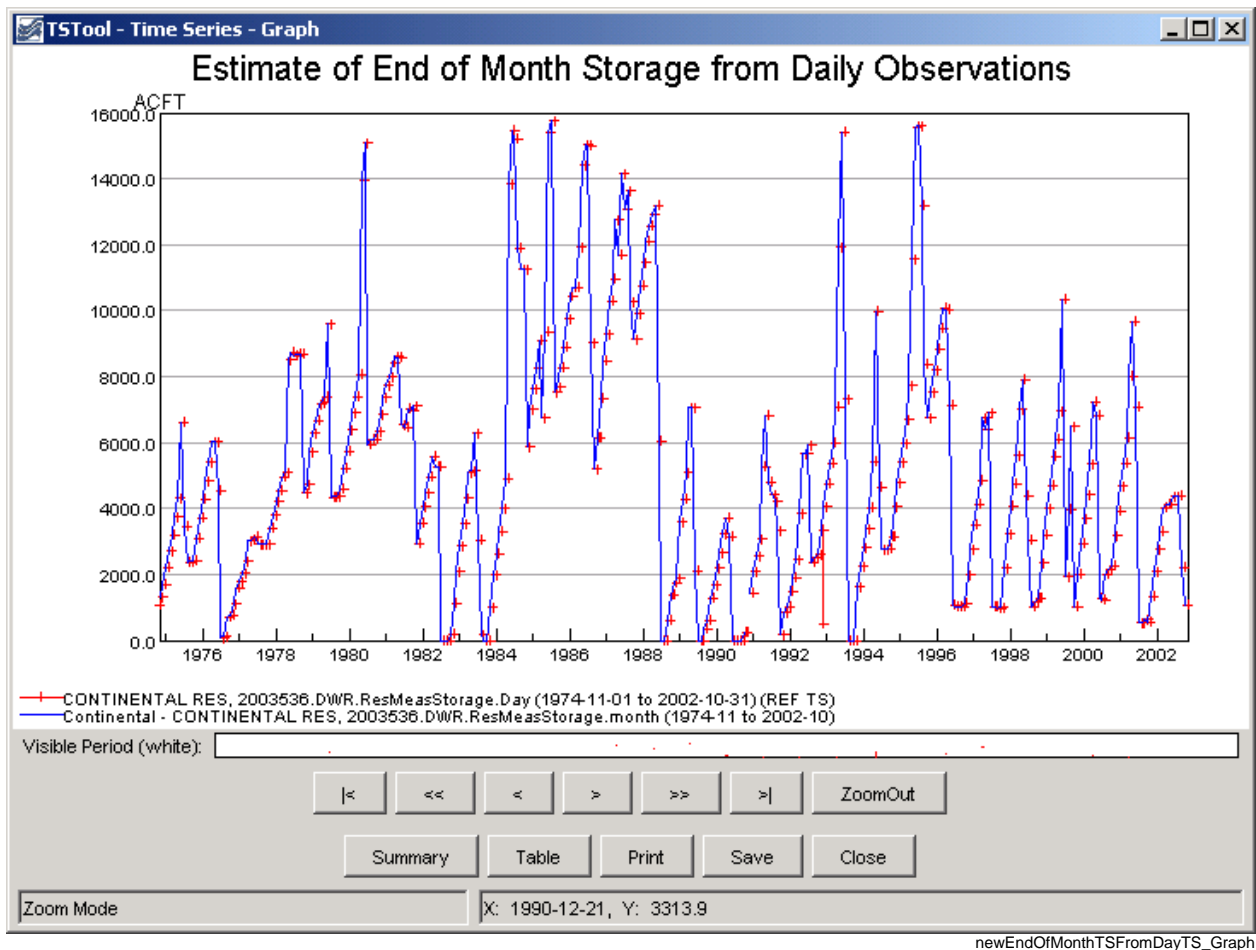
A sample command file for estimating reservoir contents, using data from the State of Colorado's HydroBase database is:

```
# 2003536 - CONTINENTAL RES
2003536.DWR.ResMeasStorage.Day~HydroBase
TS Continental = NewEndOfMonthTSFromDayTS(DayTSID="2003536.DWR.ResMeasStorage.Day",
Bracket=15)
```

A sample command file for estimating well levels is:

```
# 384549104445101 - SCO1506611ABC
384549104445101.USGS.WellLevel.Day~HydroBase
TS WellMonth =
NewEndOfMonthTSFromDayTS(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 09.05.00, 2009-10-12

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.

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 and units.
The original time series will remain available and can be accessed directly in the ensemble.
Possible future enhancements (currently disabled):
Copy the time series to isolate the ensemble from additional changes to the time series.
Specifying the input period for copied time series.

TS list: Optional - indicates time series to add to new ensemble (default=none).

TSID (for TSList=AllMatchingTSID):

EnsembleID (for TSList=EnsembleID):

New ensemble ID: Required - identifier for new ensemble.

New ensemble name: 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 (TSList=AllTS,NewEnsembleID="TestEnsemble",NewEnsembleName="Test Ensemble")`

NewEnsemble

NewEnsemble () Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none">AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards).	AllTS

	<ul style="list-style-type: none"> • 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 using time series in the main time series list. This protects the data in the ensemble from general processing commands. Envisioned as future enhancement.	Associate time series in the main time series list with the new ensemble.

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
TS ts1 = NewPatternTimeSeries(NewTSID="ts1..Flow.Year",
    SetStart="1960",SetEnd="2000",Units="ACFT",
    PatternValues="1,2,5,8,,20")
TS ts2 = NewPatternTimeSeries(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: TS Alias = NewPatternTimeSeries()

Create a new time series containing a pattern of repeating values

Version 08.15.00, 2008-05-04

The `TS Alias = 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.

Edit TS Alias = NewPatternTimeSeries() Command

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/times using a precision consistent with the data interval.
If the time series has an interval of irregular, provide the interval to define data (more options for irregular data may be added later).

Time series alias: Can use in other commands instead of TSID.
New time series ID: Specify to avoid confusion with TSID from original TS.

Interval for irregular time series: Use to initialize data (irregular time series only).

Description/Name:

Start: Starting date/time for time series (blank=setOutputPeriod() start).
End: Ending date/time for time series (blank=setOutputPeriod() end).
Data units: For example: ACFT, CFS, IN.
Pattern values: Separate by spaces or commas (default is all missing values).

Command:

```
TS Alias = NewPatternTimeSeries(NewTSID="MyLoc..MyData.Day",Description="Test data",SetStart="1950-01-01",SetEnd="1951-03-12",Units="CFS",PatternValues="5,10,12,13,75")
```

NewPatternTimeSeries

TS Alias = NewPatternTimeSeries() Command Editor

The command syntax is as follows:

`TS Alias = NewPatternTimeSeries(Parameter=Value,...)`

Command Parameters

Parameter	Description	Default
Alias	The alias of the new time series, which can be used instead of the TSID in other commands.	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.	None.
IrregularInterval	Interval to use to populate irregular time series (e.g., 1Hour, Month), necessary because data need to be assigned somehow.	None – must be specified for irregular time series.
Description	Description for the time series.	None.
SetStart	Start date/time to set data.	None – must be

Parameter	Description	Default
		specified.
SetEnd	End date/time to set data.	None – must be specified.
Units	Units for the data values.	None.
PatternValues	Data values, separated by commas.	None – must be specified.

Examples

The following example commands file illustrates how to create a pattern time series for testing:

```
TS Alias = NewPatternTimeSeries(NewTSID="MyLoc..MyData.Day",  
    Description="Test data",SetStart="1950-01-01",  
    SetEnd="1951-03-12",Units="CFS",PatternValues="5,10,12,13,75")  
WriteDateValue(OutputFile="Results\Example_NewPatternTimeSeries_out.dv")
```


Command Reference: TS Alias = NewStatisticTimeSeries()

Create a time series containing a repeating year of statistics determined from a time series

Version 09.05.01, 2009-10-19

The `TS Alias = 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 will be averaged and the resulting January 1 values for the entire time series will be 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 and are only visible 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 TS Alias = NewStatisticTimeSeries() Command

Create a time series as a repeating statistic determined from the input time series.
The statistic is computed from a sample consisting of values from the same date/time in each year of the time series.
For example, the Mean statistic applied to a daily time series will result in the output time series having the mean of all January 1 data on each January 1 in the result.
A new time series identifier can be assigned and is highly recommended if there is any chance that the new time series will be mistaken for the original.

Time series alias: Required - often the location from the TSID, or a short string.

Time series to analyze (TSID):

New time series ID: Recommended - specify to avoid confusion with TSID from original TS.

Statistic: 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:

```
TS ts1_mean =  
NewStatisticTimeSeries(TSID="ts1",NewTSID="ts1..Streamflow.Month.Mean",Statistic  
=Mean)
```

NewStatisticTimeSeries

TS Alias = NewStatisticTimeSeries() Command Editor

The command syntax is as follows:

```
TS Alias = NewStatisticTimeSeries(Parameter=value,...)
```

Command Parameters

Parameter	Description	Default
Alias	The alias of the new time series, which can be used instead of the TSID in other commands.	None – must be specified.
TSID	The time series identifier (or alias) of the time series to analyze.	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.	None – use the same identifier as the original time series.
Statistic	See the Available Statistics table below.	None – must be specified.
Allow Missing Count	The number of missing values allowed in the source interval(s) in order to produce a result. This capability should be used with care because it may result in data that are not representative of actual conditions.	Allow any number of missing values.
MinimumSampleSize	The minimum number of values required in the sample to compute the statistic. If the minimum sample size is not available, the result will be set to missing.	Minimum sample size is defined by the statistic.
AnalysisStart	The date/time for the analysis start, using a precision that matches the original time series. This controls the sample size.	Analyze the full period.
AnalysisEnd	The date/time for the analysis start, using a precision that matches the original time series. This controls the sample size.	Analyze the full period.
OutputStart	The date/time for the output start, using a precision that matches the original time series. The repeating statistic will fill this period.	Output the full period.
OutputEnd	The date/time for the analysis start, using a precision that matches the original time series. The repeating statistic will fill this period.	Output the full period.

Available Statistics

Statistic	Description	Limitations
Max	Maximum of all values in the sample.	None.
Mean	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.

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
TS ts1 = NewPatternTimeSeries(NewTSID="ts1..Streamflow.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
TS ts2 = NewPatternTimeSeries(NewTSID="ts2..Streamflow.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")
TS ts1_mean = NewStatisticTimeSeries(TSID="ts1",NewTSID="ts1..Streamflow.Month.Mean",
    Statistic=Mean)
TS ts2_mean = NewStatisticTimeSeries(TSID="ts2",NewTSID="ts2..Streamflow.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

Graph Summary Save Close

Currently-selected worksheet interval: Month

Command Reference: TS Alias = NewStatisticTimeSeriesFromEnsemble()

Create a time series containing a statistic determined from a time series ensemble

Version 09.05.01, 2009-10-26

The `TS Alias = 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 TS Alias = NewStatisticTimeSeriesFromEnsemble() 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 series identifier (TSID) be specified for the result to avoid confusion with the original time series.

Time series alias: Required - often the location from the TSID, or a short string.

Ensemble to analyze (EnsembleID):

New time series ID: Specify to avoid confusion with TSID from original TS.

Statistic: 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:
TS Mean =
NewStatisticTimeSeriesFromEnsemble (EnsembleID="TestEnsemble", NewTSID="Test..Streamflow.Month.Mean", Statistic=Mean)

NewStatisticTimeSeriesFromEnsemble

TS Alias = NewStatisticTimeSeriesFromEnsemble() Command Editor

The command syntax is as follows:

```
TS Alias = NewStatisticTimeSeriesFromEnsemble(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
Alias	The alias of the new time series, which can be used instead of the TSID in other commands.	None – must be specified.
EnsembleID	The identifier for the ensemble to analyze.	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. This parameter may be required in the future.	None – use the same identifier as the original time series.
Statistic	The statistic to compute. See the Available Statistics table below.	None – must be specified.
Allow Missing Count	The number of missing values allowed in the sample of values in order to produce a result. This capability should be used with care because it may result in data that are not representative of actual conditions.	Missing values are ignored in the sample used to compute the statistic.
MinimumSample Size	The minimum number of values in the sample that are required to compute the statistic.	Use the sample with no restrictions, although some statistics may have requirements.
AnalysisStart	The date/time for the analysis start, using a precision that matches the original time series.	Analyze the full period.
AnalysisEnd	The date/time for the analysis start, using a precision that matches the original time series.	Analyze the full period.
OutputStart	The date/time for the output start, using a precision that matches the original time series. An output period longer than the analysis period will result in missing values in output.	Output the full period.
OutputEnd	The date/time for the output start, using a precision that matches the original time series. An output period longer than the analysis period will result in missing values in output.	Output the full period.

Available Statistics

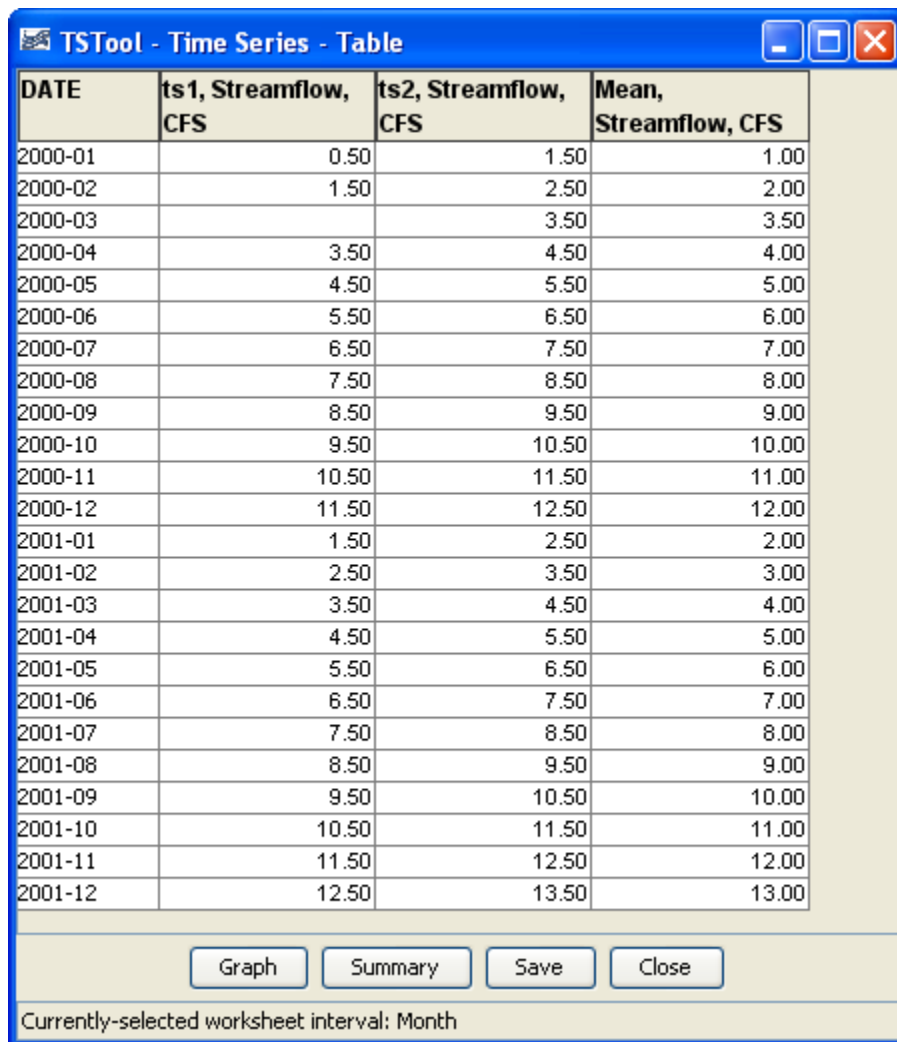
Statistic	Description	Limitations
Max	Maximum of all values in the sample.	None.
Mean	Mean of all values in the sample.	None.
Median	Median of all values in the sample.	None.
Min	Minimum of all values in the sample.	None.

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.
TS ts1 = NewPatternTimeSeries(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")
TS ts2 = NewPatternTimeSeries(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
TS Mean = NewStatisticTimeSeriesFromEnsemble(EnsembleID="TestEnsemble",
    NewTSID="Test..Streamflow.Month.Mean",
    Statistic=Mean)
```

The following figure illustrates the results:



DATE	ts1, 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		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	11.00
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

Graph Summary Save Close

Currently-selected worksheet interval: Month

NewStatisticTimeSeriesFromEnsemble_Table

NewStatisticTimeSeriesFromEnsemble() Command Results

Command Reference: TS Alias = NewStatisticYearTS()

Create a new yearly time series containing a statistic determined from each year of another time series

Version 09.06.02, 2010-03-11

The `TS Alias = 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.

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

Edit TS Alias = 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 time series identifier (TSID) information be specified for the output time series to avoid confusing the output with the original.

Time series alias: Required - for output, typically the location from the TSID, or a short string.

Time series to analyze (TSID):

New time series ID: Recommended - to avoid confusion with TSID from original time series.

Statistic: Required - statistic to calculate.

Test value: 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: Optional - analysis window within input year (default=full year).

Search start: Optional - search start (needed for some statistics, default=full year).

Command:

```
TS 3553_FrostDateL28S =
NewStatisticYearTS(TSID="3553.NOAA.TempMin.Day",NewTSID="3553.NOAA.FrostDateL28S.Year",Statistic=DayOfLastLE,TestValue=28,SearchStart="06/30")
```

NewStatisticYearTS

TS Alias = NewStatisticYearTS() Command Editor

The command syntax is as follows:

```
TS Alias = NewStatisticYearTS(Parameter=value,...)
```

Command Parameters

Parameter	Description	Default
Alias	The alias of the new time series, which can be used instead of the TSID in other commands.	None – must be specified.
TSID	The time series identifier (or alias) of the time series to analyze.	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.
AllowMissingCount	The number of missing values allowed in the source 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.	Allow any number of missing values.
MinimumSampleSize	The minimum sample size in order to compute the statistic.	No minimum, 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 the time series interval. This will limit the data being analyzed at the ends of the time series and controls the length of the output time series. The analysis period is typically set to align with years consistent with the output year type.	Analyze the full period, extending the period to include full years.
AnalysisEnd	The ending date/time for the analysis using	Analyze the full

Parameter	Description	Default
	calendar dates (e.g., 2001-01-01) , with precision consistent with the time series interval. This will limit the data being analyzed at the ends of the time series and controls the length of the output time series. The analysis period is typically set to align with years consistent with the output year type.	period, extending the period to include full years.
AnalysisWindowStart	The calendar date/time for the analysis start 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 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.	Analyze the full year.
AnalysisWindowEnd	Specify date/time for the analysis end within each year. See AnalysisWindowStart for details.	Analyze the full year.
SearchStart	Within the analysis window, this indicates the starting date/time for the search. 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. This parameter is useful in cases where the processing considers seasonal aspects of the analysis window; for example, use when determining frost dates (when temperature is 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*.	Use the analysis window start and end. Search forward for most statistics. Search backward for DayOfLast* and MonthOfLast* statistics.

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
DayOfFirstGE	Julian day of the year (1-366, relative to the start of the OutputYearType) for the first data value >= TestValue. Searches start at the start of the analysis window and move forward.	Input time series must be daily or smaller interval.
DayOfFirstGT	Similar to DayOfFirstGE, for values > TestValue.	Input time series must be daily or smaller interval.

Statistic	Description	Limitations
DayOfFirstLE	Similar to DayOfFirstGE, for values \leq TestValue.	Input time series must be daily or smaller interval.
DayOfFirstLT	Similar to DayOfFirstGE, for values $<$ TestValue.	Input time series must be daily or smaller interval.
DayOfLastGE	Julian day of the year (1-366, relative to the start of the OutputYearType) for the last data value \geq TestValue. Searches start at the start of the analysis window and move backward.	Input time series must be daily or smaller interval.
DayOfLastGT	Similar to DayOfLastGE, for values $>$ TestValue.	Input time series must be daily or smaller interval.
DayOfLastLE	Similar to DayOfLastGE, for values \leq TestValue.	Input time series must be daily or smaller interval.
DayOfLastLT	Similar to DayOfLastGE, for values $<$ TestValue.	Input time series must be daily or smaller interval.
DayOfMax	Julian day of the year (1-366, relative to the start of the OutputYearType) for the first maximum value in the time series.	Input time series must be daily or smaller interval.
DayOfMin	Julian day of the year (1-366, relative to the start of the OutputYearType) for the first minimum value in the time series.	Input time series must be daily or smaller interval.
GECount	Count of values in a year \geq TestValue.	
GEPercent	Percent of values in a year \geq TestValue, based on the total number of points in the year.	
GTCCount	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 \leq TestValue.	
LEPercent	Percent of values in a year \leq TestValue, based on the total number of points in the year.	
LTCCount	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 missing values in a year.	
MonthOfFirstGE	Month the year (1-12, relative to the start of the OutputYearType) for the first data value \geq TestValue. Searches start at the start of the analysis window and move forward.	Input time series must be monthly or smaller interval.
MonthOfFirstGT	Similar to DayOfFirstGE, for values $>$ TestValue.	Input time series must be monthly or smaller interval.
MonthOfFirstLE	Similar to DayOfFirstGE, for values \leq TestValue.	Input time series must be monthly or smaller interval.
MonthOfFirstLT	Similar to DayOfFirstGE, for values $<$ TestValue.	Input time series must be monthly or smaller

Statistic	Description	Limitations
		interval.
MonthOfLastGE	Month of the year (1-12, relative to the start of the OutputYearType) for the last data value >= TestValue. Searches start at the end of the analysis window and move backward.	Input time series must be monthly or smaller interval.
MonthOfLastGT	Similar to DayOfLastGE, for values > TestValue.	Input time series must be monthly or smaller interval.
MonthOfLastLE	Similar to DayOfLastGE, for values <= TestValue.	Input time series must be monthly or smaller interval.
MonthOfLastLT	Similar to DayOfLastGE, for values < TestValue.	Input time series must be monthly or smaller interval.
MonthOfMax	Month of the year (1-12, relative to the start of the OutputYearType) for the first maximum value in the time series.	Input time series must be monthly or smaller interval.
MonthOfMin	Month of the year (1-12, relative to the start of the OutputYearType) for the first minimum value in the time series.	Input time series must be monthly or smaller interval.
Total	Total of values in 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
TS 3553_FrostDateL28S = NewStatisticYearTS(TSID="3553.NOAA.TempMin.Day",
    NewTSID="3553.NOAA.FrostDateL28S.Year",
    Statistic=DayOfLastLE,TestValue=28,
    SearchStart="06/30")
TS 3553_FrostDateL32S = NewStatisticYearTS(TSID="3553.NOAA.TempMin.Day",
    NewTSID="3553.NOAA.FrostDateL32S.Year",
    Statistic=DayOfLastLE,TestValue=32,
    SearchStart="06/30")
TS 3553_FrostDateF32F = NewStatisticYearTS(TSID="3553.NOAA.TempMin.Day",
    NewTSID="3553.NOAA.FrostDateF32F.Year",
    Statistic=DayOfFirstLE,TestValue=32,
    SearchStart="07/01")
TS 3553_FrostDateF28F = NewStatisticYearTS(TSID="3553.NOAA.TempMin.Day",
    NewTSID="3553.NOAA.FrostDateF28F.Year",
    Statistic=DayOfFirstLE,TestValue=28,
    SearchStart="07/01")
Free(TSID="*.TempMin.*")
WriteStateCU(OutputFile="Results/Test.FrostDates")

```

This page is intentionally blank.

Command Reference: NewTable ()

Create a new table

Version 09.04.02, 2009-07-28

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 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 and time
- double - double precision number
- float - single precision number
- integer - integer
- short - short integer
- string

Table ID: Required - unique identifier for the table.

Column definitions:

Command:

NewTable

NewTable () Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TableID	Identifier for the table – should be unique among tables that are defined.	None – must be specified.
Columns	<p>The column names and data types are defined using the format <code>ColumnName,DataType;</code> <code>ColumnName,DataType</code>. 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:</p> <ul style="list-style-type: none">• <code>datetime</code> – date and time• <code>double</code> – double precision number• <code>float</code> – single precision number• <code>integer</code> – integer (-2147483648 to 2147483647)• <code>short</code> – short integer (-32768 to 32767)• <code>string</code> – Unicode string	No columns will be defined.

Command Reference: TS Alias = NewTimeSeries()

Create a new time series

Version 09.08.01, 2010-09-15

The `NewTimeSeries()` command creates a new time series in memory and assigns it an alias. This time series can then be manipulated (e.g., added to, filled). This 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.

The following dialog is used to edit the command and illustrates the syntax for the command. The new time series identifier is edited by pressing the **Edit** button.

Edit TS Alias = 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.

Time series alias: Required - use in other commands instead of TSID.

New time series ID: Required - specify to avoid confusion with TSID from original TS.

Description/Name:

Start: Optional - starting date/time for time series (default=SetOutputPeriod() start).

End: Optional - ending date/time for time series (default=SetOutputPeriod() end).

Data units: Optional - for example: ACFT, CFS, IN (default=no units).

Initial value: Optional - default is initialize with missing value.

Command:

```
NewTimeSeries(NewTSID="Station1.MyModel.Streamflow.Month", Description="Example Description", SetStart="1950-01", SetEnd="2002-12", Units="CFS", InitialValue=20)
```

NewTimeSeries

NewTimeSeries() Command Editor

The command syntax is as follows:

```
TS Alias = NewTimeSeries(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
Alias	The alias of the new time series, which can be used in place of the TSID in other commands.	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, interval).
Description	The description for the time series, used in output.	Blank.
SetStart	The start of the time series data period, or blank to use the output period defined with the SetOutputPeriod() command.	Use the start from SetOutputPeriod().
SetEnd	The end of the time series data period, or blank to use the output period defined with the SetOutputPeriod() command.	Use the end from SetOutputPeriod().
Units	Data units for the time series.	Blank.
InitialValue	The initial value to populate the time series.	Initialize the time series to missing data.

The example commands 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 are not standardized.

```
SetOutputPeriod(OutputStart="1950-01",OutputEnd="2002-12")
TS station1 = NewTimeSeries(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

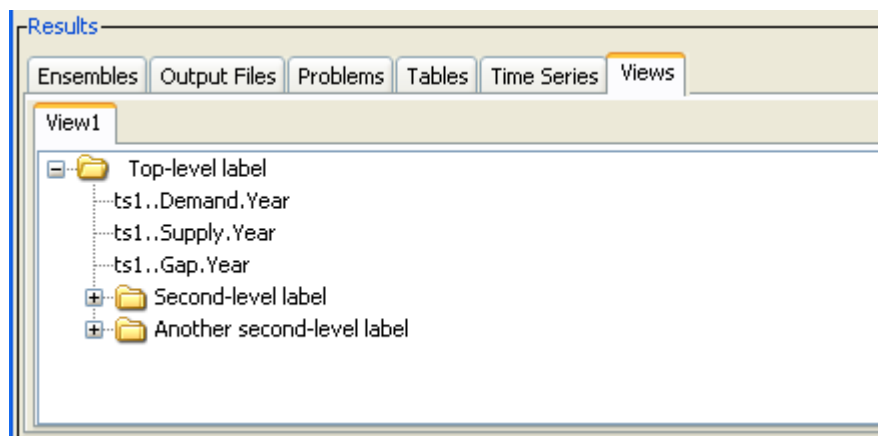
The `NewTreeView()` command creates a tree view, which is a hierarchical listing of time series. The resulting view is displayed in the **Views** section of the TSTool **Results** area and provides interactive access to data. The view is defined using a simple text file, as shown in the following example:

```
# 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.

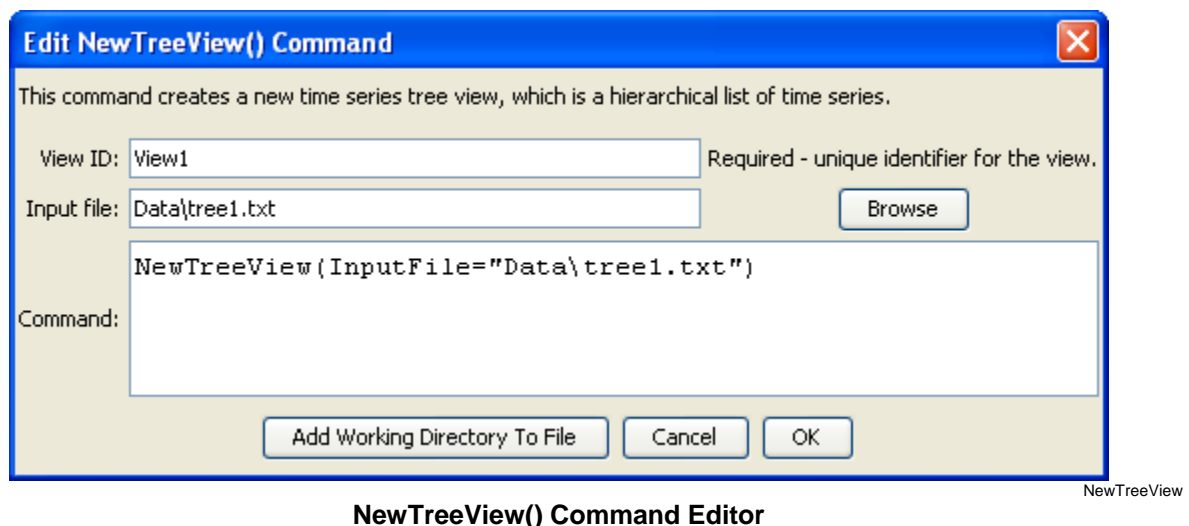
The following figure illustrates the resulting view that is displayed in TSTool for the above example, using contrived data. The time series in the tree view can be selected and a pop-up menu can be used to generate graphs. Consequently, the view allows the results of processing to be presented in a 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.



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.



NewTreeView() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
ViewID	Identifier to assign to the view, which allows the view to be used with other commands.	None – must be specified.
InputFile	The name of the view definition file to read, as an absolute path or relative to the command file location.	None – must be specified.

Command Reference: TS Alias = Normalize()

Create a normalized time series

Version 08.16.04, 2008-09-22

A `Normalize()` command can be inserted to create 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 TS Alias = Normalize() Command

Create a new time series by normalizing the data from a time series.
Use the alias to reference the new time series. Data units are set to blank because the result is dimensionless.

Time series alias:

Time Series to Normalize:

Minimum data value to process: Required.

Minimum output value: Required - for example 0.0.

Maximum output value: Required - for example 1.0.

Command:

```
TS NormalizedTS =  
Normalize(TSID="06730500.USGS.Streamflow.Month",MinValueMethod=MinFromTS,MinValue=0.0,MaxValue=1.0)
```

normalize

Normalize() Command Editor

The command syntax is as follows:

```
TS Alias = Normalize(Parameter=Value,...)
```

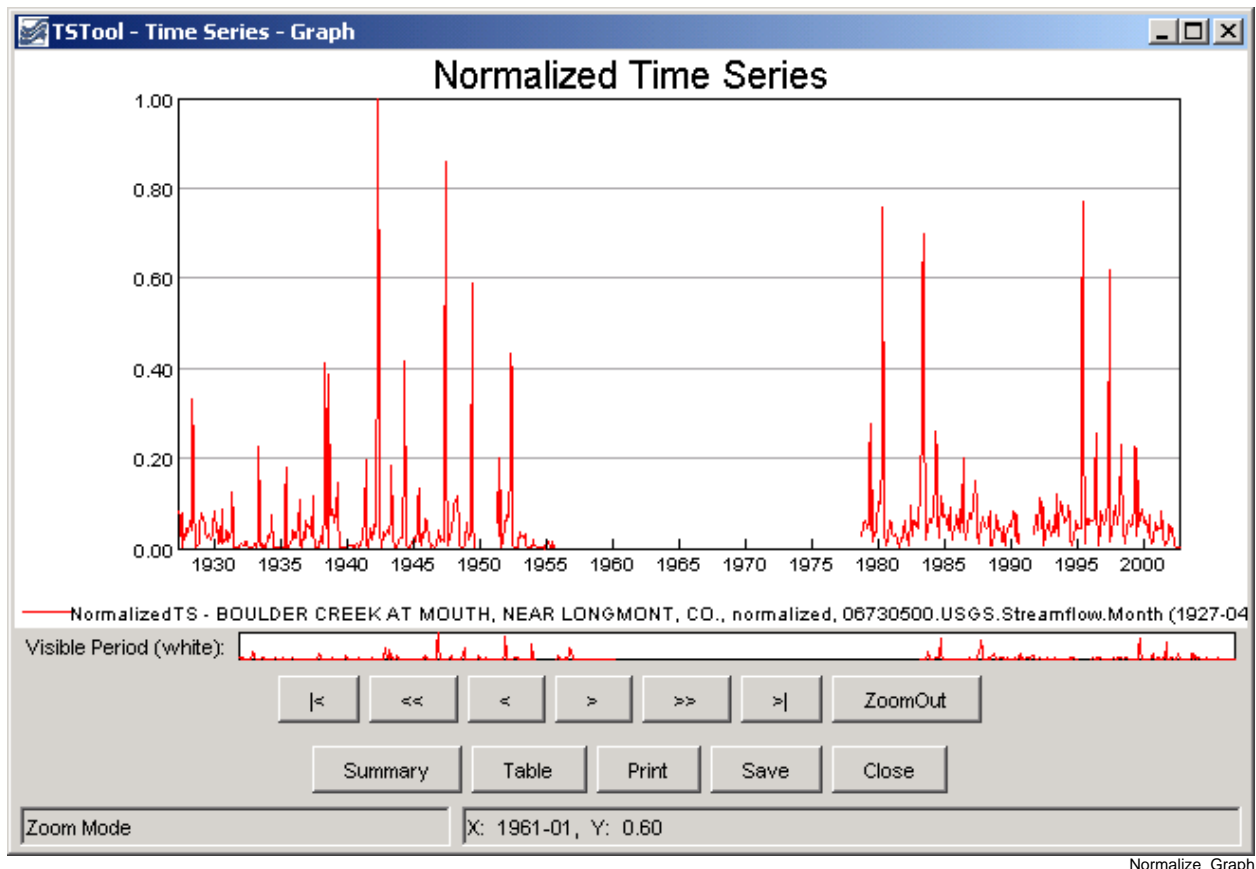
Command Parameters

Parameter	Description	Default
Alias	The alias for the new time series.	None – must be specified.
TSID	The time series identifier or alias for the time series to be normalized.	None – must be specified.
MinValue Method	Indicates how to determine the minimum data value to process, one of: <ul style="list-style-type: none"> MinFromTS – get the minimum value from the time series (typical) MinZero – use zero (e.g., if negative values are to be ignored) 	None – must be specified.
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
TS NormalizedTS = Normalize(TSID="06730500.USGS.Streamflow.Month",
    MinValueMethod=MinFromTS,MinValue=0.0,MaxValue=1.0)
```

The results are as follows:



Results of Normalize() Command

Command Reference: OpenCheckFile()

Open a check file

Version 08.16.03, 2008-08-21

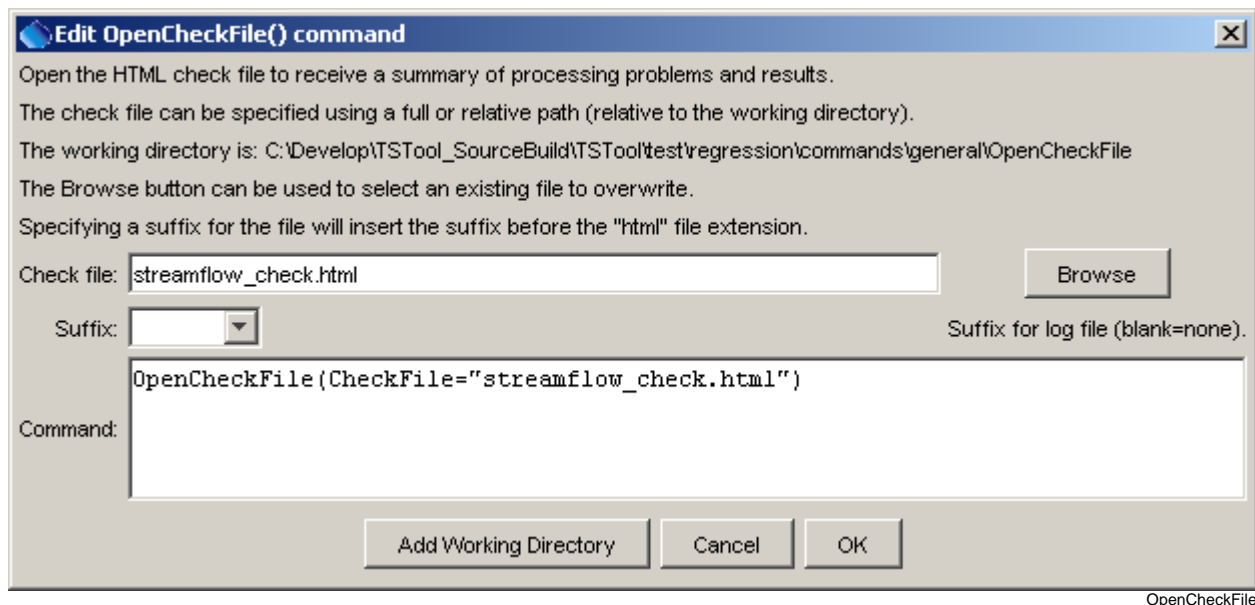
The `OpenCheckFile()` command opens an HTML check file, which is intended to provide a summary of results, in contrast to other feedback features:

- the log file provides a sequential and at times highly-detailed list of information, warning, and optionally debug messages that help troubleshoot processing errors; however, it is often not suitable for end users
- the interactive graphical icons that indicate problems with commands focus on commands and the processing sequence; however, even if all commands run there may be problems with results
- the check file checks the results, providing various tabular lists of issues

This command should be inserted near the top of commands, in order to accumulate checks generated during processing. The information is formatted when command processing is complete and the file can be viewed as results.

A useful standard is to name the check file the same as the command file, with an additional `_check.html` extension. A date or date/time can optionally be added to the check file name.

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



OpenCheckFile() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
CheckFile	The name of the check file to write surrounded by double quotes. The extension of <i>.html</i> will automatically be added if not specified.	None – must be specified.
Suffix	<p>Indicates that a suffix will be added before the <i>.html</i> extension, one of:</p> <ul style="list-style-type: none">▪ Date – add a date suffix of the form YYYYMMDD.▪ DateTime – add a date/time suffix of the form YYYYMMDD_HHMMSS. <p>This is useful for automatically archiving check files corresponding to command files, to allow checking the output at a later time. However, date/time stamping the files may use a lot of disk space for repeated processes.</p>	Do not add the suffix.

A sample command file to open a check file is as follows:

```
OpenCheckFile(CheckFile="streamflow_check.html")
```

Example_OpenCheckFile

Command Reference: OpenHydroBase()

Open a connection to a HydroBase database

Version 09.03.00, 2009-04-12

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 databases, perhaps to compare the contents of the databases – in this case two `OpenHydroBase()` 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 OpenHydroBase() command

This command opens a connection to a HydroBase database, closing the previous connection with the same input name.
This command is used, for example, when making connections to more than one HydroBase database, or running in batch mode.
The RunMode can also be set to control whether the command is run in batch mode, in interactive sessions, or both.
The connection can be made either by specifying a database server/name for SQL Server,
or an ODBC Data Source Name (DSN) for a Microsoft Access HydroBase database (only for very old HydroBase databases).

Database type: Required - indicates whether a database server/name or ODBC DSN is specified below.

Database server: Required - when using SQL Server.

Database name: 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

OpenHydroBase() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
DatabaseServer	Used with a SQL Server HydroBase. Specify the SQL Server database machine name. A list of choices will be shown, corresponding to properties in the <i>CDSS.cfg</i> configuration file.	Required if a SQL Server database is used, and accepts the generic value DatabaseServer=local, which will automatically be translated to the name of the local computer.
DatabaseName	Used with a SQL Server HydroBase. The name of the database typically follows a pattern similar to: HydroBase_CO_YYYYMMDD. A list of choices will be shown, corresponding to properties in the <i>CDSS.cfg</i> configuration file.	HydroBase
OdbcDsn	The ODBC DSN to use for the connection, used only when working with a Microsoft Access database.	Required if a Microsoft Access database is used.
InputName	The input name corresponding to the ~InputType~InputName information in time series identifiers. This is used when more than one HydroBase connection is used in the same commands file.	Blank (no input name).
UseStoredProcedures	Used with SQL Server, indicating whether stored procedures are used. Stored procedures are the default and should be used except when testing software.	True (used stored procedures).
RunMode	Indicates when the command should be run, one of: 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.	GUIAndBatch

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")
TS ts = ReadHydroBase(TSID="BOXHUDCO.DWR.Streamflow.Month")
```

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")
TS ts_sp =
ReadHydroBase(TSID="BOXHUDCO.DWR.Streamflow.Month~HydroBase~SP")
TS ts_nosp =
ReadHydroBase(TSID="BOXHUDCO.DWR.Streamflow.Month~HydroBase~NoSP")
```

Example_OpenHydroBase_TwoConnections.TSTool

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")
```

This page is intentionally blank.

Command Reference: ProcessTSProduct()

Process a time series product file to produce output

Version 09.07.02, 2010-08-20

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.

The following dialog is used to edit the ProcessTSPProduct () 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 ProcessTSPProduct() 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\doc\Training\02-intro-CDSS-Data\example2-ColoradoWaterSMS

TS product file (TSP): Example_ProcessTSPProduct.tsp Browse

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).

Command:

```
ProcessTSPProduct {TSPProductFile="Example_ProcessTSPProduct.tsp"}
```

Add Working Directory to TSP
Cancel
OK

ProcessTSPProduct

ProcessTSPProduct() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TSProductFile	The time series product file to process. 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.
RunMode	Indicate the run mode to process the product, one of: <ul style="list-style-type: none"> 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. 	None – must be specified.
View	Indicates whether the output should be previewed interactively, one of: <ul style="list-style-type: none"> True – display the graph. False – do not display the graph (specify the output file instead to automate image creation). 	None – must be specified.
OutputFile	The absolute or relative path to an output file. Use this parameter with 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).	Graph file will not be created.
DefaultSaveFile	Used with experimental feature to enabling editing in the time series table that corresponds to a graph view. Specify the default DateValue filename to save edits.	Editing is disabled.

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
ProcessTSPProduct(TSPProductFile="Example_ProcessTSPProduct.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 TSPProduct** 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: ReadDateValue()

Read all time series from a DateValue File

Version 09.07.02, 2010-08-20

The `ReadDateValue()` command reads all the time series in a DateValue file into memory (see the **DateValue Input Type Appendix**).

The following dialog is used to edit the command and illustrates the syntax for the command. 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 read (if a relative path is desired, delete the leading path after the select). Use the `TS Alias = ReadDateValue()` command to read a single time series from a DateValue file. Note that reading a DateValue file that was created for time series that have aliases will result in the aliases being assigned to the result.

Edit ReadDateValue() Command

Read all the time series from a DateValue file, using information in the file to assign the identifier and alias.
Specify a full path or relative path (relative to the working directory) for a DateValue file to read.
The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\ReadDateValue
Specifying the input period will limit data that are available for fill commands but can increase performance.

DateValue file to read:

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.

Command:

```
ReadDateValue ( InputFile="Data\08251500.DWR.Streamflow.IRREGULAR.dv" )
```

ReadDateValue

ReadDateValue() Command Editor

The command syntax is as follows:

```
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 <code>\${PropertyName}</code> (see also the <code>SetProperty()</code> command).	None – must be specified.
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: TS Alias = ReadDateValue()

Read a single time series from a DateValue File

Version 08.16.04, 2008-09-24

The `TS Alias = ReadDateValue()` command reads a single time series from a DateValue file (see the **DateValue Input Type Appendix**) and assigns an alias to the result. This command should not be confused with the `ReadDateValue()` command that does not use the alias, which reads all time series in a DateValue file. Currently the file being read **must contain only one time series**.

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

Edit TS Alias = ReadDateValue() Command

Read a single time series from a DateValue format file and assign an alias to the time series.
Specify a full path or relative path (relative to working directory) for a DateValue file to read.
The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\ReadDateValue_Alias
Specifying the input period will limit data that are available for fill commands but can increase performance.

Time series alias:

DateValue file to read:

Identifier/Alias to Read: Pattern to match a time series to read (under development).

Units to convert to:

Period to read: to

Command:

```
TS TS_ARMA = ReadDateValue(InputFile="TS_ARMA.out",TSID="*")
```

ReadDateValue_Alias

TS Alias = ReadDateValue() Command Editor

The command syntax is as follows:

```
TS Alias = ReadDateValue (Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
Alias	Alias for the new time series that is read from the file, which can be used instead of the TSID in other commands.	None – must be specified.
InputFile	The name of the DateValue input file to read, surrounded by double quotes. The path to the file can be absolute or relative to the working directory.	None – must be specified.
TSID	A time series identifier pattern to filter the read – this parameter is currently not used. Therefore the DateValue file should contain only one time series.	Currently ignored.
NewUnits	The new units for the time series. The data values will be converted to these units.	Use the units read from the file.
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.

A sample command file is as follows:

```
TS TS_ARMA = ReadDateValue (InputFile="TS_ARMA.out")
```

Command Reference: ReadDelimitedFile()

Read time series from a delimited file

Version 09.06.04, 2010-05-25

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 three main 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 embedded in 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.

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 below.

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
<code>InputFile</code>	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 <code>\${PropertyName}</code> (see also the <code>SetProperty()</code> command).	None – must be specified.
<code>Delimiter</code>	The delimiter character(s) that separate columns.	None – must be specified.
<code>TreatConsecutiveDelimitersAsOne</code>	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 character)
<code>Comment</code>	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 numbers and ranges a-b, separated by commas. Rows are skipped prior to other processing.	No rows will be skipped.
SkipRowsAfterComments	Indicate the number of rows to skip after header comments. Use this parameter to skip column headers prior to the data lines. This parameter is typically not used if column names are read from the file.	No rows will be skipped.
ColumnNames	The user-specified names for columns in the file, used to ensure that column headings in files are properly 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.	None – must be specified.
DateTimeColumn	The column matching a value in ColumnNames, which indicates the date/time column in the file.	None – must be specified.
DateTimeFormat	The format for date/time strings in the date/time column.	Under development – the format is automatically determined in most cases.
DateColumn	The column matching a string in ColumnNames, which indicates the date column in the file.	Under development.
TimeColumn	The column matching a string in ColumnNames, which indicates the time column in the file.	Under development.
ValueColumn	The column(s) matching a string in ColumnNames, which indicate the data value columns. Separate column names with commas. The FC[start:stop] notation discussed for ColumnNames can also be used.	None – must be specified.
LocationID	The location identifier(s) to assign to time series for each of the value columns (or specify one value to apply to all columns). The FC[start:stop] notation discussed for ColumnNames can also be used.	None – must be 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.

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:

```
#----- Provisional Data -----
#This system is maintained by the Colorado Division of Water Resources.
#Contact: Colorado Division of Water Resources (303) 866-3581
#
#All data presented on the Colorado Surface Water Conditions web site are
#provisional and subject to revision. Data users are cautioned to consider
#carefully the provisional nature of the information before using it for
#decisions that concern personal or public safety or the conduct of business
#that involves substantial monetary or operational consequences.
#
#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/Time      GAGE_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. Note that the column headings are skipped because they are assigned with a command parameter.

Edit ReadDelimitedFile() Command

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 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:stop] 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. Specify a full path or relative path (relative to working directory) for a delimited file to read. The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ReadDelimitedFile

Delimited file to read:

Delimiter: Required - delimiter character (use \t for tab).
Optional (default=False).

Treat consecutive delimiters as one?: Optional - character(s) that indicate comment lines (default=#).

Comment character(s): Optional - comma-separated numbers (1+) and ranges (e.g., 1,3-7) (default=none).

Rows to skip (by row number): Optional - number of rows to skip after header comments (default=0).

Rows to skip (after header comments): Optional - number of rows to skip after header comments (default=0).

Column name(s): Required - column names for file, used below to read data.

Date/time column: Required - if date and time are in the same column.

Date/time format: Optional - date/time format MM/DD/YYYY, etc. (under development).

Date column: Required - if date and time are in separate columns (under development).

Time column: Required - if date and time are in separate columns (under development).

Value column(s): Required - specify column names for time series values, separated by commas.

Location ID(s): Required - location ID for each value column, separated by commas.

Data provider: Optional - data 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: 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 file data (default=blank values).

Alias to assign: Insert: Optional - use %L for location, etc. (default=no alias).

Command:

```
ReadDelimitedFile (InputFile="Data\CO-DWR-ADATUNCO-tab.txt", Delimiter="\t", ColumnNames="ID,DateTime,GAGE_HT,DISCHRG", DateTimeColumn="DateTime", ValueColumn="GAGE_HT,DISCHRG", SkipRows="1,4-6,21-25", SkipRowsAfterComments="1", LocationID="ADATUNCO", Provider="DWR", DataType="GAGE_HT,DISCHRG", Interval=15Minute, Units="ft,cfs", Alias="%L%T")
```

ReadDelimitedFile

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\ 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() Command

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 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:stop] 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.
 Specify a full path or relative path (relative to working directory) for a delimited file to read.
 The working directory is: K:\PROJECTS\1091_CWCB_Basin Needs DSS\Tasks\Task02_DataLoading\Population

Delimited file to read:

Delimiter: Required - delimiter character (use \t for tab).

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): Required - column names for file, used below to read data.

Date/time column: Required - if date and time are in the same column.

Date/time format: Optional - date/time format MM/DD/YYYY, etc. (under development).

Date column: Required - if date and time are in separate columns (under development).

Time column: Required - if date and time are in separate columns (under development).

Value column(s): Required - specify column names for time series values, separated by commas.

Location ID(s): Required - location ID for each value column, separated by commas.

Data provider: Optional - data 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: 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 file data (default=blank values).

Alias to assign: Insert: Optional - use %L for location, etc. (default=no alias).

Command:

```
ReadDelimitedFile (InputFile="DOLA-counties1yr-trans.csv", Delimiter=",", ColumnNames="Year,
FC[2:]", DateTimeColumn="Year", ValueColumn="FC[2:]", LocationID="FC[2:]", Provider="DOLA", Da
taType="Population", Interval=Year, Units="Persons", Alias="%L-pop")
```

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.

```
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-countieslyr.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

Version 09.03.04, 2009-04-22

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 ReadHecDss() 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 parts, specify the DSS pathname to read a specific time series (the path will be used before the parts).

The alias can be assigned for time series based on time series properties, for example use %L for location (A-part:B-part), %T for data type (C-part), %Z for scenario (F-part).

Specify a full path or relative path (relative to working directory) for a HEC-DSS file to read.

The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ReadHecDss

Specifying the input period will limit data that are available for fill commands but can increase performance.

HEC-DSS file to read:

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).

Period to read: to

TSID location to assign: Optional - use %A for A-part, etc. (default=%A:%B).

Alias to assign: Optional - use %L for location, etc. (default=no alias).

Command:

ReadHecDss

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, surrounded by double quotes to protect whitespace and special characters.	None – must be specified.
A	The A part (basin name) to match, using * as a wildcard. The location part of the TSTool time series identifier is set to A:B.	Match all.
B	The B part (location) to match, using * as a wildcard. The location part of the TSTool time series identifier is set to A:B.	Match all.
C	The C part (parameter) to match, using * as a wildcard. The TSTool data type is set to this value.	Match all.
E	The E part (interval) to match, using * as a wildcard.	Match all.
F	The F part (scenario) to match, using * as a wildcard.	Match all.
Pathname	The HEC-DSS pathname for a time series, as specified in the HEC-DSS documentation. Currently wildcards are not allowed. If specified, this will be used instead of the A-F parameters.	Use the A-F parameters.
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.
Location	The location to assign for the time series identifier. Use %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.	Apart:Bpart (%A : %B).
Alias	Alias to assign to the output time series. See the LegendFormat property described in the TSView Time Series Viewing Tools appendix. For example, %L is full location, %T is data type (parameter in HEC-DSS notation), %I is interval, and %Z is scenario.	None is assigned. However, if the location contains periods that are in conflict with time series identifier conventions, the 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 09.07.02, 2010-08-20

The `ReadHydroBase()` command reads one or more time series from the HydroBase database (see the **HydroBase Input Type Appendix**). It is designed to utilize query criteria to process large numbers of time series.

The following special actions occur, depending on data type:

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 (Nov to Oct). HydroBase only stores full months of data when non-zero observations or non-zero filled values occur in a month. Therefore, this filling action should only provide additional zero values. Irrigation years with no observations remain as missing after the read. See the `FillHistMonthAverage()` command, which is often used to fill completely missing years.
2. Daily, monthly, and yearly diversion and reservoir release time series can optionally be filled using diversion comments, which indicate when irrigation years should be treated as missing. See the `FillUsingDivComments` parameter below. Note that diversion comments should not conflict with more detailed records but and provide additional information. The older `FillUsingDivComments()` command is also available for filling.

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

Edit ReadHydroBase Command

Read one or more time series from a HydroBase database.
 The data type and interval must be specified. Constrain the query using the "where" clauses, if necessary.
"Where" choices have only been fully implemented for structure time series (e.g., DivTotal, DivClass, RelTotal, RelClass).
 Refer to the HydroBase Input Type documentation for possible values.
 Specifying the period will limit data that are available for fill commands but can increase performance.
 Filling with diversion comments applies only to diversion and reservoir records.

Data type: Required (e.g., Streamflow, DivTotal).
 Data interval: Required (e.g., Day, Month, Year).
 Input name: Optional - HydroBase connection name (blank for default).

Where:
 Where:
 Where:
 Where:

Input start: Optional - overrides the global input start.
 Input end: Optional - overrides the global input end.

Fill using diversion comments: Optional - may result in 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).

Command:

```
ReadHydroBase (DataType="DivClass", Interval="Day", Where1="District;Equals;3", Where2="Structure ID;Equals;905", Where3="SFUT;Contains;s:2", Where4="SFUT;Contains;u:Q", FillUsingDivComments=True)
```

ReadHydroBase

ReadHydroBase() 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 `ReadHydroBase()` command reads the time series list and the corresponding data for the time series. This can greatly shorten commands files and simplify command logic, especially when processing large amounts of data.

Currently the **Data type** and **Data interval** must be entered manually (drop-down choices are not available), according to the **HydroBase Input Type Appendix**. Currently, only the structure data types (in particular diversions) are supported in the above dialog and have been tested. Support for other data types will be added as resources allow.

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
DataType	The data type to be queried, as documented in the HydroBase Input Type Appendix . The following conditions apply: <ul style="list-style-type: none"> For diversions, use <code>DivClass</code> without the SFUT sub-type. The SFUT sub-type will be added after data are queried. For reservoir releases, use <code>RelClass</code> without the SFUT sub-type. The SFUT sub-type will be added after data are queried. 	None – must be specified.
Interval	The data interval for the time series, as documented in the HydroBase Input Type Appendix (e.g. Day, Month, Year).	None – must be specified.
InputName	The HydroBase database connection input name to use for the connection, as initialized in <code>OpenHydroBase()</code> , which allows reading from more than one HydroBase in the same commands file.	Use the default HydroBase connection.
WhereN	The “where” clauses to be applied when querying data, matching the values in the Where fields in the command editor dialog and the TSTool main interface. The parameters should be named <code>Where1</code> , <code>Where2</code> , etc., with a gap resulting in the remaining items being ignored. The format of each value is: <p>“Item;Operator;Value”</p> <p>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.</p> <p>Warning: Currently the <code>>=</code> and <code><=</code> operators will produce errors – this issue is being evaluated. Work around by using the <code>Less Than</code> and <code>Greater Than</code> operators with appropriate Value.</p>	If not specified, the query will not be limited and very large numbers of time series may be queried.
InputStart	Start of the period to query, specified as a date/time with a precision that matches the requested data interval.	Read all available data.
InputEnd	End of the period to query, specified as a date/time with a precision that matches the requested data interval.	Read all available data.
FillUsingDivComments	Indicate whether to fill diversion and reservoir release time series using diversion comments.	False

Parameter	Description	Default
FillUsingDivCommentsFlag	If specified as a single character, data 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.	No flag is assigned.
IfMissing	Indicate the action to be taken if the requested time series is missing, one of: <ul style="list-style-type: none">• Ignore – ignore the time series (do not warn and the time series will not be available)• Warn – generate a failure for the command	Warn

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")
```


Command Reference: TS Alias = ReadHydroBase()

Read a single time series from a HydroBase Database

Version 08.16.03, 2008-08-20

The `TS Alias = ReadHydroBase()` command reads a single time series from a HydroBase database (see the **HydroBase Input Type Appendix**) and assigns an alias to the result. This command should not be confused with the `ReadHydroBase()` command that does not use the alias, which reads one or more matching time series from a HydroBase database.

The following special actions occur, depending on data type:

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 (Nov to Oct). HydroBase only stores full months of data when non-zero observations or non-zero filled values occur in a month. Therefore, this filling action should only provide additional zero values. Irrigation years with no observations remain as missing after the read. See the `FillHistMonthAverage()` command, which is often used to fill completely missing years.
2. Daily, monthly, and yearly diversion and reservoir release time series can optionally be filled using diversion comments, which indicate when irrigation years should be treated as missing. See the `FillUsingDivComments` parameter below. Note that diversion comments should not conflict with more detailed records but and provide additional information. The older `FillUsingDivComments()` command is also available for filling.

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

Edit TS Alias = ReadHydroBase Command

Read a single time series from the HydroBase database.
Refer to the HydroBase Input Type documentation for possible values.
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.
Filling with diversion comments applies only to diversion and reservoir release time series.

Time series alias:

Location: For example, station or structure ID.

Data source: For example: USGS, MWS.

Data type: For example: Streamflow.

Data interval: For example: 6Hour, Day, Month.

Input name: HydroBase connection name (blank for default).

TSID (full):

Period to read: to

Fill using diversion comments: Whether to use diversion comments to fill more zero values (blank=False).

Fill using diversion comments flag: 1-character flag to indicate filled diversion comment values.

If missing: How to handle missing time series (blank=VWarn).

Command:

```
TS NorthPoudreDiv =
ReadHydroBase(TSID="0300905999.DWR.DivTotal.Day~HydroBase",IfMissing=Ignore)
```

Cancel OK

ReadHydroBase_Alias

TS Alias = ReadHydroBase() Command Editor

The command syntax is as follows:

```
TS Alias = ReadHydroBase (Parameter=Value...)
```

Command Parameters

Parameter	Description	Default
Alias	Alias for the new time series that is read from the file.	None – must be specified.
TSID	A time series identifier to read – see the HydroBase Input Type Appendix .	Required – specify the time series identifier to read.
InputName	The HydroBase database connection input name to use for the connection, as initialized in <code>OpenHydroBase()</code> , which allows reading from more than one HydroBase in the same commands file.	Use the default HydroBase connection.
InputStart	The start of the period to read data – specify if the period should be different from the global input period.	Use the global input period.
InputEnd	The end of the period to read data – specify if the period should be different from the global input period.	Use the global input period.
FillUsingDivComments	Indicate whether to fill diversion and reservoir release time series using diversion comments.	False
FillUsingDivCommentsFlag	If specified as a single character, data 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.	No flag is assigned.
IfMissing	Indicate the action to be taken if the requested time series is missing, one of: <ul style="list-style-type: none"> Ignore – ignore the time series (do not warn and the time series will not be available) Warn – generate a failure for the command 	Warn

A sample command file to read a diversion time series is as follows:

```
TS NorthPoudreDiv = ReadHydroBase(TSID="0300905.DWR.DivTotal.Day~HydroBase")
```

This page is intentionally blank.

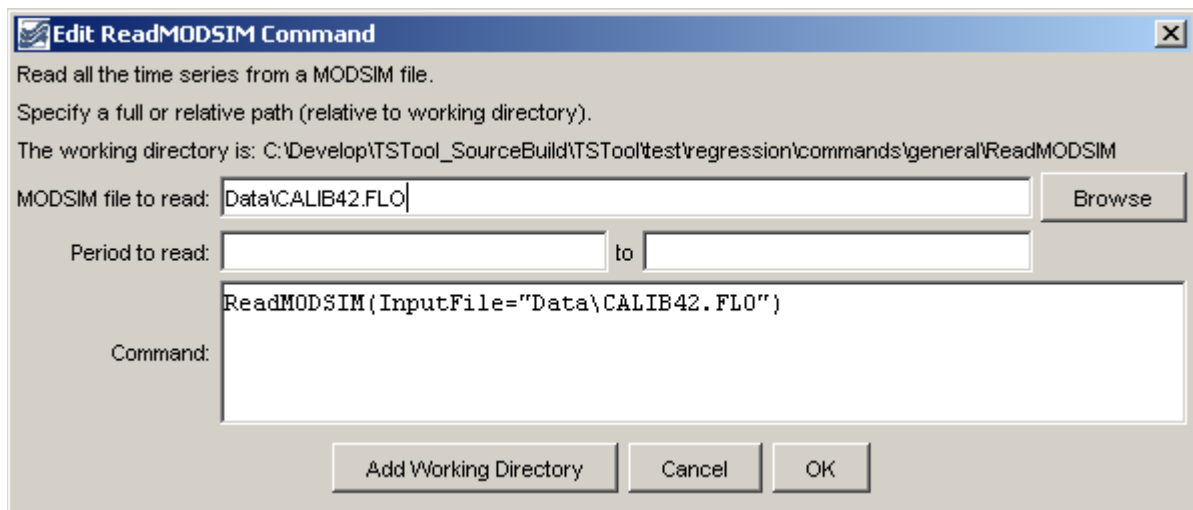
Command Reference: ReadMODSIM()

Read all time series from a MODSIM output file

Version 08.16.04, 2008-09-09

The ReadMODSIM() command reads all the time series in a MODSIM output file (see the **MODSIM Input Type Appendix**). The actual reading occurs as the commands are being processed.

The following dialog is used to edit the command and illustrates the syntax for the command. Use the TS Alias = ReadMODSIM() command to read a single time series from a MODSIM file.



ReadMODSIM

ReadMODSIM() Command Editor

The command syntax is as follows:

ReadMODSIM(Parameter=Value...)

Command Parameters

Parameter	Description	Default
InputFile	The name of the MODSIM file to read, surrounded by double quotes. The path to the file can be absolute or relative to the working directory.	None – must be specified.
InputStart	The start of the period to read data, specified to the precision of the dates used with data.	Use the global query period.
InputEnd	The end of the period to read data, specified to the precision of the dates used with data.	Use the global query period.

A sample command file is as follows:

```
ReadMODSIM( InputFile= "Data\CALIB42.FLO" )
```

Command Reference: TS Alias = ReadMODSIM()

Read a single time series from a MODSIM output file

Version 08.16.04, 2008-09-09

The `TS Alias = ReadMODSIM()` command reads a single time series from a MODSIM file (see the **MODSIM Input Type Appendix**) and assigns an alias to the result. This command should not be confused with the `ReadMODSIM()` command that does not use the alias, which reads all time series in a MODSIM file.

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).

Edit TS Alias = ReadMODSIM() Command

Read a single time series from a MODSIM format file.
The 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\TestCases\CommandReference\ReadMODSIM_Alias

Time series alias:

MODSIM file to read:

Node/link name to read:

Data type to read:

TSID (full):

Period to read: to

Command:

```
TS BIGTOM = ReadMODSIM(InputFile="Data\BIGTOM17.RES",TSID="BIGTOM..STOR_TRG..")
```

ReadMODSIM_Alias

TX Alias = ReadMODSIM() Command Editor

The command syntax is as follows:

```
TS Alias = ReadMODSIM(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
Alias	Alias for the new time series that is read from the file, which can be used instead of the TSID in other commands.	None – must be specified.
InputFile	The name of the MODSIM file to read, surrounded by double quotes. The path to the file can be absolute or relative to the working directory.	None – must be specified.
TSID	A time series identifier pattern to filter the read – this is constructed in the editor dialog from individual identifier parts – the location and data type are specified and used in the time series identifier.	None – must be specified to match a single time series.
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.

A sample command file is as follows:

```
TS Alias =
ReadMODSIM( InputFile="BIGTOM17.RES",TSID="GREELEYCBT..STOR_TRG.." )
```


Command Reference: ReadPatternFile()

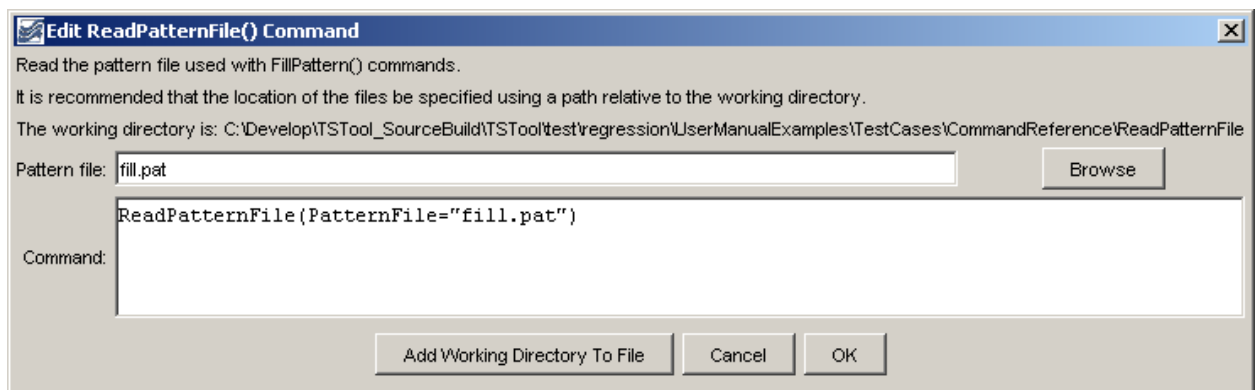
Read the pattern file to be used with FillPattern() commands

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 = 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.
# -e-b-----eb-----eb-----eb-----eb-----eb-----eb-----eb-----eb-----eb-----e
10/1908 -          9/1996 ACFT WYR
1909 09034500      AVG      AVG      AVG      WET      WET      WET      AVG      AVG      WET      WET      WET      WET
1910 09034500      WET      WET      WET      WET      WET      WET      WET      AVG      AVG      AVG      AVG      WET
1911 09034500      AVG      AVG      WET      AVG      AVG      AVG      AVG      WET      WET      WET      AVG      WET
1912 09034500      WET      WET      WET      WET      WET      WET      AVG      WET      WET      WET      WET      WET
...omitted...
```

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



ReadPatternFile

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: TS Alias = ReadRiverWare()

Read a single time series from a RiverWare file

Version 08.16.04, 2008-09-09

The `TS Alias = 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 TS Alias = ReadRiverWare() Command

Read a single time series from a RiverWare time series file.
It is assumed that the filename follows the convention ObjectName.SlotName.
The ObjectName and SlotName will be used for the time series identifier location and data type, respectively.
Specify a full path or relative path (relative to working directory) for a RiverWare file to read.
Specifying the period will limit data that are available for fill commands but can increase performance.
Specifying units causes conversion during the read (new units must be understood by TSTool). See also Scale() and ConvertDataUnits().
If not specified, the period defaults to the global input period.
The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\ReadRiverWare_Alias

Time series alias:

RiverWare file to read:

Units to convert to:

Period to read: to

Command:

```
TS ts1 = ReadRiverWare(InputFile="SouthHolstonData.SOGPoolElevation")
```

ReadRiverWare_Alias

TS Alias = ReadRiverWare() Command Editor

The command syntax is as follows:

```
TS Alias = ReadRiverWare(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
Alias	Alias for the new time series that is read from the file, which can be used instead of the TSID in other commands.	None – must be specified.
InputFile	The name of the RiverWare file to read, surrounded by double quotes. The path to the file can be absolute or relative to the working directory.	None – must be specified.
Units	The units for the time series. The data values will be converted to these units. TSTool by default understands certain 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.	Use the units read from the file.
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.

A sample command file is as follows:

```
TS ts1 = ReadRiverWare(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 ReadStateCU() Command

Read 1+ (or all) time series from one of the following file types, using data in the file to assign the identifier:

- Crop pattern time series file (StateCU input file).
- Irrigation practice time series file (input).
- StateCU IWR or WSL report file (output).

Specify a full or relative path (relative to working directory).

The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\ReadStateCU

The time series identifier pattern, if specified, will filter the read ONLY for IWR and WSL files;

specify blank to read all or use * wildcards to match a time series identifier.

For example, to read all monthly IWR time series for locations starting with ABC, specify:

ABC.*.IWR.Month

Location can be X, X*, or *. Data type and interval can be * or combinations as follows:

- CropArea-AllCrops (Year)
- IWR or WSL (Month, Year)
- IWR_Depth or WSL_Depth (Month, Year)

StateCU file to read:

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=none).

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).

Command:

```
ReadStateCU( InputFile="Data\ ym2004. iwr")
```

ReadStateCU

ReadStateCU() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
InputFile	The name of the StateCU time series or report file to read, surrounded by double quotes.	None – must be specified.
InputStart	The starting date/time to read, specified to a precision (month or year) that matches the data file.	Read all the data.
InputEnd	The ending date/time to read, specified to a precision (month or year) that matches the data file.	Read all the data.
TSID	A time series identifier pattern that will be used to filter the list of time series that are read. See the figure above for examples.	Read all time series.
NewScenario	A new scenario to use for the TSID. This is useful when reading data from multiple model runs that otherwise would have the same TSIDs.	No scenario.
AutoAdjust	Indicate whether to automatically adjust 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.	True
CheckData	Indicate whether to check the data for 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.	True

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() Command

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 time series in StateCU binary files, the list of time series identifiers in the file will NOT be available in other command editors.
Specify a full or relative path (relative to working directory).
The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ReadStateCUB
The time series identifier pattern, if specified, will filter the read.
Use blank or * to read all time series.
Use A* to read all time series with alias or location starting with A.
Use *.XXXXX.* to read all time series with data type XXXXX.
Currently, data source, interval, and scenario are internally defaulted to *.

StateCU binary file to read:

Time series ID: Optional - specify a TSID pattern to match.

Input start: Optional - default is global input start or all data.

Input end: Optional - default is global input end or all data.

Command:

```
ReadStateCUB (InputFile="Data\SP2008_OutOfOrder.BD1", InputStart="1970-01", InputEnd="1971-12")
```

ReadStateCUB

ReadStateCUB() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
InputFile	The name of the StateCU binary time series file to read, surrounded by double quotes. The path to the file can be absolute or relative to the working directory.	None – must be specified.
TSID	Time series identifier pattern to filter the read.	Read all time series.
InputStart	The starting date/time to read data, specified to Month precision.	Read all data.
InputEnd	The ending date/time to read data, specified to Month precision.	Read all data.

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:

Input start: Optional - overrides the global input start.

Input end: Optional - overrides the global input end.

Alias to assign: Insert: 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).

Command:

```
ReadStateMod (InputFile="ym2004.ddh")
```

ReadStateMod

ReadStateMod() Command Editor

The command syntax is as follows:

```
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 <code>\${PropertyName}</code> (see also the <code>SetProperty()</code> 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: <ul style="list-style-type: none"> 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 StateMod binary output file, using information in the file to assign the identifier.
Specify a full or relative path (relative to working directory).
The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ReadStateModB
The time series identifier pattern, if specified, will filter the read.
Use blank or * to read all time series.
Use A* to read all time series location starting with A.
Use *,*,XXXXX,*,* to read all time series with data type XXXXX.
Currently, data source, interval, and scenario are internally defaulted to *.

StateMod binary file to read:

Time series ID: 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: Insert: Optional - use %L for location, etc. (default=no alias).

Command:

```
ReadStateModB ( InputFile="Data/ex7_Version12.29.b43", TSID="*,*,Available_Flow.*,*", Alias="%L-%T" )
```

ReadStateModB() Command Editor

ReadStateModB

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
InputFile	The name of the StateMod binary 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 <code>\${PropertyName}</code> (see also the <code>SetProperty()</code> command).	None – must be specified.
TSID	Time series identifier pattern to filter the read. Use periods to indicate separate TSID parts and use <code>*</code> to match patterns within the parts.	Read all time series.
InputStart	The starting date/time to read data, specified to Day or Month precision based on whether a daily or monthly model run.	Read all data.
InputEnd	The ending date/time to read data, specified to Day or Month precision based on whether a daily or monthly model run.	Read all data.
Version	StateMod version number using the form NN.NN (padded with leading zero for version 9) 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 for version 11+ is automatically detected.	Detect from the file if possible.
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.

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: 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: Required - unique identifier for the table.

Input file:

Command:

```
ReadTableFromDBF (TableID="ClimateStations", InputFile="Data\div1_climatestations.dbf")
```

ReadTableFromDBF

ReadTableFromDBF() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TableID	Identifier to assign to the table that is read, which allows the table data to be used with other commands.	None – must be specified.
InputFile	The name of the file to read, as an absolute path or relative to the command file location.	None – must be specified.

Command Reference: ReadTableFromDelimitedFile()

Read a table from a delimited file

Version 09.04.00, 2009-06-16

The `ReadTableFromDelimitedFile()` command reads a table from a comma-delimited file. Table files have the following characteristics:

- Comments indicated by lines starting with # are stripped during the read.
- Extraneous non-data 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.
- Once read, row numbers (1+) can be referenced by other commands.

Tables are used by other commands when performing lookups of information or generating summary information from processing.

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

Edit ReadTableFromDelimitedFile() Command

This command reads a table from a delimited file. The table can then be used by other commands.
Columns in the file should be delimited by commas (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):

```
1 | # This is a comment
2 | # This is another comment
3 | # Double-quoted fields in the 1st non-comment line will be treated as headers (see also HeaderLines)
4 | "Header1", "Header2", "Header3"
5 | 1 | 1.0, 1.5
6 | 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 | 3.0, 4.5
```

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

Table ID: Required - unique identifier for the table.

Input file:

File lines to skip: 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" )
```

ReadTableFromDelimitedFile

ReadTableFromDelimitedFile() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TableID	Identifier to assign to the table that is read, which allows the table data to be used with other commands.	None – must be specified.
InputFile	The name of the file to read, as an absolute path or relative to the command file location.	None – must be specified.
SkipLines	Indicates the number of lines in the file to skip, which otherwise would interfere with reading row data. Individual row numbers and ranges can be specified, for example: 1, 5-6, 17	No lines are skipped.
HeaderLines	Indicate the rows that include header information, which should be used for column names. Currently this should only be one row, although a range may be fully supported in the future.	If the first non-comment line contains quoted field names, they are assumed to be headers. Otherwise, no headers are read.

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

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

An excerpt from a simple delimited file is:

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

Command Reference: TS Alias = ReadTimeSeries()

Read a single time series using a full time series identifier

Version 08.16.04, 2008-09-23

The `TS Alias = ReadTimeSeries()` reads a single time series from any input type. 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.

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

Edit TS Alias = ReadTimeSeries() Command

This command is a general time series read command.
Its main purpose is to assign an alias to a time series, which is more convenient to use than the long time series identifier.
Read commands for specific input types generally offer more options and should be used if available.
The alias should be descriptive and should not contain spaces, periods, or parentheses.
See also the `CreateFromList()` command.

Time series alias:

Time series identifier:

If time series not found?: Required - how to handle time series that are not found.

Default units: Optional - units when `IfNotFound=Default`.

Command:

```
TS Alamosa =  
ReadTimeSeries(TSID="08235350.USGS.Streamflow.Day~HydroBase", IfNotFo  
und=Warn)
```

ReadTimeSeries_Alias

ReadTimeSeries() Command Editor

The command syntax is as follows:

```
TS Alias = ReadTimeSeries(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
Alias	The alias assigned to the time series.	None – must be specified.
TSID	The time series identifier of the time series to read. The identifier should include the input type (and input name, if required). See the input type appendices for examples of time series identifiers for various input types.	None – must be specified.
IfNotFound	Indicates how to handle missing time series, one of: <ul style="list-style-type: none"> 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 read data from the State of Colorado's HydroBase is as follows:

```
TS Alamosa =
  ReadTimeSeries(TSID="08235350.USGS.Streamflow.Day~HydroBase")
```

Command Reference: TS Alias = ReadUsgsNwis()

Read a single time series from a USGS NWIS file

Version 08.16.04, 2008-09-05

The `TS Alias = ReadUsgsNwis()` command reads a single time series from a USGS NWIS file (see the **USGSNWIS Input Type Appendix**) and assigns an alias to the result. 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`.

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

Edit TS Alias = ReadUsgsNwis Command

Read a single time series from a USGS NWIS format file. Currently only daily streamflow is supported.
Specify a full path or relative path (relative to working directory) for a USGS NWIS 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 global input period, or read all.

The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\ReadUsgsNwis

Time series alias:

USGS NWIS file to read:

Period to read: to

Command:

```
TS nwis = ReadUsgsNwis(InputFile="Data\G03451500.txt")
```

ReadUsgsNwis

TS Alias = ReadUsgsNwis() Command Editor

The command syntax is as follows:

```
TS Alias = ReadUsgsNwis(Parameter=Value,...)
```

Command Parameters

Parameter	Description	Default
Alias	Alias for the new time series that is read from the file, which can be used instead of the TSID in other commands.	None – must be specified.
InputFile	The name of the USGS NWIS file to read, surrounded by double quotes. The path to the file can be absolute or relative to the working directory.	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.	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.

A sample command file is as follows:

```
TS tsl = ReadUsgsNwis(InputFile="G03451500.txt")
```

Command Reference: TS Alias = RelativeDiff()

Create a relative difference time series

Version 08.16.04, 2008-09-23

A `RelativeDiff()` command can be inserted to create 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 TS Alias = RelativeDiff() Command

Create a new unitless time series with data values:

$$TS = (TS1 - TS2) / \text{Divisor}$$

where the Divisor is either TS1 or TS2.

Attempting to divide by zero sets the data point to the missing value.

The metadata for the result is a copy of TS1 except for the alias.

Time series alias:

Time series 1 (TS1):

Time series 2 (TS2):

Divisor:

Command:

```
TS RelativeDiff =  
RelativeDiff(TSID1="DelNorte",TSID2="Alamosa",Divi  
sor=DivideByTS1)
```

RelativeDiff_Alias

RelativeDiff() Command Editor

The command syntax is as follows:

```
TS Alias = RelativeDiff(Parameter=Value)
```

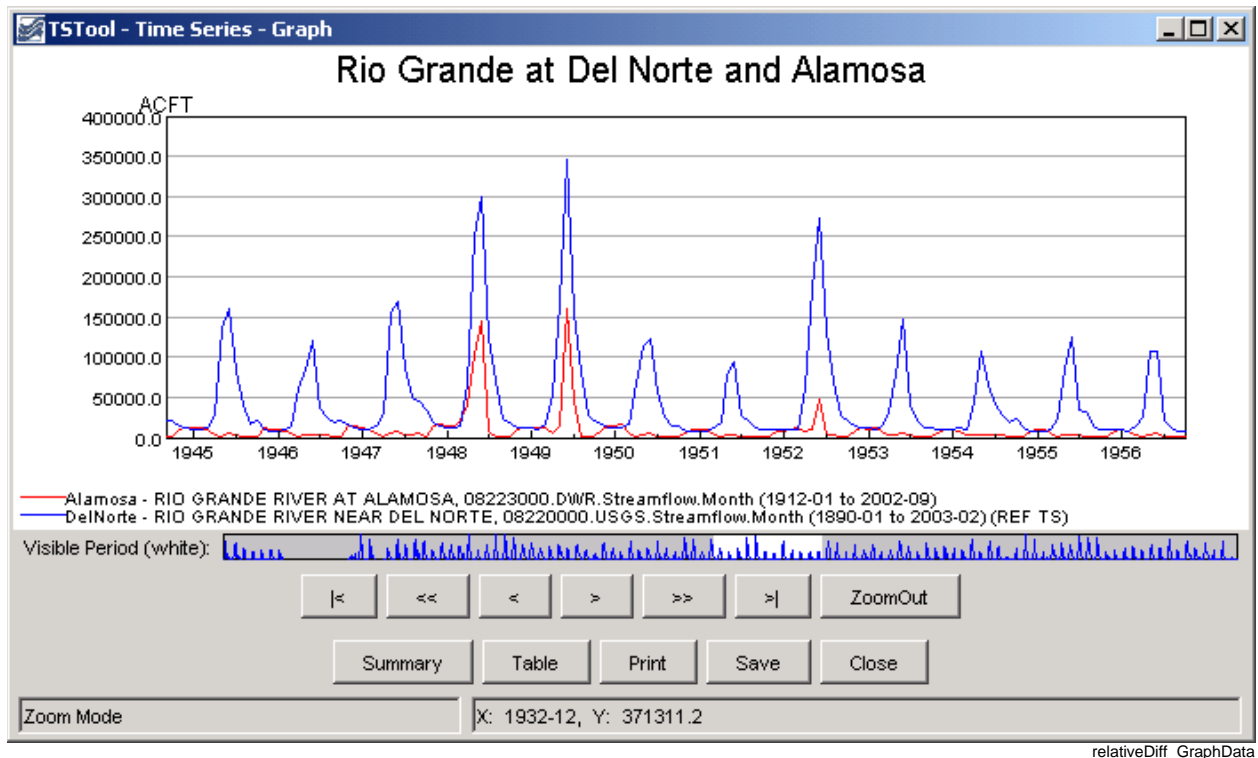
Command Parameters

Parameter	Description	Default
Alias	The alias for the new time series.	None – must be specified.
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 (subtracted from the first).	None – must be specified.
Divisor	Indicates whether the first time series is the divisor (DivideByTS1) or the second time series is the divisor (DivideByTS2).	None – must be specified.

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

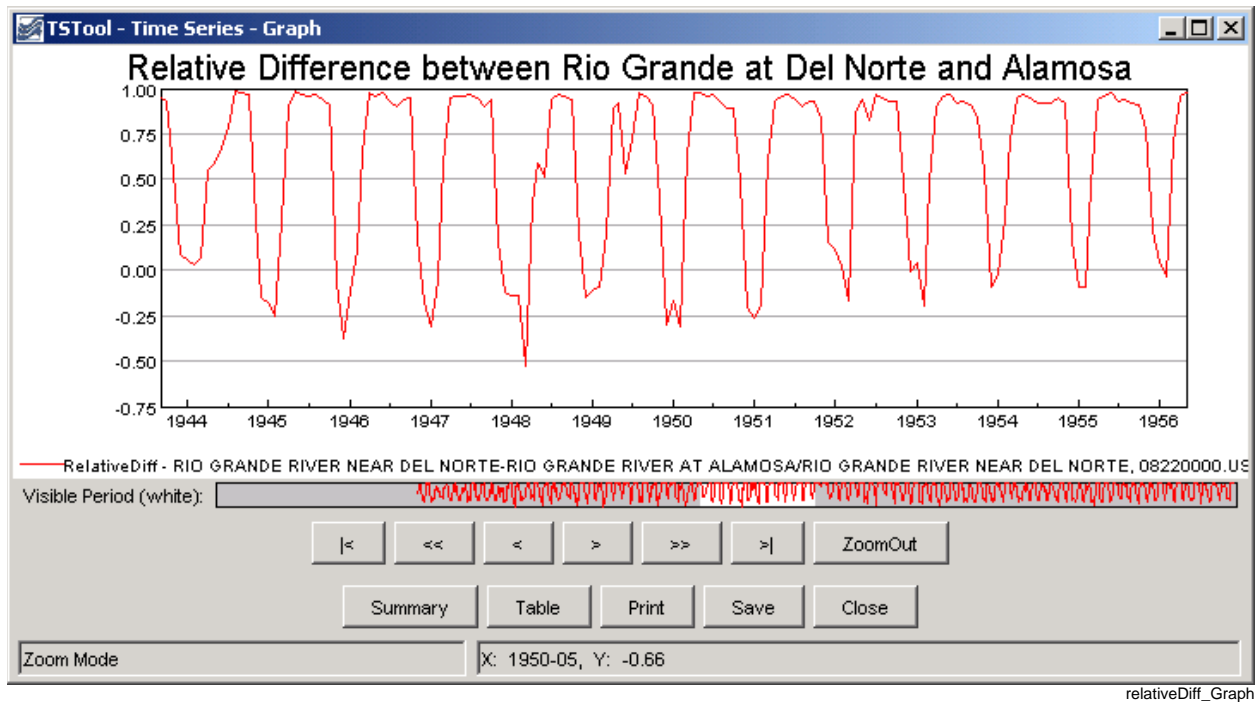
```
StartLog(LogFile="Example_RelativeDiff.log")
SetOutputPeriod(OutputStart="01/1912",OutputEnd="12/1998")
# (1912-1998) RIO GRANDE AT ALAMOSA, CO. DWR Streamflow Monthly
TS Alamosa = readTimeSeries("08223000.DWR.Streamflow.Month~HydroBase")
# (1890-1998) RIO GRANDE NEAR DEL NORTE, CO. DWR Streamflow Monthly
TS DelNorte = readTimeSeries("08220000.USGS.Streamflow.Month~HydroBase")
TS RelativeDiff =
  RelativeDiff(TSID1="DelNorte",TSID2="Alamosa",Divisor=DivideByTS1)
```

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

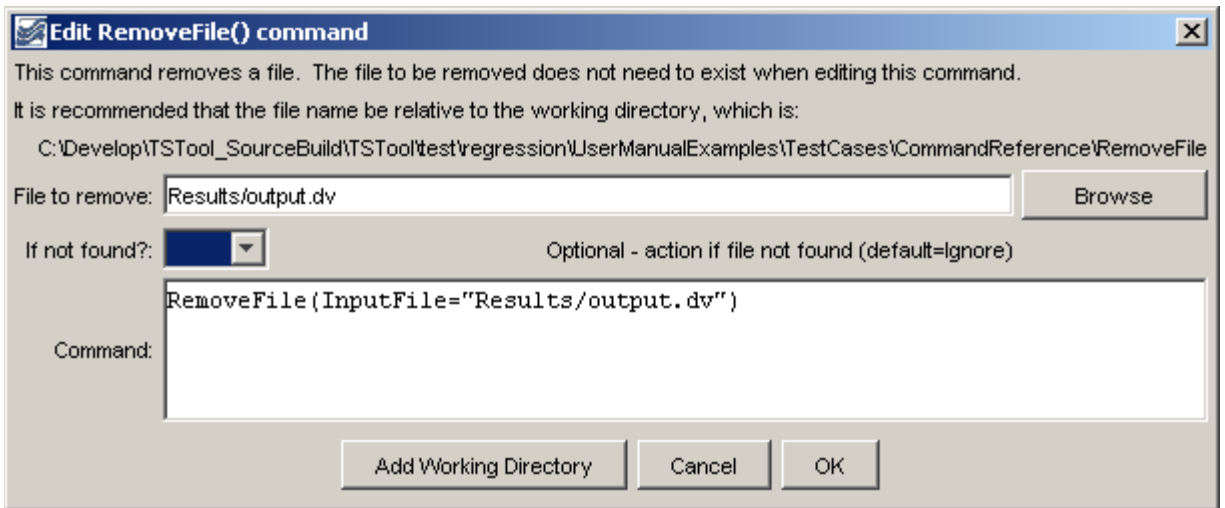
Version 09.02.00, 2009-04-03

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.



RemoveFile

RemoveFile() Command Editor

The command syntax is as follows:

```
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: <ul style="list-style-type: none">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).	Ignore

The following example command file illustrates how to remove a file:

```
RemoveFile (InputFile="Results/output.dv" )
```

Command Reference: ReplaceValue()

Replace a range of data values with a constant Value

Version 09.07.01, 2010-08-18

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. 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() Command

Replace a single data value or range of data values with a constant.
 Optionally, set missing, or remove the values entirely (if an irregular interval time series).
 If the missing value indicator is a number in the given range, missing values also will be replaced.
 Specify dates with precision appropriate for the data.

TS list: Optional - indicates the time series to process (default=AllTS).

TSID (for TSList=AllMatchingTSID):

EnsembleID (for TSList=EnsembleID):

Minimum value to replace: Required (maximum value can also be specified).
 Maximum value to replace: Optional - use when specifying range.
 Constant value to replace with: Required - or specify action.

Action: Optional - action for matched values (default=no action).

Replacement start: Optional - start of replacement (default is all).
 Replacement end: Optional - end of replacement (default is all).

☐ Analysis window: Optional - analysis window within each year (default=full year).
 Month: Day: Hour: Minute: Month: Day: Hour: Minute:

Command: `ReplaceValue (TSList=AllMatchingTSID, TSID="08235700.DWR.Streamflow.Month", MinValue=-100000, MaxValue=0, NewValue=0)`

ReplaceValue

ReplaceValue() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none"> 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. 	AllTS

	<ul style="list-style-type: none"> 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 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.
MinValue	The minimum value to replace.	None – must be specified.
MaxValue	The maximum value to replace.	If not specified, only data values that exactly match the minimum value will be replaced.
NewValue	The new data value.	Required, unless the Action parameter is specified.
Action	An action to take with values that are matched: <ul style="list-style-type: none"> Remove – remove the data points. This can only be specified for irregular interval time series and will be interpreted as SetMissing for regular interval time series. SetMissing – set values to missing. 	No action is taken and the NewValue parameter must be specified.
SetStart	The date/time to start filling, if other than the full time series period.	Check the full period.
SetEnd	The date/time to end filling, if other than the full time series period.	Check the full period.
AnalysisStart	The starting date/time within the calendar year to replace data. The window CANNOT cross calendar year boundaries (this may be allowed in the future). Use multiple commands if necessary.	Process each full year.
AnalysisEnd	The ending date/time within the calendar year to replace data.	Process each full year.

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)
```

Command Reference:

ResequenceTimeSeriesData()

Resequence time series data (shuffle years of data)

Version 09.05.02, 2009-11-04

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" 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 indicated year and have a time series identifier that is the same as the original time series, additionally with the indicated scenario.
 If not specified, the output start is taken from the global output start or the time series.

TS list: **AllTS** Optional - indicates the time series to process (default=AllTS).

TSID (for TSList=AllMatchingTSID):

EnsembleID (for TSList=EnsembleID):

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:

Last row number in table for year sequence:

Output year type: **Calendar** 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** Optional - use %L for location, etc. (default=no alias).

Command:
 ResequenceTimeSeriesData(TSList=AllTS, TableID="KNN_Seq", TableColumn="1", OutputStart="1908", NewScenario="KNN01", Alias="%L-KNN01")

Cancel OK

ResequenceTimeSeriesData

ResequenceTimeSeriesData() Command Editor

The command syntax is as follows:

ResequenceTimeSeriesData (Parameter=Value,...)

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none"> 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	Required when

Parameter	Description	Default
	ensemble.	TSList=EnsembleID.
TableID	The identifier for the sequence table to use, which indicates the dates to use when resequencing data (e.g., list of years for data sequence). For example, see the ReadTableFromDelimitedFile() command. The years should be consistent with the OutputYearType.	None – must be specified.
TableColumn	The column name containing the sequence information. Note that the input table must have column names in a header record.	None – must be specified.
TableRowStart	The first data row number (1+) containing the first year in the new sequence.	Use all rows.
TableRowEnd	The last data row number (1+) containing the first year in the new sequence.	Use all rows.
OutputYearType	The output year type, indicating the year extent for the resequencing, one of: <ul style="list-style-type: none"> • 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. 	Calendar
OutputStart	The output start as a four-digit year that is consistent with OutputYearType. For example, if processing water years, the OutputStart would be the first water year in the output (and start in October of the previous calendar year). The output end is relative to the output start and includes the number of years in the sequence.	Same as the original input data or use the global output start if specified. The output months will be adjusted for the output year type.
NewScenario	The new scenario to assign to the created time series, resulting in a unique TSID.	Not specified, but a new scenario and/or alias must be specified.
Alias	Alias to assign to the output time series. See the LegendFormat property described in the TSView Time Series Viewing Tools appendix. For example, %L is full location, %T is data type, %I is interval, and %Z is scenario.	Not specified, but a new scenario and/or alias must be specified.

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="Tracel",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
"Tracel","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

Version 09.00.00, 2010-09-30

The RunCommands () command runs a command file using a separate command 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.

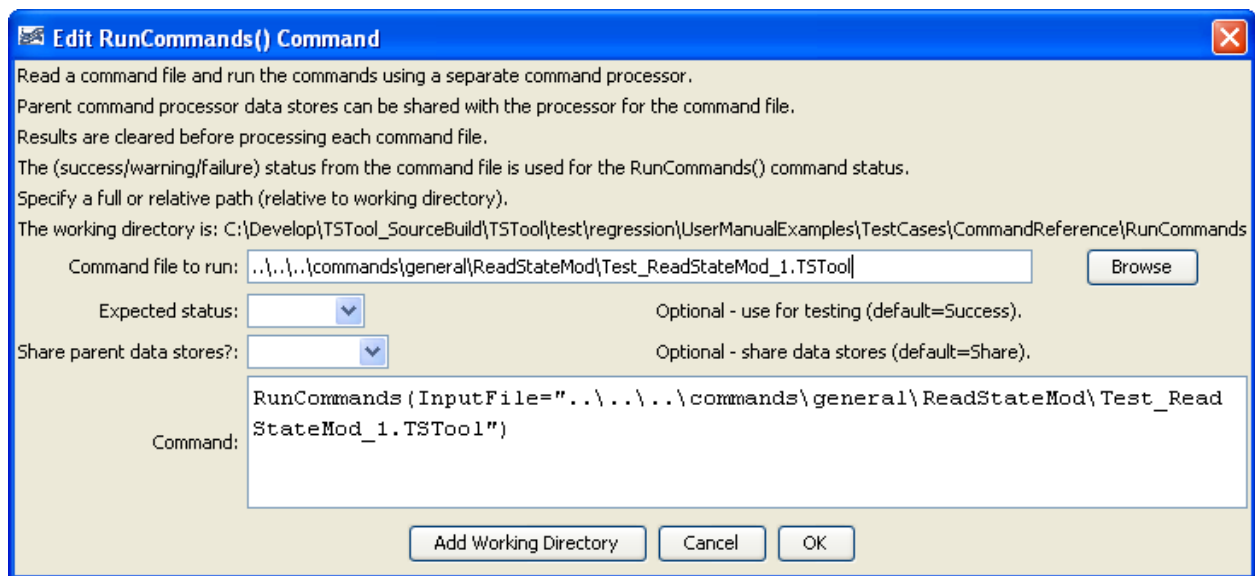
Command files that are run can themselves include RunCommands () commands. Each command file that is run has knowledge of its initial working directory and relative paths referenced in the command file are relative to this directory. This allows a master command file to reside in a different location 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.

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



RunCommands

RunCommands() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
InputFile	The name of the command file to run, enclosed in double quotes if the file contains spaces or other special characters. A path relative to the master command file can be specified.	None – must be specified.
ExpectedStatus	Used for testing – indicates the expected status from the command, one of: <ul style="list-style-type: none"> Unknown Success Warning Failure 	Success
ShareDataStores	Indicate whether data stores in the parent should be 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.	Share

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")
TS Alias = NewPatternTimeSeries(NewTSID="MyLoc..MyData.Day",
    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 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.

Command Reference: RunningAverage()

Convert time series data to running average values

Version 09.07.02, 2010-08-19

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. There are two versions of the command. The centered running average requires that the number intervals on each side of a point be specified (e.g., specifying 1 will average 3 values at each point). 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.

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

Edit RunningAverage() Command

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 one side of the current point, and optionally inclusive of the current point.
An N-year running average averages the values for the date/time and previous years (N years total).

TS list: Indicates the time series to process (default=AllTS).

TSID (for TSList=AllMatchingTSID):

EnsembleID (for TSList=EnsembleID):

Type of Average:

Number of intervals on each side:

Command: `RunningAverage (TSList=AllMatchingTSID,TSID="Center",AverageMethod=Centered,Bracket=3)`

RunningAverage_centered

RunningAverage() Command Editor for Centered Running Average

The following dialog illustrates the N-year running average command syntax.

Edit RunningAverage() Command

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 one side of the current point, and optionally inclusive of the current point.
An N-year running average averages the values for the date/time and previous years (N years total).

TS list: Optional - indicates the time series to process (default=AllTS).

TSID (for TSList=AllMatchingTSID):

EnsembleID (for TSList=EnsembleID):

Type of average: Required.

Number of years: Required.

Command:

RunningAverage_nyear

RunningAverage() Command Editor for N-Year Running Average

The command syntax is as follows:

RunningAverage (Parameter=Value, ...)

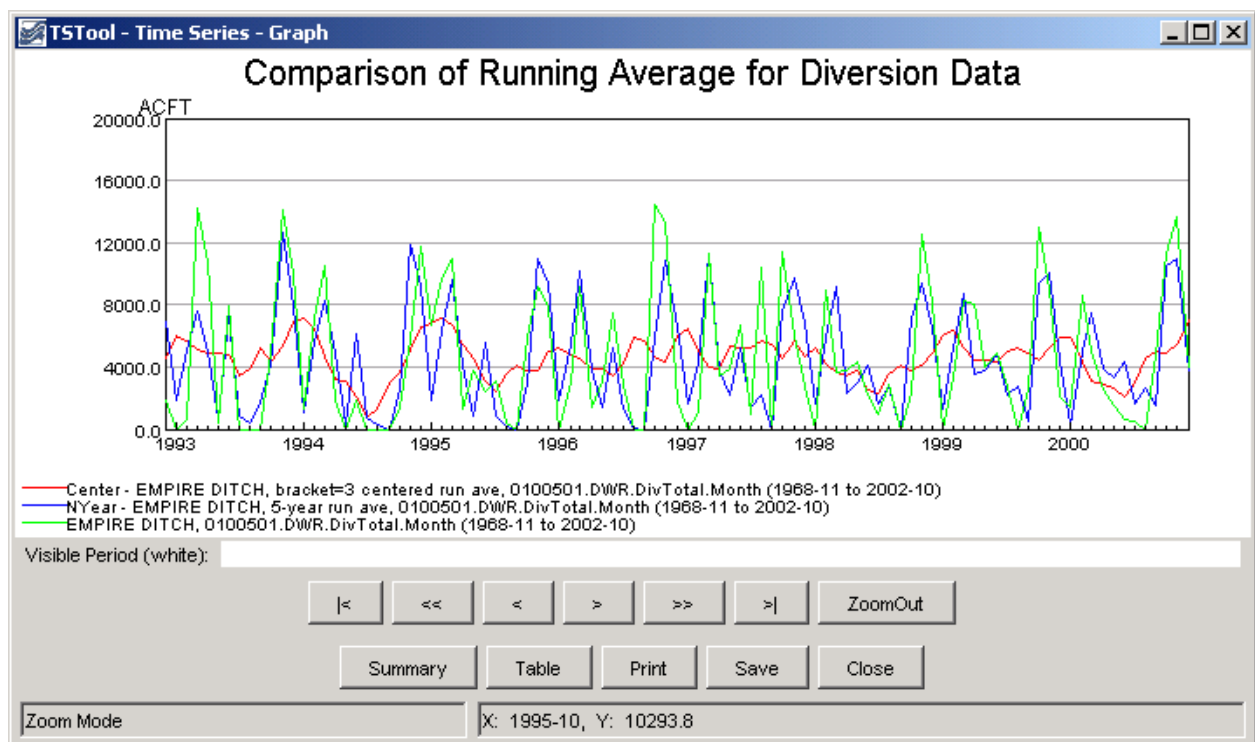
Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> 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. N-Year – values for the current year and (N – 1) preceding years, for the same date/time, are averaged. Previous – average the previous N (bracket) values but do not include the current value. PreviousInclusive – average the previous N (bracket) values and also 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
TS Center = readTimeSeries("0100501.DWR.DivTotal.Month~HydroBase")
RunningAverage(TSList=AllMatchingTSID,TSID="Center",
  AverageMethod=Centered,Bracket=3)
TS NYear = readTimeSeries("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:



RunningAverage_graph

Results from RunningAverage() Commands

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.

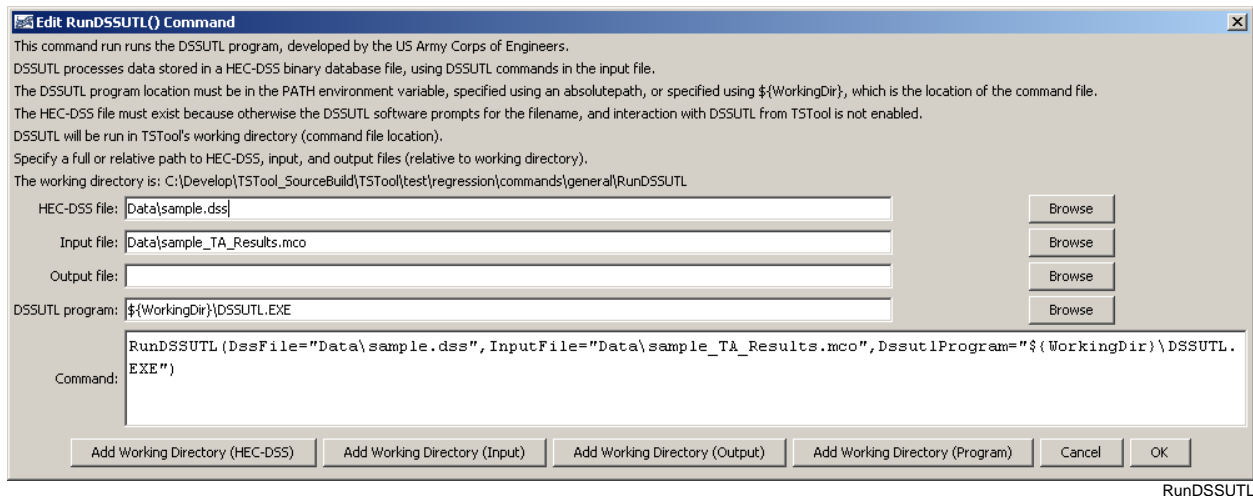
The following table summarizes how the command treats input for various utility programs. Required arguments are for the RunDSSUTL () command but may be optional if the program is run on the command line.

RunDSSUTL() Command Handling of HEC-DSS Utility Program Input

Program	Description	DSSFILE= Argument	INPUT= Argument	OUTPUT= Argument
DSSUTL	Data Storage System Utility Program	Required	Required	Optional
DSPLAY	Data Storage System Graphics Utility	Required	Required	Optional
DSSMATH	Utility Program for Mathematical Manipulation of HEC-DSS Data	Not used – use OPEN() command.	Required	Optional
REPGEN	Report Generator – not fully supported due to different command line argument conventions.			
DSSTS	Regular Interval Time-Series Data Entry Program	Required	Required	Optional
DSSITS	Irregular Interval Time-Series Data Entry Program	Required	Required	Optional
DSSPD	Paired Data Entry Program	Required	Required	Optional
DSSTXT	Text Data Entry Program	Required	Required	Optional
DWINDO	Interactive Data Entry and Editing	This interactive program is not supported by RunDSSUTL () command.		
WATDSS	Watstore to DSS Data Entry Program – not fully supported due to different command line argument	Required	Required	Optional

Program	Description	DSSFILE= Argument	INPUT= Argument	OUTPUT= Argument
	conventions.			
NWSDSS	National Weather Service to Data Storage System Conversion Utility – not fully supported due to different command line argument conventions.	Required	Required	Optional
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.



RunDSSUTL() Command Editor when Specifying Command Line

The command syntax is as follows:

```
RunDSSUTL (Parameter=Value...)
```

Command Parameters

Parameter	Description	Default
DssFile	The HEC-DSS filename as an absolute path or relative to the working directory. The file must exist because TSTool does not interface with the program interactive mode prompts. The parameter is passed to the program using the DSSFILE= command line argument.	None – must be specified for most programs.
InputFile	The DSS utility program command file to run. The file must exist because TSTool does not interface with the utility program interactive mode prompts. The input file name is passed to the program using the INPUT= command line argument.	None – must be specified.
OutputFile	The DSS utility program output file, which contains logging information. This is passed to the program using the OUTPUT= command line argument. Specifying the argument will cause 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 argument.	Not required – output will be to screen if command shell window is shown.
DssutlProgram	The DSS utility program to run. The PATH environment variable is used to locate the executable if a full path is not specified. Specify the specific DSS utility program to run if the default value is not appropriate.	If not specified, <i>DSSUTL.EXE</i> will be used and must be located in a directory listed in the PATH environment variable.

This page is intentionally blank.

Command Reference: RunProgram()

Run an external program

Version 09.03.00, 2009-04-08

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() 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):

```
echo Hello > ${WorkingDir}/Results/Test_RunProgram_CommandLine_echo_out.txt
```

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

The command syntax is as follows:

```
RunProgram(Parameter=Value...)
```

Command Parameters

Parameter	Description	Default
CommandLine	<p>The full program command line, with arguments. If the program executable is found in the PATH environment variable, then only the program name needs to be specified. Otherwise, specify an absolute path to the program or run TSTool from a command shell the same directory.</p> <p>The <code>\${WorkingDir}</code> property can be used in the command line to indicate the working directory (command file location) when specifying file names.</p> <p>For Windows, it may be necessary to place a <code>\</code> at the start and end of the command line, if a full command line is specified.</p>	<p>Must be specified if the Program parameter is not specified.</p> <p>The Program parameter will be used if both are specified.</p>
Program	The name of the program to run. Program arguments are specified using the ProgramArg# parameter(s). See the CommandLine parameter for more information about parameter formatting and locating the executable.	Must be specified if the CommandLine parameter is not specified.
ProgramArg1, ProgramArg2, etc.	Command like arguments used with Program. If necessary, use <code>\${WorkingDir}</code> to specify the working directory to locate files.	No arguments will be used with Program.
UseCommandShell	If specified as False, the program will be run without using a command shell. A command shell is needed if the program is a script (batch file), a shell command, or uses <code>></code> , <code> </code> , etc.	True, using <code>cmd.exe /C</code> on Windows and <code>/bin/sh -c</code> on UNIX/Linux.
Timeout	The timeout in seconds – if the program has not yet returned, the process will be ended. Zero indicates no timeout. This behavior varies and is being enhanced.	No timeout.
ExitStatus Indicator	By default, the program exit status is determined from the process that is run. Normally 0 means success and non-zero indicates an error. 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).	Determine the exit status from the process exit value.

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 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:	<input type="text" value="echo"/>	Required - if full command line is not specified above.
Program argument 1:	<input type="text" value="Hello"/>	Optional - as needed if Program is specified.
Program argument 2:	<input type="text" value=">"/>	Optional - as needed if Program is specified.
Program argument 3:	<input type="text" value="\${WorkingDir}/Results/Test_RunProgram_Program_echo_out.txt"/>	Optional - as needed if Program is specified.
Program argument 4:	<input type="text"/>	Optional - as needed if Program is specified.
Program argument 5:	<input type="text"/>	Optional - as needed if Program is specified.
Program argument 6:	<input type="text"/>	Optional - as needed if Program is specified.
Program argument 7:	<input type="text"/>	Optional - as needed if Program is specified.
Program argument 8:	<input type="text"/>	Optional - as needed if Program is specified.
Use command shell:	<input checked="" type="checkbox"/>	Optional - use command shell (default=True).
Timeout (seconds):	<input type="text"/>	Optional - default is no timeout.
Exit status indicator:	<input type="text"/>	Optional - output string to indicate status (default=use process exit status).

Command:

```
RunProgram(Program=echo,ProgramArg1=Hello,ProgramArg2=>,ProgramArg3=${WorkingDir}/Results/Test_RunProgram_Program_echo_out.txt)
```

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: ☐ 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}/testecho.exe", ProgramArg1="STOP
2", ExitStatusIndicator="STOP")
```

Cancel OK

RunProgram_Program_ExitStatusIndicator

RunProgram() Command Editor when Specifying Program, Arguments, and Exit Status Indicator

Command Reference: RunPython()

Run a Python script

Version 09.07.00, 2010-07-14

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:

RunPython() Supported Python Interpreters

Interpreter (Website)	Language, Program Name (Example Install Home)	Comments
IronPython (ironpython.net)	.NET, ipy (C:\Program Files\IronPython 2.6)	Useful for integrating with .NET 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 (www.jython.org)	Java, jython (C:\jython2.5.1)	Useful for integrating with Java applications, such as TSTool. Can use Java code (but this code in a Python script is only recognized by Jython).
Jython embedded (www.jython.org)	Java (C:\jython2.5.1, but must use the installer option to create a JAR file in order to embed – this is the file that is distributed with TSTool).	Useful for integrating with Java applications, such as TSTool. Can use Java code (but this code in a Python script is only recognized by Jython). Integration can occur within a running Java application (essentially extending the functionality of the Java application).
Python (www.python.org)	C, python (C:\Python25)	The original Python interpreter, which defines the Python language specification.

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 RunPython() Command

Run a Python script, by calling a stand-alone interpreter or embedded Jython interpreter.
 Python scripts are useful for manipulating data outside of TSTool's capabilities.
 IronPython is the .NET implementation of Python and Jython is the Java implementation, offering integration with packages available for each language.
 Specify a full or relative path to the script file (relative to working directory).
 Strings with special meaning include:
 \" - literal quote, needed to surround arguments that include spaces.
 \${InstallDir} - the software installation directory.
 \${WorkingDir} - the working directory (location of command file).
 The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\RunPython

Interpreter: 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:

Arguments:

Command:

RunPython

RunPython() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
Interpreter	The Python interpreter to run, one of: <ul style="list-style-type: none"> IronPython Jython JythonEmbedded Python 	None – must be specified.
Program	The Python interpreter program to run. Specify as a full path to the installed program, or only the program name (in which case the path to the program must be included in the PATH environment variable).	Determined based on the Interpreter parameter: <ul style="list-style-type: none"> IronPython: ipy Jython: jython Python: python
PythonPath	Additional locations for modules, to be added to the Python path. Specify paths separated by ; or :. For embedded Jython, the sys.path is updated prior to running the script. For non-embedded interpreters, the JYTHONPATH environment variable is updated for the interpreter, which results in sys.path being updated.	None – the core Python capabilities are available.
InputFile	The Python script to run, specified as an absolute path or relative to the command file. See the Arguments parameter for information about using properties to specify the location.	None – must be specified.
Arguments	Arguments to pass to the script, such as the names of files to process. Use the \${WorkingDir} property to specify the location of the command file. Use \${InstallDir} for the TSTool install folder. Use \" to surround arguments that include spaces. Separate arguments by a space.	None – arguments are optional.

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)
```

Command Reference: Scale()

Scale time series data values by a constant value

Version 08.15.00, 2008-05-11

The `Scale()` command scales each non-missing value in the specified time series.

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

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 processed, one of: <ul style="list-style-type: none"> 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 	AllTS

Parameter	Description	Default
	selected with the <code>SelectTimeSeries()</code> 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.
ScaleValue	One of the following: <ul style="list-style-type: none"> 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.
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 09.09.00, 2010-09-23

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.

Edit SelectTimeSeries() Command

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 series.

When matching a time series identifier (TSID) pattern:

- The dot-delimited time series identifier parts are 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 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 work 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: 1,5-10,13

When selecting time series by matching a property:

- Currently only string properties are supported.
- Comparisons are case-independent.

TS list: Optional - indicates the time series to process (default=AllTS).

TSID (for TSLIST=AllMatchingTSID):

EnsembleID (for TSLIST=EnsembleID):

Time series position(s) (for TSLIST=TSPosition): For example, 1,2,7-8 (positions are 1+).

Deselect all first?: Optional - eliminates need for separate deselect (default=False).

Property name: Optional - use to match user-defined properties.

Property criterion: Required if matching user-defined property.

Property value: Required if matching user-defined property.

Command:

```
SelectTimeSeries(TSLIST=AllMatchingTSID, TSID="40*", DeselectAllFirst=True)
```

SelectTimeSeries

SelectTimeSeries() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none"> 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 list (see TSPosition parameter below). 	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
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.

A sample command file is as follows:

```
TS 401234 = NewPatternTimeSeries(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)
```

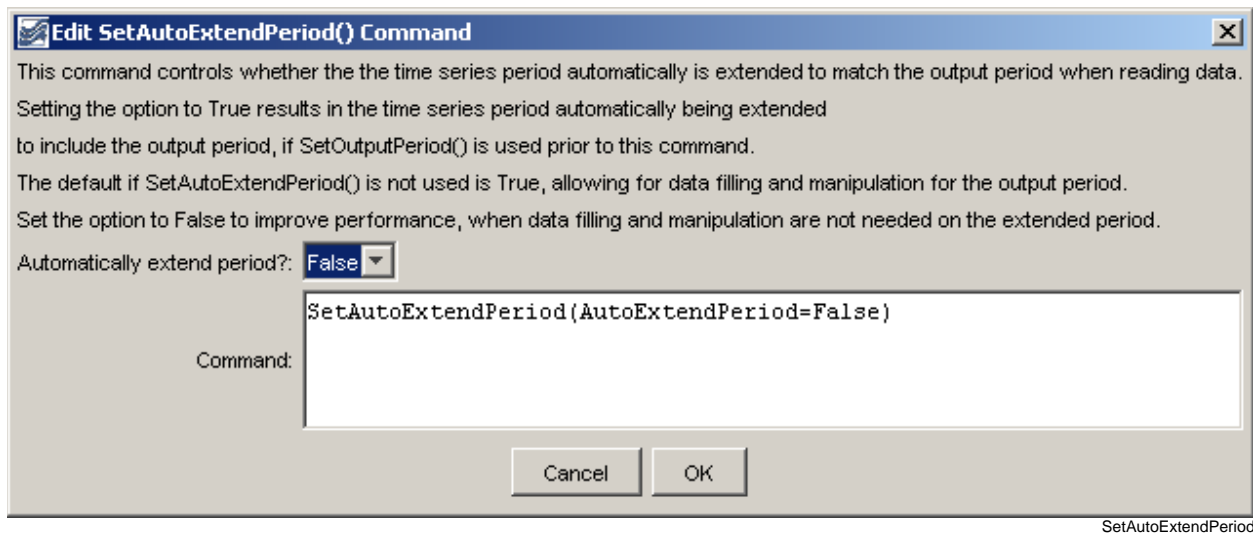
Command Reference: SetAutoExtendPeriod()

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.



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, True or False.	None – must be specified. The default is True if this command is not used.

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 averages for data filling is only supported for monthly and yearly time series.
Averages are by default computed for the available period.
Enter dates as MM/YYYY or YYYY-MM (or YYYY for yearly). Enter * to use all available data.

Average period start:

Average period end:

Command:

SetAveragePeriod() Command Editor

SetAveragePeriod

The command syntax is as follows:

```
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 yearly precision.	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 or smaller, constant values for each month can be specified.
In this case, each date/time that matches a month will have its corresponding value set.

TS list: **LastMatchingTSID** Indicates the time series to process (default=AllTS).

TSID (for TSList=AllMatchingTSID): 08235700.DWR.Streamflow.Month

EnsembleID (for TSList=EnsembleID):

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=LastMatchingTSID,TSID="08235700.DWR.Streamflow.Month",ConstantValue=0,SetEnd="1950-01")

Cancel OK

SetConstant

SetConstant() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none"> 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.
ConstantValue	The constant value to use as the data value.	None – must be specified, or specify monthly values.
MonthValues	Monthly values to use as the data values. Twelve values can be specified, separated by commas. If the time series data interval is less than monthly, each date/time will be set for a specific month.	None – must be specified, or specify a constant value.
SetStart	The starting date/time for the data set.	Set data for the full period.
SetEnd	The ending 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 SetDataValue() Command

Set a data value for a specific date/time.

Specify the date/time to an appropriate precision using standard formats like the following:

- YYYY for year interval
- MM/YYYY or YYYY-MM for month interval
- 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=AllTS).

TSID (for TSList=AllMatchingTSID): 08235700.DWR.Streamflow.Month

EnsembleID (for TSList=EnsembleID):

Date/time to set value: 1950-01

New data value: 550

Command:

```
SetDataValue(TSList=AllMatchingTSID,TSID="08235700.DWR.Streamflow.Month",SetDateTime="1950-01",NewValue=550)
```

Cancel OK

SetDataValue

SetDataValue() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none"> 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 processing an ensemble.	Required if TSList=EnsembleID.
SetDateTime	The date/time at which the data value should be set. Specify the date/time precision according to the time series that is being manipulated.	None – must be specified.
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 - ALAMOS 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 SetDebugLevel() command

Set the level for screen and/or log file debug messages.
Debug information is useful for troubleshooting. The default debug level is 0.
Setting the debug level to a higher number prints more information.
Debug levels can be increased before and decreased after specific commands to troubleshoot the commands.

Screen debug level: 0=none, 100=all, blank=no change.

Log file debug level: 0=none, 100=all, blank=no change.

Command:

Cancel OK

SetDebugLevel

SetDebugLevel() Command Editor

The command syntax is as follows:

```
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:

```
SetDebugLevel (ScreenLevel=0,LogFileLevel=10)
```


Command Reference: SetFromTS()

Set time series data using another time series

Version 08.16.03, 2008-08-18

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.

SetFromTS

SetFromTS() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none"> AllMatchingTSID – all time series that 	AllTS

Parameter	Description	Default
	<p>match the TSID (single TSID or TSID with wildcards) will be modified.</p> <ul style="list-style-type: none"> • 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=EnsembleID.
Independent TSList	Indicates how to determine the list of independent time series (see the explanation of TSList).	AllTS
Independent TSID	The time series identifier or alias for the independent time series (see the explanation of TSID).	Required when a IndependentTSList=*TSID
Independent EnsembleID	The ensemble identifier for the independent time series (see the explanation of EnsembleID).	Required when IndependentTSList=EnsembleID.
SetStart	The date/time to start setting data, if other than the full time series period.	Full period if * is specified.
SetEnd	The date/time to end setting data, if other than the full time series period.	Full period if * is specified.
TransferHow	<p>Indicates how to transfer data:</p> <ul style="list-style-type: none"> • ByDateTime – a date/time in one time series will be lined up with the other time series. • Sequentially – data from the independent will be transferred sequentially, even if the date/time does not align (used when transferring continuous data over Feb 28/29, without gaps). 	None – must be specified.
HandleMissingHow	<p>Indicates how to handle missing data in the independent time series:</p> <ul style="list-style-type: none"> • IgnoreMissing – missing values in the 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. 	SetMissing
RecalcLimits	Available only for monthly time series. Indicate	False (only the values in

Parameter	Description	Default
	whether the original data limits for the time series should be recalculated after the setting the time series values. Setting to True is appropriate if the independent time series provides observations consistent with the original data.	the initial time series will be used for historical 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.

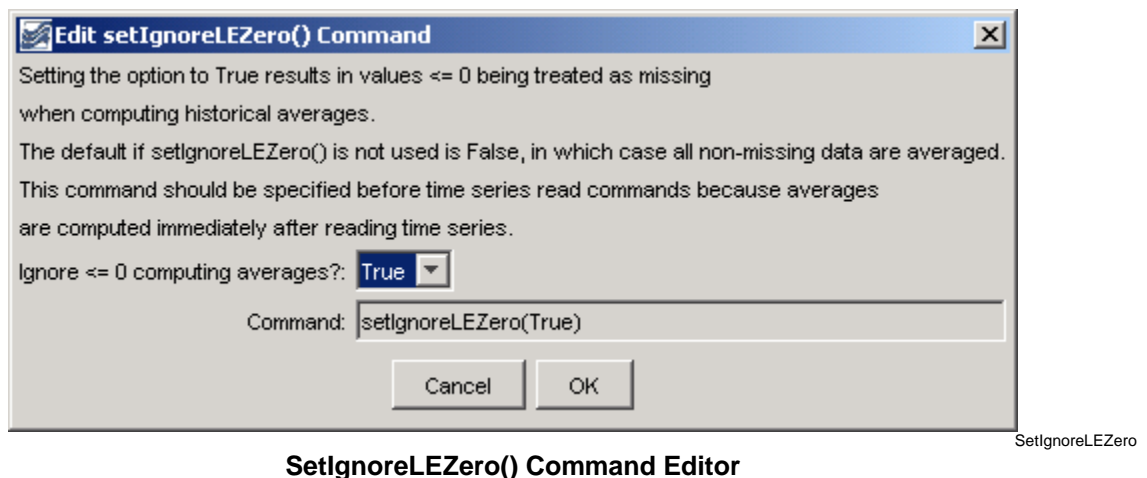
Command Reference: SetIgnoreLEZero()

Indicate whether time series data values \leq 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.



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.

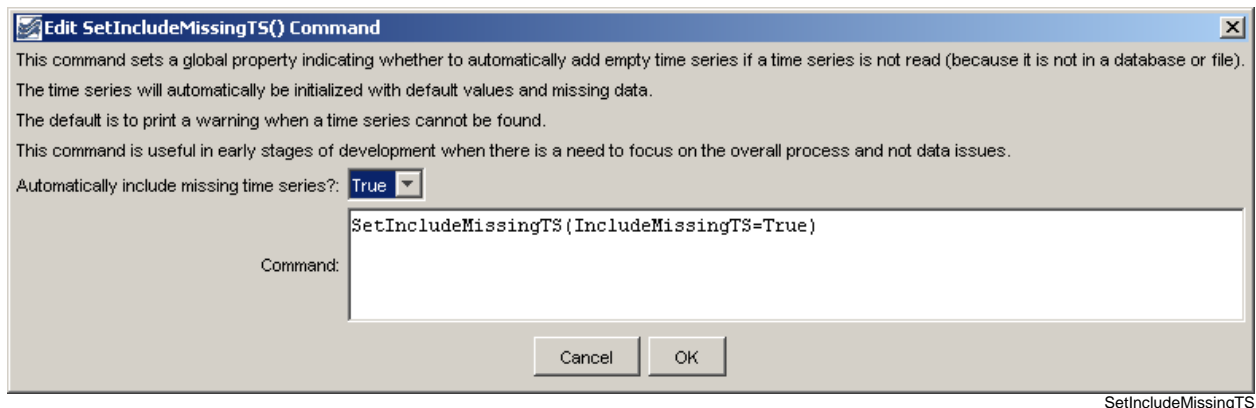
Command Reference: SetIncludeMissingTS()

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.



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 found with read commands should automatically be added with missing data.	If this command is not used, the default is False.

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: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:

Input period end:

Command:

SetInputPeriod() Command Editor

SetInputPeriod

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
InputStart	<p>The date/time to start reading/querying time series data, one of:</p> <ul style="list-style-type: none"> • 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	<p>The date/time to end reading/querying time series data, one of:</p> <ul style="list-style-type: none"> • 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 SetOutputPeriod() Command

Set the global (default) output period for time series and output products.
 The time series period after reading typically will be extended to the output period by using the missing value.
 Use a SetOutputPeriod() command to guarantee a longer period when filling/extending data.
 Specify the command at the top of commands when filling a specific period.
 Enter date/times to a precision appropriate for time series being processed. 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: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

See also the SetInputPeriod() command, which will constrain the period that is read.

Output period start: Required

Output period end: Required

Command:

SetOutputPeriod

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:

```
SetOutputPeriod(OutputStart="1950-01", OutputEnd="2002-12")
```

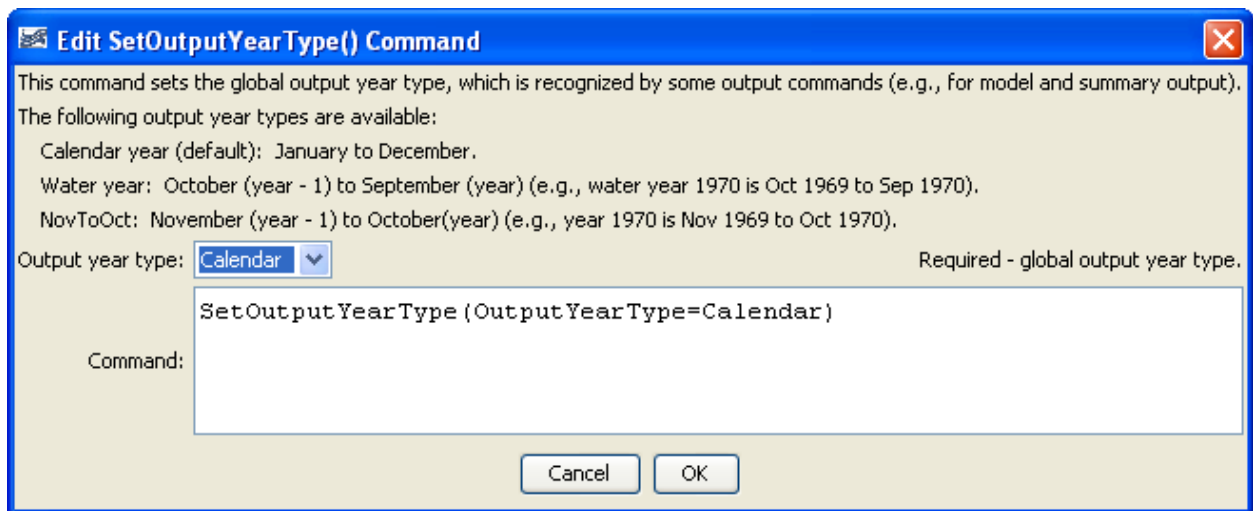
Command Reference: SetOutputYearType()

Set the output year type for time series

Version 09.05.02, 2009-11-02

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



SetOutputYearType

SetOutputYearType() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
OutputYearType	<p>The output year type, one of:</p> <ul style="list-style-type: none">• <code>Calendar</code> – January through December.• <code>NovToOct</code> – November of the previous calendar year to October of the current calendar year. For example, year 1970 spans Nov 1969 to Oct 1970.• <code>Water</code> – 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. <p>In the future, more generic types like <code>NovToOct</code> may be implemented.</p>	If this command is not specified, <code>Calendar</code> is the default.

A sample commands file is as follows:

```
SetOutputYearType (OutputYearType=Calendar)
```

Command Reference: SetPatternFile()

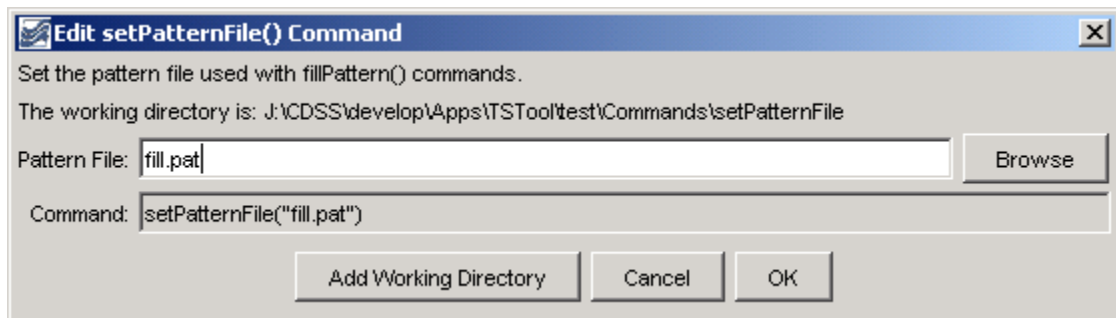
Set the pattern file to be used with FillPattern() commands

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.



SetPatternFile

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 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).	None – must be specified.

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 09.08.02, 2010-09-23

The `SetProperty()` command sets the value of a property used by the time series processor, which means that the properties are available to all commands. These properties will ultimately be accessible by any command using `${Property}` notation, for example to specify filenames more dynamically. However, currently only a few commands utilize the properties. 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 SetProperty() Command

Set a property for the processor. The property can be referenced in parameters using `${Property}` notation.

Property name: Required - do not use spaces \$, { or } in name.

Property type: Required - to ensure proper initialization.

Property value: Required.

Command: `SetProperty(PropertyName="Scenario",PropertyType=DateTime,PropertyValue="Likely")`

SetProperty

SetProperty() Command Editor

The command syntax is as follows:

```
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: <ul style="list-style-type: none">• DateTime – a date/time.• Double – a floating point number• Integer – an integer• String – a string	None – must be specified.
PropertyValue	The value of the property, adhering to property type constraints.	None – must be specified.

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

Version 09.09.00, 2010-09-23

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 matching input from a table.
For example, set properties for a location associated with the time series.
The table value is determined from a row 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=AllTS).

TSID (for TSList=AllMatchingTSID):

EnsembleID (for TSList=EnsembleID):

Table ID: Required - table to process.

Table TSID column: Required - column name for TSID.

Format of TSID: Insert: Optional - use %L for location, etc. (default=alias or TSID).

Table input columns: Required - column names from which to set properties.

Command:

```
SetTimeSeriesPropertiesFromTable (TableID="Table1", TableTSIDColumn="loc", TableTSIDFormat="%L", TableInputColumns="Scenario,Status")
```

SetTimeSeriesPropertiesFromTable

SetTimeSeriesPropertiesFromTable() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none"> 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. 	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
TableID	The identifier for the table that contains properties.	None – must be specified.
TableTSIDColumn	Table column name that is used to match the time series identifier for processing.	None – must be specified.
TableTSIDFormat	The specification to format the time series identifier to match the TSID column. Use the format choices and other characters to define a unique identifier.	Time series alias if available, or otherwise the time series identifier.
TableInputColumns	The name(s) of the column(s) to supply properties for the matching time series. Separate column names with commas.	None – must be specified.

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.

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: <ul style="list-style-type: none"> 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 last time 	AllTS

	<p>series that matches the TSID (single TSID or TSID with wildcards) will be modified.</p> <ul style="list-style-type: none"> • 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.
Description	The description to assign to the time series. Use the format choices and other characters to define a unique alias.	None.
Units	The data units to assign to the time series. The units should agree with the time series data values.	None.
Editable	If set to True, then graphing the time series will enable interactive editing features, including the ability to save the edited time series.	False
PropertyName	Name of user-defined property.	
PropertyType	Property type, to ensure proper initialization and data check.	Required if PropertyName is specified.
PropertyValue	Value for property, adhering to the property type requirements.	Required if PropertyName is 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.

Edit SetToMax() Command

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=AllTS).

Independent TSID (for Independent TSList=AllMatchingTSID): 08236500.DWR.Streamflow.Month

Independent EnsembleID (for Independent TSList=EnsembleID):

Independent specified TSID (for IndependentTSList=SpecifiedTSID): 08236000.DWR.Streamflow.Month
08236500.DWR.Streamflow.Month

Command: SetToMax(TSID="08236000.DWR.Streamflow.Month", IndependentTSList=SpecifiedTSID, IndependentTSID="08236500.DWR.Streamflow.Month")

Cancel OK

SetToMax

SetToMax() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TSID	The time series identifier or alias for the time series to be modified.	None – must be specified.
IndependentTSList	Indicates how the list of time series is specified, one of: <ul style="list-style-type: none"> AllTS – all time series before the command. 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. 	AllTS (the time series receiving the result will not be checked)
IndependentTSID	If the IndependentTSList=SpecifiedTSID, provide the list of time series identifiers (or alias) to process, separated by commas. If the IndependentTSList parameter is AllMatchingTSID, FirstMatchingTSID, or LastMatchingTSID, specify a single TSID or a TSID with wildcards.	Required if TSList=*TSID.
IndependentEnsembleID	Ensemble identifier.	Required if 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
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.

Edit SetToMin() Command

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 Indicates the time series to process (default=AllTS).

Independent TSID (for Independent TSList=AllMatchingTSID): 08236500.DWR.Streamflow.Month

Independent EnsembleID (for Independent TSList=EnsembleID):

Independent specified TSID (for IndependentTSList=SpecifiedTSID):

- 08236000.DWR.Streamflow.Month
- 08236500.DWR.Streamflow.Month

Command:

```
SetToMin(TSID="08236000.DWR.Streamflow.Month", IndependentTSList=SpecifiedTSID, IndependentTSID="08236500.DWR.Streamflow.Month")
```

Cancel OK

SetToMin

SetToMin() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TSID	The time series identifier or alias for the time series to be modified.	None – must be specified.
IndependentTSList	Indicates how the list of time series is specified, one of: <ul style="list-style-type: none"> AllTS – all time series before the command. 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. 	AllTS (the time series receiving the result will not be checked)
IndependentTSID	If the IndependentTSList=SpecifiedTSID, provide the list of time series identifiers (or alias) to process, separated by commas. If the IndependentTSList parameter is AllMatchingTSID, FirstMatchingTSID, or LastMatchingTSID, specify a single TSID or a TSID with wildcards.	Required if TSList=*TSID.
IndependentEnsembleID	Ensemble identifier.	Required if 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.



SetWarningLevel

SetWarningLevel() Command Editor

The command syntax is as follows:

```
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 08.16.03, 2008-07-30

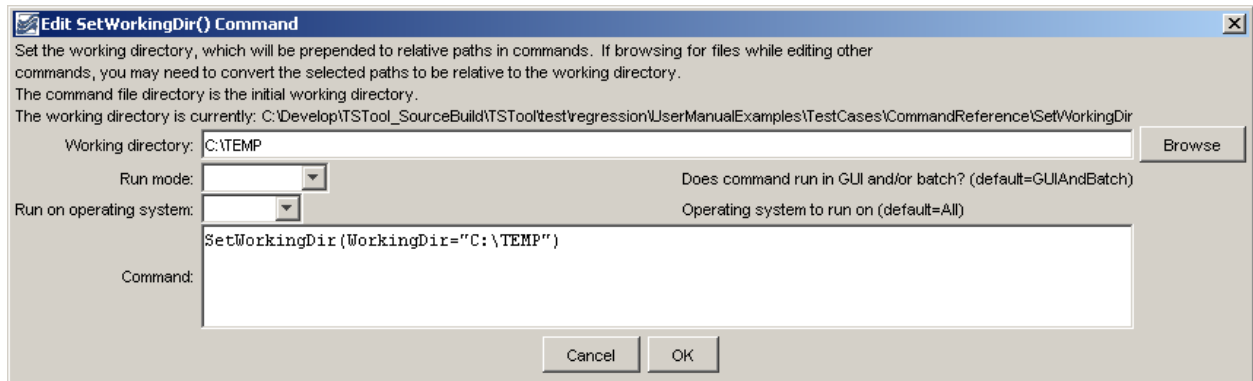
The `SetWorkingDir()` command sets the working directory for following commands. The working directory is normally set in one of the following ways, with the current setting being defined by the most recent action that has occurred:

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.



SetWorkingDir

SetWorkingDir() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
WorkingDir	The working directory that should be used. Specify a relative path (e.g., "..") to adjust the current working directory.	None – must be specified.
RunMode	Indicate the run mode in which the command should be applied, one of: <ul style="list-style-type: none">• 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	GUIAndBatch

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: Indicates the time series to process (default=AllTS).

TSID (for TSList=AllMatchingTSID):

EnsembleID (for TSList=EnsembleID):

Shift offset, weight pairs:

Command:
`ShiftTimeByInterval(TSList=AllMatchingTSID,TSID="08213500.DWR.Streamflow.Day",ShiftData="-1,1")`

ShiftTimeByInterval

ShiftTimeByInterval() Command Editor

The command syntax is as follows:

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

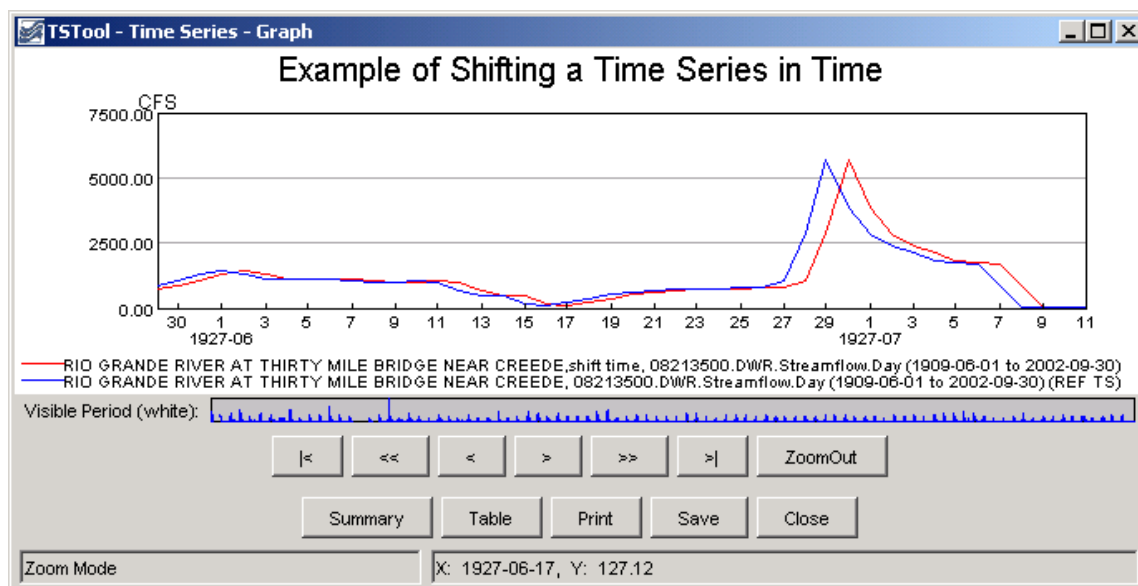
Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none">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	AllTS

Parameter	Description	Default
	modified. <ul style="list-style-type: none"> • 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
```



ShiftTimeByInterval_graph

Results from ShiftTimeByInterval() Command

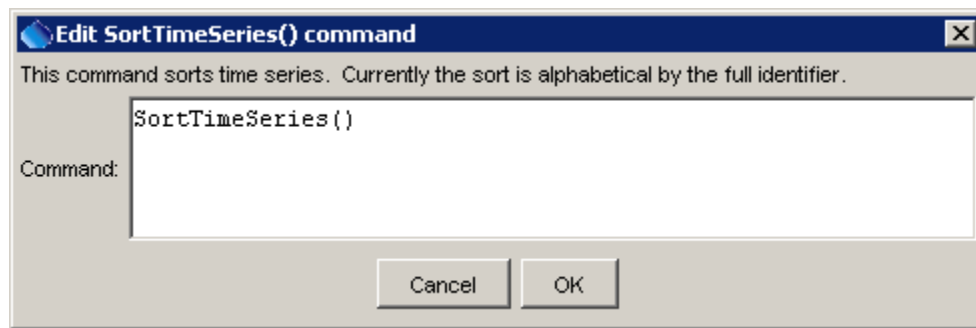
Command Reference: SortTimeSeries()

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.



SortTimeSeries

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 StartLog() command

(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:

Suffix: Optional - suffix for log file (blank=none).

Command:

StartLog

StartLog() Command Editor

The command syntax is as follows:

```
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	<p>Indicates that a suffix will be added before the <i>.log</i> extension, one of:</p> <ul style="list-style-type: none"> ▪ Date – add a date suffix of the form YYYYMMDD. ▪ DateTime – add a date/time suffix of the form YYYYMMDD_HHMMSS. <p>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.</p>	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",AddTSLList=AllTS,HandleMissingHow="IgnoreMissing")
```

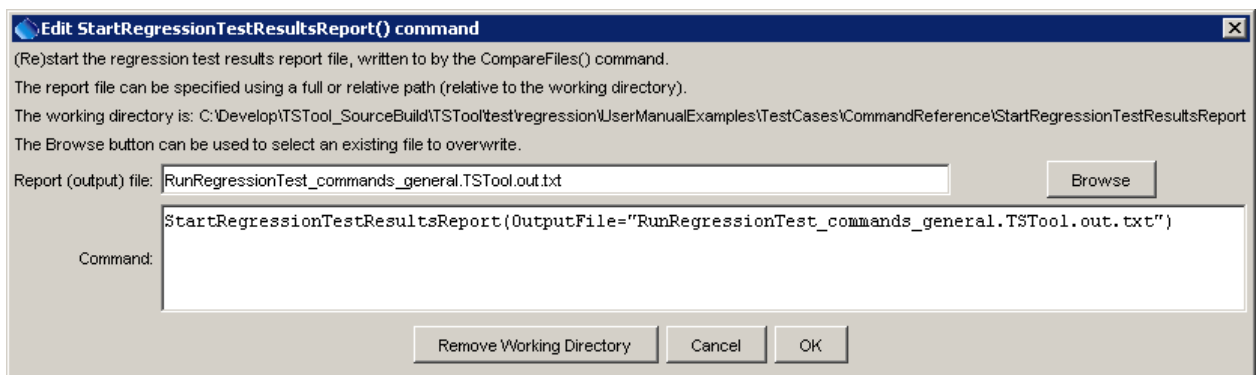
Command Reference: StartRegressionTestResultsReport()

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.



StartRegressionTestResultsReport

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 the file contains spaces or other special characters. A path relative to the command file can be specified.	None – must 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 C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\ChangePeriod\Test_ChangePeriod_1.TSTool
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\CreateTracesAlias\Test_CreateTraces_1.TSTool
SUCCESS C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\cumulate\Test_Cumulate_1.TSTool
SUCCESS
C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\DeselectTimeSeries\Test_DeselectTimeSeries_1.TSTool
SUCCESS C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\DisaggregateAlias\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 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). The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\StateModMax

First StateMod file to read:

Second StateMod file to read:

Command:

StateModMax

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 series file to read, surrounded by double quotes. The path to the file can be absolute or relative to the working directory.	None – must be specified.
InputFile2	The name of the second StateMod time series file to read, which must have the same data interval and units as the first file.	None – must be specified.

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 ensemble of time series). The receiving time series (or ensemble) is modified.

The time series to be subtracted are selected using the TS list parameter:

- AllMatchingTSID - subtract all previous time series with matching identifiers.
- AllTS - subtract all previous time series.
- SelectedTS - subtract time series selected with selectTimeSeries() commands
- SpecifiedTSID - subtract time series selected from the list below

Time series to receive results: 0100501.DWR.DivTotal.Month

Ensemble to receive results:

Time series to subtract (SubtractTSLIST): SpecifiedTSLIST Indicates the time series to process (default=AllTS).

Subtract TSLIST (for TSLIST=AllMatchingTSLIST): 0100503.DWR.DivTotal.Month

Add EnsembleID (for SubtractTSLIST=EnsembleID):

Subtract specified TSLIST (for SubtractTSLIST=SpecifiedTSLIST):

- 0100501.DWR.DivTotal.Month
- 0100503.DWR.DivTotal.Month

Handle missing data how?: IgnoreMissing

Command:

```
Subtract(TSID="0100501.DWR.DivTotal.Month",SubtractTSLIST=SpecifiedTSLIST,SubtractTSID="0100503.DWR.DivTotal.Month",HandleMissingHow="IgnoreMissing")
```

Cancel OK

Subtract

Subtract() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TSID	The time series identifier or alias for the time series to receive the result.	TSID or EnsembleID must be specified.
EnsembleID	The ensemble to receive the result, if processing an ensemble.	TSID or EnsembleID must be specified.
Subtract TSList	Indicates how the list of time series is specified, one of: <ul style="list-style-type: none"> AllTS – all time series before the command. AllMatchingTSID – all time series that match the AddTSID (single TSID or TSID with wildcards) will 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. SelectedTS – the time series are those selected with the SelectTimeSeries() command. SpecifiedTSID – the specified list of time series given by the SubtractTSID parameter. If using version 8.02.00 or earlier, use SpecifiedTS. 	AllTS (the time series receiving the sum will not be subtracted from itself)
SubtractTSID	If the SubtractTSList parameter is SpecifiedTSID, provide the list of time series identifiers (or alias) to subtract, separated by commas. If the SubtractTSList parameter is AllMatchingTSID, specify a single TSID or a TSID with wildcards.	Must be specified if TSList=SpecifiedTSID, ignored otherwise.
Subtract EnsembleID	If the EnsembleID parameter is specified, providing an ensemble ID will subtract the ensembles.	Use if an ensemble is being subtracted from another ensemble.
Handle MissingHow	Indicates how to handle missing data in a time series, one of: <ul style="list-style-type: none"> 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

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 `ts1` are multiplied by the number 2).

Edit TableMath() Command

Perform simple math 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 enhancements may provide more cell range addressing - currently full columns are processed.

Table ID: Required - table to process.

Input 1: Required - first input column name.

Math operator: Required - math calculation to perform on input.

Input 2: Required - second input column name, or constant.

Output column: Required - output column name.

Non-value: Optional - non-value for missing, unable to compute (default=Null).

Command:

```
TableMath(TableID="Table1", Input1=ts1, Operator="*", Input2=2, Output=result)
```

TableMath

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: Input1 Operator Input2 = Output For example: Input1 * Input2 = Output	None – must be specified.
Input2	Second input column name, or a constant value to use as input.	None – must be specified.
Output	Output column name. If the column is not found it will be added to the table and will contain the results of processing.	None – must be specified.
NonValue	The value to use in cases where an output result could not 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.	Null

Command Reference: TableTimeSeriesMath()

Perform simple math operation on time series using table input

Version 09.08.01, 2010-09-15

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.

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.

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 series by a value from a table.
The table value is determined from a row 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=AllTS).

TSID (for TSList=AllMatchingTSID):

EnsembleID (for TSList=EnsembleID):

Math operator: Required - math calculation to perform on input.

Table ID: Required - table to process.

Table TSID column: Required - column name for TSID.

Format of TSID: Insert: Optional - use %L for location, etc. (default=alias or TSID).

Table input column: Required - column name for table input.

Command:

```
TableTimeSeriesMath(TSList=AllMatchingTSID, TSID="ts1", Operator="*",  
TableID="Table1", TableTSIDColumn="TSID", TableTSIDFormat="%L", TableI  
nputColumn="DataValue")
```

TableTimeSeriesMath

TableTimeSeriesMath() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none"> 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 <code>SelectTimeSeries()</code> command. 	AllTS
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.
Operator	The operator to be applied to the time series and table input.	None – must be specified.
TableID	Identifier for table that provides input.	None – must be specified.
TableTSIDColumn	Table column name that is used to match the time series identifier for processing.	None – must be specified.
TableTSIDFormat	The specification to format the time series identifier to match the TSID column. Use the format choices and other characters to define a unique identifier.	Time series alias if available, or otherwise the time series identifier.
TableInput Column	Table column name to retrieve the table value.	None – must be specified.

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 `ts1` and `ts2`.

```
# Simple test data
"TSID","DataValue"
ts1,2
ts2,3
```

Command Reference: TimeSeriesToTable()

Copy one or more time series into a table

Version 09.05.00, 2009-10-06

The `TimeSeriesToTable()` command copies one or more time series into a table. The time series must be regular interval (no irregular interval time series) and the intervals must match. 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. This command is useful when performing table analysis processing and outputting table formats (e.g., with the `WriteTableToDelimitedFile()` command).

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

TimeSeriesToTable

TimeSeriesToTable() Command Editor

The command syntax is as follows:

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

Command Parameters

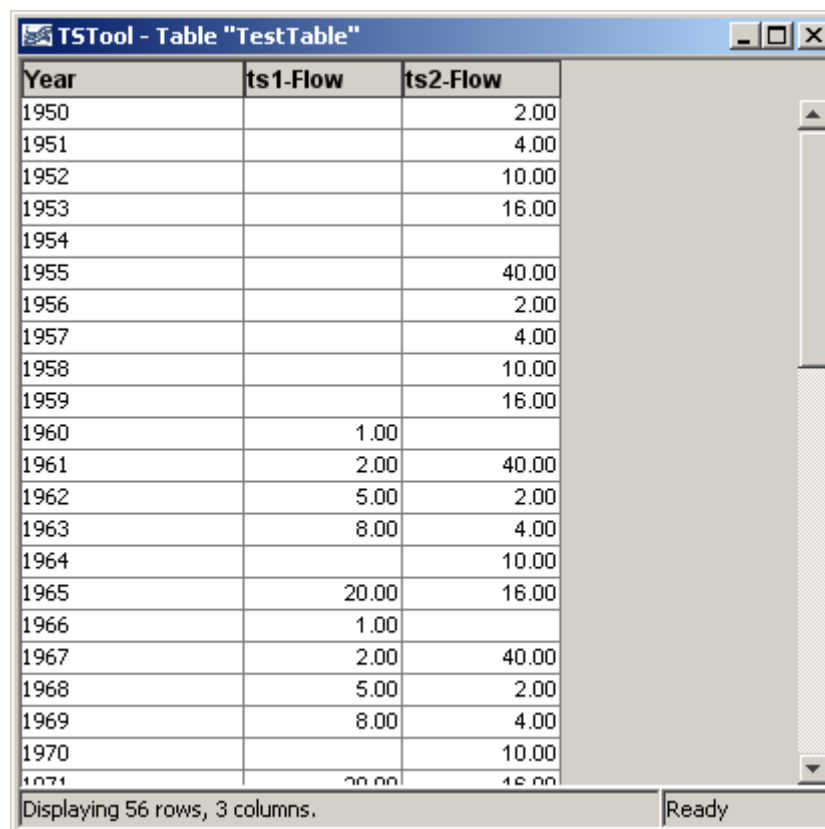
Parameter	Description	Default
TsList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none"> AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will 	AllTS

Parameter	Description	Default
	<p>be modified.</p> <ul style="list-style-type: none"> 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 <code>SelectTimeSeries()</code> 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 <code>TSList=*TSID</code>
EnsembleID	The ensemble to be modified, if processing an ensemble.	Required when <code>TSList=EnsembleID.</code>
TableID	The identifier for the table to copy data into (or the identifier for the new table to create if <code>IfTableNotFound=Create</code>).	None – must be specified.
DateTimeColumn	The table column name to receive date/time information.	None – must be specified.
DataColumn	The data column name(s) to receive time series data. This parameter may in the future allow multiple names separated by a delimiter. However, multiple names are currently supported by using 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.	None – must be specified.
DataRow1	First table row for data (1+), where the row number is data only (column names are not considered a data row).	None – must be specified.
OutputStart	The starting date/time for the copy.	Available period.
OutputEnd	The ending date/time for the copy.	Available period.
IfTableNotFound	<p>Indicate action if the table identifier is not matched, one of:</p> <ul style="list-style-type: none"> Create – create a new table Warn – warn that the table was not matched 	Warn

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)
TS ts1 = NewPatternTimeSeries(NewTSID="ts1..Flow.Year",SetStart="1960",
  SetEnd="2000",Units="ACFT",PatternValues="1,2,5,8,,20")
TS ts2 = NewPatternTimeSeries(NewTSID="ts2..Flow.Year",SetStart="1950",
  SetEnd="2005",Units="ACFT",PatternValues="2,4,10,16,,40")
TimeSeriesToTable(TableID=TestTable,DateTimeColumn=Year,DataColumn=%L-%T,
  DataRow=1,IfTableNotFound="Create")
# Generate the results.
WriteTableToDelimitedFile(TableID="TestTable",
  OutputFile="Results\Test_TimeSeriesToTable_Year_Create_out.csv")
# Uncomment the following to recreate expected results
# WriteTableToDelimitedFile(TableID="TestTable",
#   OutputFile="ExpectedResults\Test_TimeSeriesToTable_Year_Create_out.csv")
CompareFiles(InputFile1="ExpectedResults/Test_TimeSeriesToTable_Year_Create_out.csv",
  InputFile2="Results/Test_TimeSeriesToTable_Year_Create_out.csv",IfDifferent=Warn)
```

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:



Year	ts1-Flow	ts2-Flow
1950		2.00
1951		4.00
1952		10.00
1953		16.00
1954		
1955		40.00
1956		2.00
1957		4.00
1958		10.00
1959		16.00
1960	1.00	
1961	2.00	40.00
1962	5.00	2.00
1963	8.00	4.00
1964		10.00
1965	20.00	16.00
1966	1.00	
1967	2.00	40.00
1968	5.00	2.00
1969	8.00	4.00
1970		10.00
1971	20.00	16.00

TimeSeriesToTabl2

This page is intentionally blank.

Command Reference: VariableLagK()

Lag and attenuate (route) a time series with parameters that vary by rate

Version 09.03.04, 2009-04-22

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**).

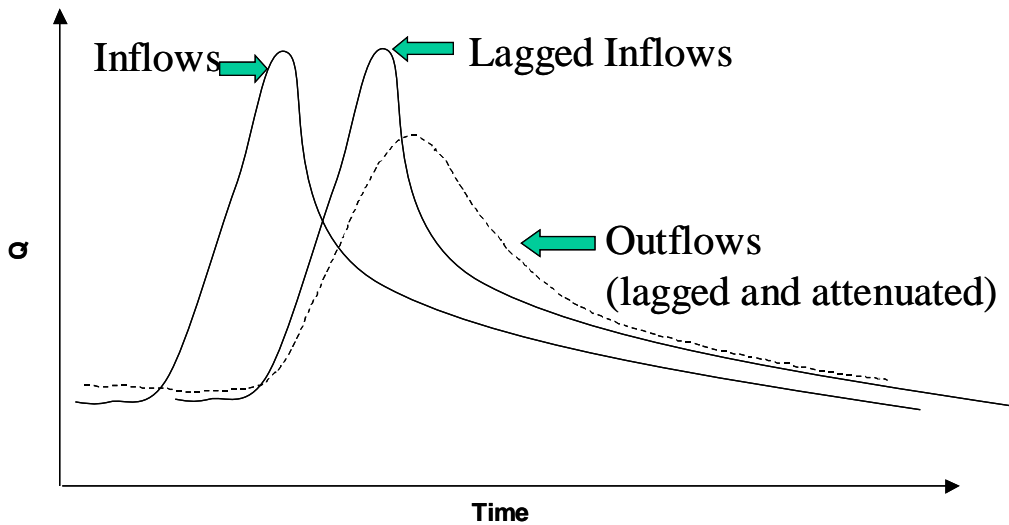


Figure 1: Lag and K Routing

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:

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

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() Command

Lag and attenuate a time series, creating a new time series using variable Lag and K technique.
 The time series to be routed cannot contain missing values.
 See the documentation for a complete description of the algorithm.
The input and output state parameters are currently ignored - states default to zero.

Time series to lag (TSID):

New time series ID: Specify to avoid confusion with TSID from original TS.

Flow units: Required - units of Lag and K flow values, compatible with time series.

Lag interval: Required - Lag and K interval units.

Lag: Optional - flow, Lag; flow, Lag pairs.

K: Optional - flow, K; flow, K pairs.

Input time series states: Optional - separate values by commas (default=0 for all).

Output time series states: Optional - separate values by commas (default=0 for all).

Time series alias: Optional - alternate ID, (e.g., location from the TSID).

Command:

VariableLagK

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 <code>InflowStates</code> 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
TS Inflow = ReadNwsCard(InputFile="Data\3HR_INPUT.SQIN")
#
# 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
#   24.0   200.0   12.0  600.00   9.0  1500.0   42.0  3000.0
#
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 09.06.03, 2010-04-05

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 from 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.

Edit WebGet() command

This command retrieves content from the web using a Uniform Resource Identifier (URI) and saves the content to a local file. 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: `ADATUNCO.txt` Browse

Command: `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="ADATUNCO.txt")`

Add Working Directory to Local File Cancel OK

WebGet

WebGet() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
URI	The Uniform Resource Identifier (URI) for the content to be retrieved. This is often also referred to as the Uniform Resource Locator (URL).	None – must be specified.
LocalFile	The local file in which to save the content.	None – must be specified.

Command Reference: TS Alias = WeightTraces()

Create a time series by weighting data from time series ensemble traces

Version 08.15.00, 2008-04-24,

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 by weighting traces from an ensemble. The result is identified by its alias.
Enter trace years and weights (0.0 to 1.0), one value per line.
Any trace value that is missing will cause the weighted result to be missing.

Time series alias:

Ensemble to process:

Specify weights how?:

Trace years:

Weights:

New time series ID parts: Specify to avoid confusion with TSID from original TS.

Command:

TS_weightTraces

WeightTraces() Command Editor

The command syntax is as follows:

```
TS Alias = WeightTraces(Parameter=Value,...)
```

Command Parameters

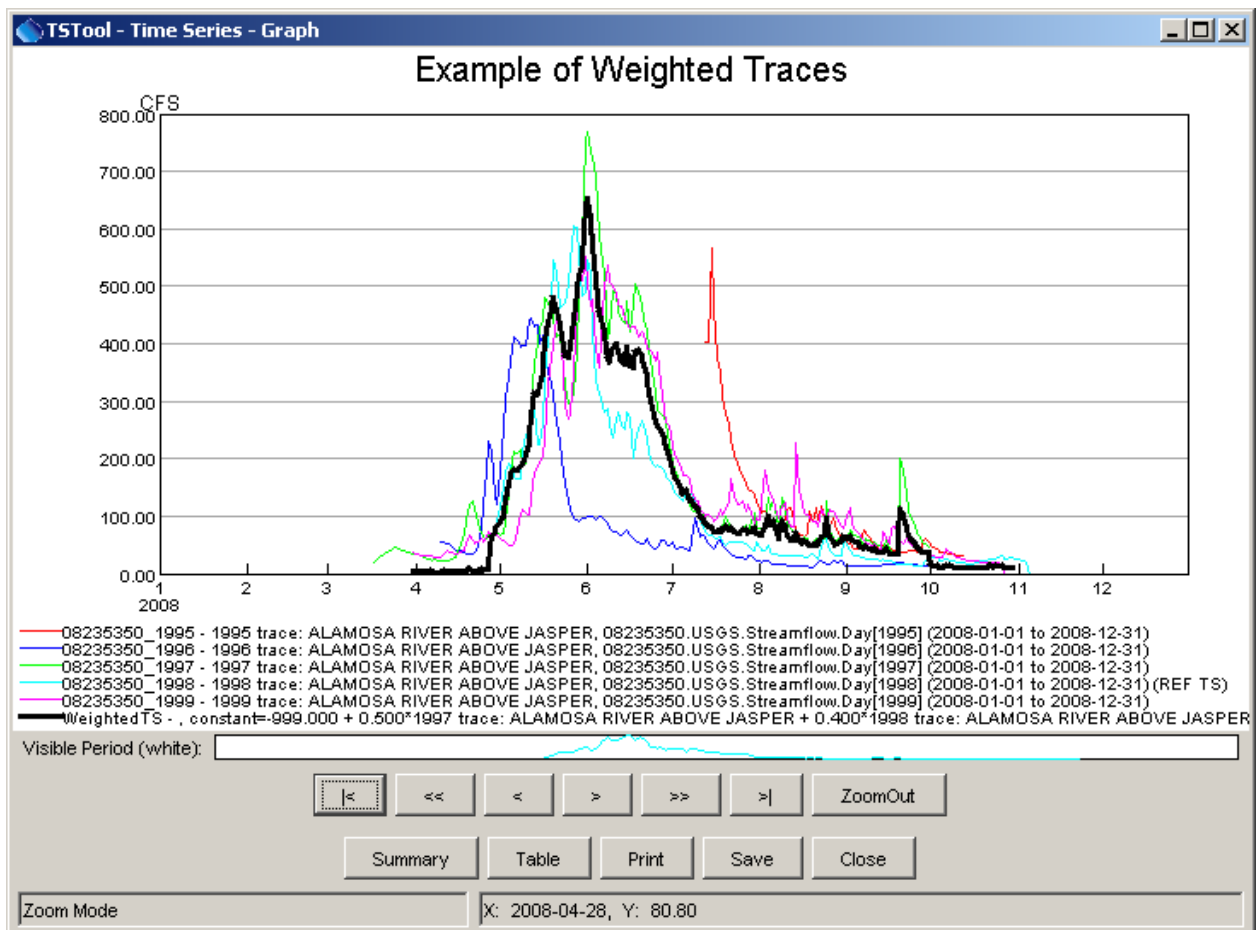
Parameter	Description	Default
Alias	The alias of the new time series.	None – must be specified.
EnsembleID	The ensemble identifier indicating time series to be processed (e.g., from a CreateEnsemble() command). Time series matching the years specified by the Weights parameter will be processed.	None – must be specified.
SpecifyWeightsHow	Weights are currently only applied as AbsoluteWeights (in the future an option may be added to normalized weights to 1.0 accounting for missing data in the traces).	Must be AbsoluteWeights.
Weights	Specify pairs of trace year and weights (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	None – must be specified.
NewTSID	The time series identifier for the new time series that is created. This typically uses the same information as the original time series, with an added scenario.	None – must be specified.

A sample commands file is as follows (longer commands that wrap are shown indented):

```
# Create annual traces from a time series shifted to the current year
# The original time series is read from HydroBase
#
# (1995-1998) ALAMOSA RIVER ABOVE JASPER, CO USGS Streamflow Day
08235350.USGS.Streamflow.Day~HydroBase
CreateEnsemble(TSID="08235350.USGS.Streamflow.Day",TraceLength=1Year,
  EnsembleID="Ensemble_Jasper",
  EnsembleName="ALAMOSA RIVER ABOVE JASPER, CO",
  ReferenceDate="2008-01-01",ShiftDataHow=ShiftToReference)
TS WeightedTS = WeightTraces(EnsembleID="Ensemble_Jasper",
  SpecifyWeightsHow="AbsoluteWeights",
  Weights="1997,.5,1998,.4,1999,.1",
  NewTSID="08235350.USGS.Streamflow.Day.weighted")
WriteDateValue(OutputFile="Results/WeightTraces_out.dv")
```

UserManualExamples/TestCases/CommandReference/WeightTraces/WeightTraces.TSTool

The results from the commands are shown in the following graph:



WeighTraces_Graph

Results of the WeightTraces() Command

This page is intentionally blank.

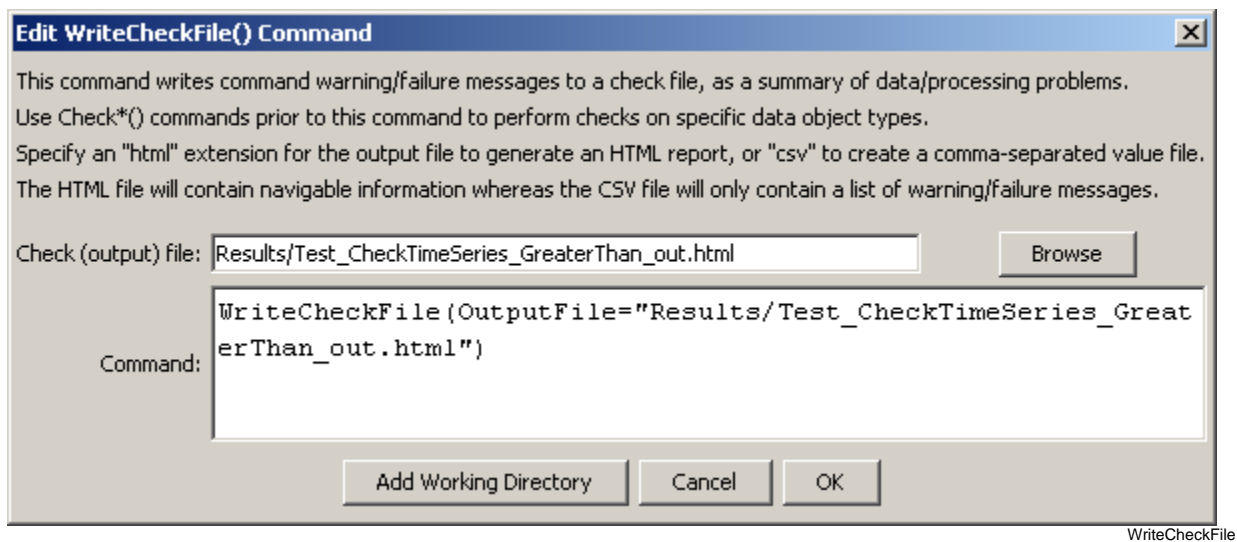
Command Reference: WriteCheckFile()

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.



WriteCheckFile() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
OutputFile	<p>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.</p> <p>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.</p>	None – must be specified.

This page is intentionally blank.

Command Reference: WriteDateValue()

Write time series to a DateValue format file

Version 08.18.02, 2008-11-18

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.

WriteDateValue() Command Editor

The command syntax is as follows:

WriteDateValue (Parameter=Value,...)

Command Parameters

Parameter	Description	Default
OutputFile	The DateValue output file. The path to the file can be absolute or relative to the working directory (command file location).	None – must be specified.
Delimiter	The delimiter character to use between data values. Comma is the only other allowed value other than the default space.	Space.
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

Parameter	Description	Default
		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.
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none"> • 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. 	AllTS
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 modified, if processing an ensemble.	Required if TSList=EnsembleID.

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

```
# 0100501 - EMPIRE DITCH
0100501.DWR.DivTotal.Month~HydroBase
# 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 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\WriteHecDss
The Browse button can be used to select an existing file to update (or edit the file name after selection to specify a new file).
Specifying a file that does not exist will create a new file.
Enter date/times to a precision appropriate for output time series.
HEC-DSS parts will be taken from the time series identifier as follows: Apart:Bpart.*.Cpart.Epart.Fpart
The defaults can be overridden with the optional part fields.

HEC-DSS file to write:

Type: Required - HEC-DSS time series type.

TS list: Optional - indicates the time series to process (default=AllTS).

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).

B: Optional - B part to write (default=second : part of TSID location).

C: 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?: ☐ Optional - close HEC-DSS file after write (default=False).

Command:

```
WriteHecDss (OutputFile="Results/Test_WriteHecDss_Day_out.dss", Type=PER-AVER)
```

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, surrounded by double quotes to protect whitespace and special characters. If the file does not exist it will be created.	None – must be specified.
Type	The HEC-DSS time series type, indicating whether the time series is instantaneous, mean, or accumulated.	None – must be specified.
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none"> AllMatchingTSID – all time series that match the TSID (single TSID or TSID with wildcards) will be processed. 	AllTS

Parameter	Description	Default
	<ul style="list-style-type: none"> 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 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 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 numerical output.	HEC-DSS default.
A	The DSS path A-part to use for the time series as written to the HEC-DSS file.	Time series identifier location part before the : (if : is present) or the entire location if : is not present.
B	The DSS path B-part to use for the time series as written to the HEC-DSS file.	Time series identifier location part after the : (if : is present) or the blank if : is not present.
C	The DSS path C-part to use for the time series as written to the HEC-DSS file.	Time series identifier data type.
E	The DSS path E-part to use for the time series as written to the HEC-DSS file.	Time series identifier data interval, converted to HEC-DSS conventions.
F	The DSS path F-part to use for the time series as written to the HEC-DSS file.	Time series identifier scenario.
Replace	Under development – whether to replace the contents of the previous time series in the HEC-DSS file.	Only replace what is actually written.
Close	Indicate whether to close connections to the HEC-DSS file and allow other processes to move/rename/delete the file. Specifying as True may slow the software as files are repeatedly opened and closed.	False – let the HEC-DSS internal software close the connection after timing out.

A sample command file is as follows:

```
WriteHecDss(OutputFile="sample.dss",TYPE=PER-AVER,  
            OutputStart="1992-01-01",OutputEnd="1992-12-31")
```

Command Reference: WriteProperty()

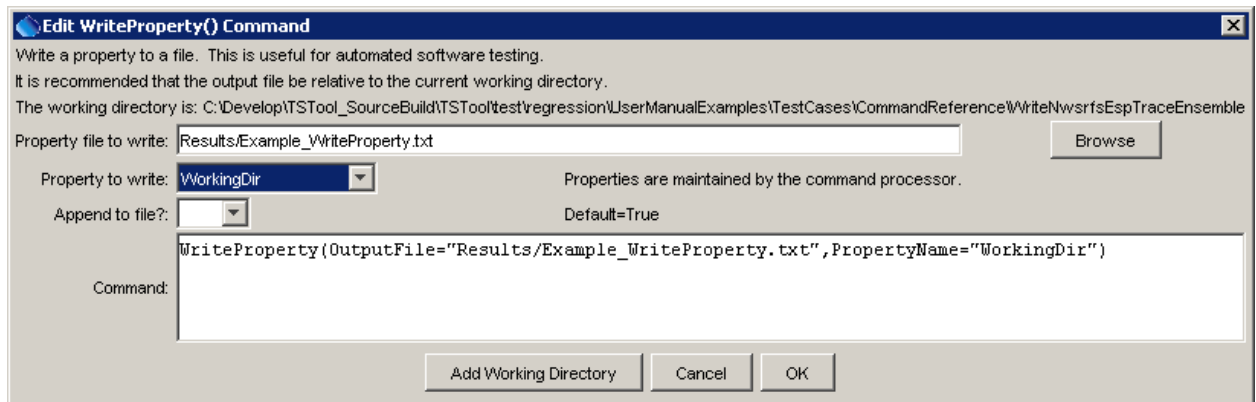
Write a time series processor property to a file

Version 08.15.00, 2008-05-11

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.



WriteProperty

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: 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 RiverWare 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\WriteRiverWare
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 "AllMatchingTSID", pick a single time series, or enter a wildcard time series identifier pattern.
Only the first time series will be written (limitation of file format).

TS list: How to get the time series to write (default=AllTS).

Identifier (TSID) to match:

RiverWare file to write:

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: Number of digits after decimal (default=4).

Command:

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: <ul style="list-style-type: none"> 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 <code>SelectTimeSeries()</code>). 	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:

Output start: Overrides the global output start, specify as YYYY.

Output end: Overrides the global output end, specify as YYYY.

Command:

WriteStateCU

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 path to the file can be absolute or relative to the working directory.	None – must be specified.
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 WriteStateMod() Command

Write time series to a StateMod format file.

It is recommended that the file name be relative to the working directory.

The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\WriteStateMod

The Browse button can be used to select an existing file to overwrite (or edit the file name after selection).

For the precision, a negative integer allows auto-adjustment to prevent overflow.

A precision of -2001 will default to 2 digits, adjusted for overflow, and also use no decimal (special precision option).

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: **AllTS** 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).

Command: `WriteStateMod(TSList=AllTS,OutputFile="RioGrande.rih")`

Add Working Directory **Cancel** **OK**

WriteStateMod

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 series to process, one of: <ul style="list-style-type: none"> 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 <code>SelectTimeSeries()</code>). 	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 StateMod file to write. The path to the file can be absolute or relative to the working directory (command file location).	None – must be specified.
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 decimal point, for data values. A negative number indicates that if the formatted number is larger than the allowed output 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.	The default output precision if not specified is -2, which is then reset based on the data units (see the <i>system\DATAUNIT</i> file).

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 HTML. 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\TSTool_SourceBuild\TSTool\test\regression\UserManualExamples\TestCases\CommandReference\WriteSummary
Specify the file extension as "html" to write an HTML file (default is text).

Summary file to write:

TS list: Optional - indicates the time series to process (default=AllTS).

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).

Command:

```
WriteSummary(TSList=AllTS, OutputFile="RioGrandeStreamflow.txt")
```

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" extension will result in HTML output, which is color-coded for missing values and has notes for flagged values.	None – must be specified.
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none"> 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 <code>SelectTimeSeries()</code> command. 	AllTS
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: WriteTableToDelimitedFile()

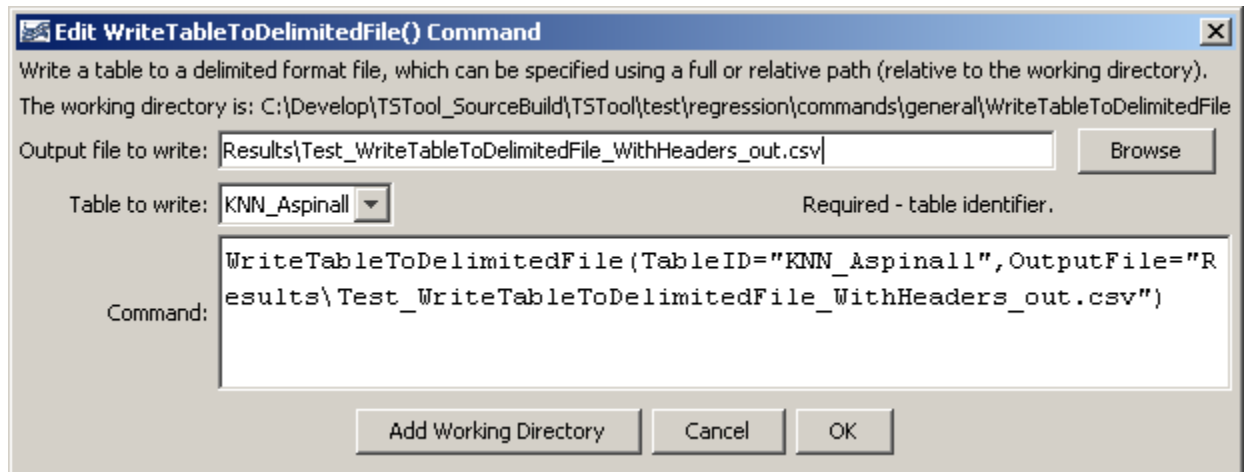
Write a table to a delimited file

Version 09.04.00, 2009-06-16

The `WriteTableToDelimitedFile()` command writes a table to a comma-delimited file. This command is the analog to the `ReadTableFromDelimitedFile()` command. It can be used to provide tabular data to other programs, such as spreadsheet programs.

A standard file header is written with 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 precision of floating point numbers may be inappropriate – this will be addressed in future updates.

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



WriteTableToDelimitedFile

WriteTableToDelimitedFile() Command Editor

The command syntax is as follows:

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

Command Parameters

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

Version 08.16.03, 2008-08-18

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 WriteTimeSeriesProperty() 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 be relative to the current working directory.
The working directory is: C:\Develop\TSTool_SourceBuild\TSTool\test\regression\commands\general\WriteTimeSeriesProperty

TS list: Indicates the time series to process (default=AllTS).

TSID (for TSList=AllMatchingTSID):

EnsembleID (for TSList=EnsembleID):

Property file to write: Results/Test_WriteTimeSeriesProperty_PropertyName=DataLimitsOriginal_out.txt

Property to write: DataLimitsOriginal Properties are maintained by the command processor.

Append to file?: ☐ Default=True

Command:
`WriteTimeSeriesProperty(OutputFile="Results/Test_WriteTimeSeriesProperty_PropertyNameDataLimitsOriginal_out.txt",PropertyName="DataLimitsOriginal",Append=False)`

WriteTimeSeriesProperty

WriteTimeSeriesProperty() Command Editor

The command syntax is as follows:

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

Command Parameters

Parameter	Description	Default
TSList	Indicates the list of time series to be processed, one of: <ul style="list-style-type: none"> 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 <code>SelectTimeSeries()</code> 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 when TSList=*TSID
EnsembleID	The ensemble to be modified, if processing an ensemble.	Required when TSList=EnsembleID.
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")
```

Appendix: TSTool Installation and Configuration for CDSS

CDSS Version, 09.08.00, 2010-09-08

1. Overview

This appendix describes how to install TSTool in the CDSS (Colorado's Decision Support Systems) environment. CDSS consists of the HydroBase database, modeling, and data viewing/editing software. TSTool can be used within this system to process time series from the HydroBase database, CDSS model files, and other databases and files.

2. File Locations

Standard locations of TSTool software files are as follows. Files are normally installed on Windows on the C: drive but can be installed in a shared location on a server.

<i>C: \CDSS\TSTool-Version</i>	Top-level install directory.
<i>bin\</i>	Software program files directory.
<i>batik*.jar</i>	Scalable Vector Graphics (SVG) output packages.
<i>Blowfish*.jar</i>	Used for encryption/security.
<i>cdss.domain*.jar</i>	CDSS components.
<i>h2*.jar</i>	H2 embedded database.
<i>HydroBaseDMI*.jar</i>	State of Colorado HydroBase database interface package.
<i>jcommon.jar, jfreechart.jar</i>	Plotting package.
<i>jsr173_1.0_api.jar, libXMLJava.jar</i>	XML support.
<i>jython.jar</i>	Jython support.
<i>msbase.jar</i>	Microsoft SQL Server packages.
<i>mssqlserver.jar</i>	
<i>msutil.jar</i>	
<i>NWSRFS_DMI*.jar</i>	National Weather Service River Forecast System (NWSRFS) package.
<i>RiversideDB_DMI*.jar</i>	Riverside Technology, inc., RiversideDB database package.
<i>RTi_Common*.jar</i>	Riverside Technology, inc. supporting packages.
<i>SatmonSysDMI*.jar</i>	State of Colorado Satellite Monitoring System package.
<i>shellcon.exe</i>	Executable program used to read from the Windows registry (e.g., to determine the default web browser and list available ODBC data source names). – PHASING OUT.
<i>StateMod*.jar</i>	State of Colorado's StateMod and StateCU model packages.

<i>TSCmdProcessor*.jar</i>	Time series command processor package.
<i>tstool</i>	Shell script to run TSTool on Linux.
<i>TSTool.bat</i>	Batch file to run TSTool using the JRE software. This may need to be edited if the installation is not standard.
<i>TSTool.exe</i>	Executable program to run TSTool using the JRE software, recommended over batch file.
<i>TSTool.l4j.ini</i>	Configuration file for <i>TSTool.exe</i> launcher.
<i>TSTool*.jar</i>	TSTool program components.
<i>doc\TSTool\UserManual\</i>	Main documentation directory for TSTool.
<i>TSTool.pdf</i>	TSTool documentation as PDF.
<i>examples\</i>	Example data and command files.
<i>logs\</i>	Directory for TSTool log files (should be writable).
<i>system\</i>	Directory for system files.
<i>CDSS.cfg</i>	CDSS configuration file for HydroBase database configuration.
<i>DATAUNIT</i>	Data units file.
<i>TSTool.cfg</i>	Configuration file to modify TSTool defaults.
<i>jre*\</i>	Java Runtime Environment used by TSTool

3. Installing TSTool

TSTool can be installed either as part of the HydroBase Tools CD/DVD installation, or as a separate installation. In both cases, is recommended that the normal CDSS file structure be used.

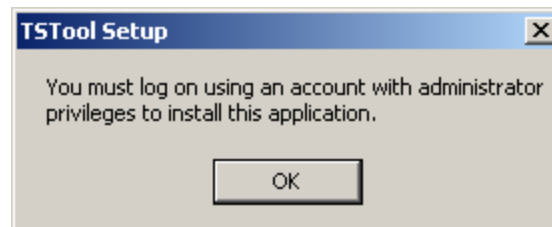
3.1 Installing TSTool from the “HydroBase data set Analysis Query Tools CD/DVD”

If you have purchased a HydroBase CD/DVD, TSTool will be installed during the CD install process. Refer to the installation instructions for that distribution. The version that is installed may be older than the version available on the CDSS web site; however, multiple versions can be installed and run independently.

3.2 Installing TSTool from the TSTool Setup File

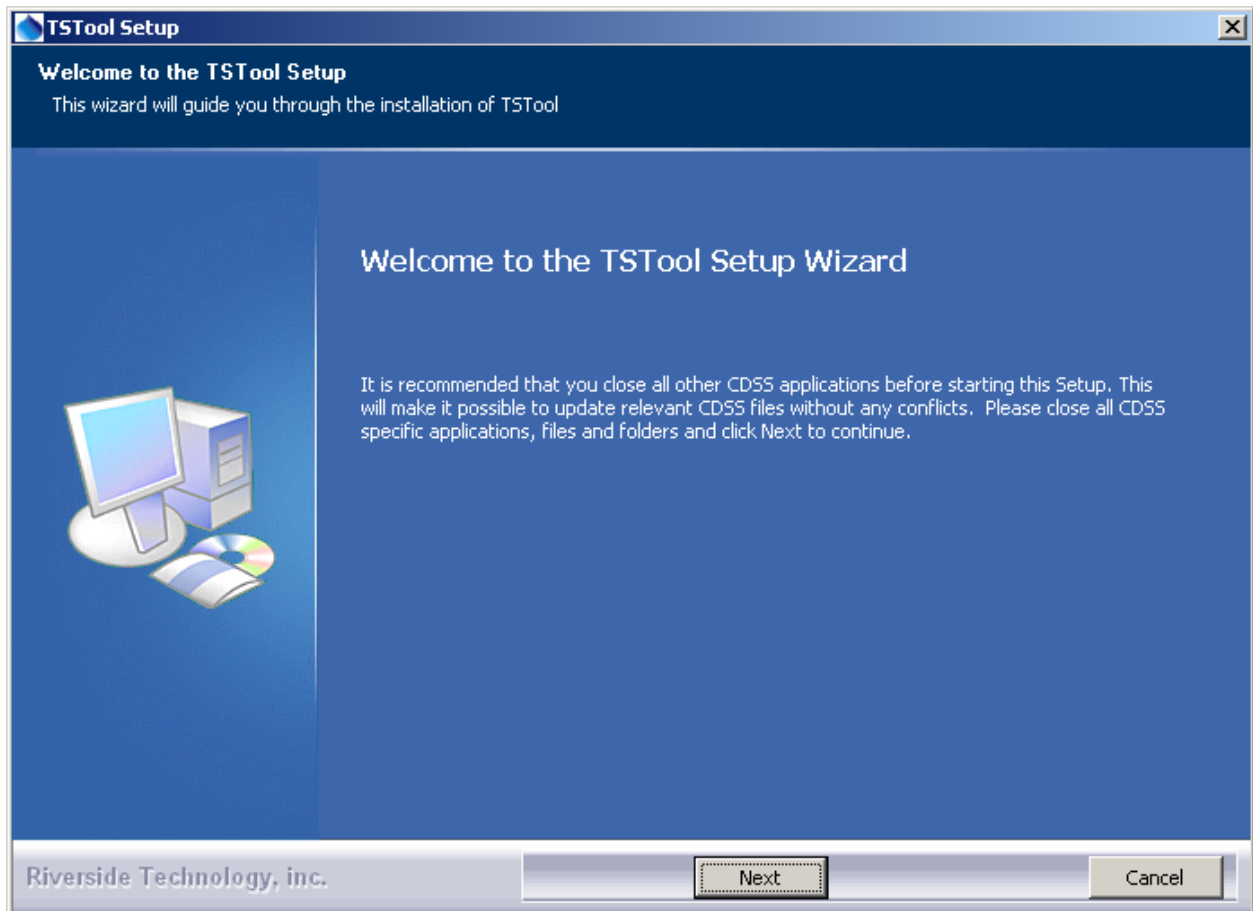
Use the following instructions to install TSTool using the *TSTool_CDSS_Version_Setup.exe* installer program, for example if TSTool software was downloaded from the CDSS web site (<http://cdss.state.co.us>):

1. Run the *TSTool_CDSS_Version_Setup.exe* file by selecting from Windows Explorer, the **Start... Run...** menu, or from a command shell. You must be logged into the computer using an account with administrator privileges. Otherwise, the following warning will be displayed:



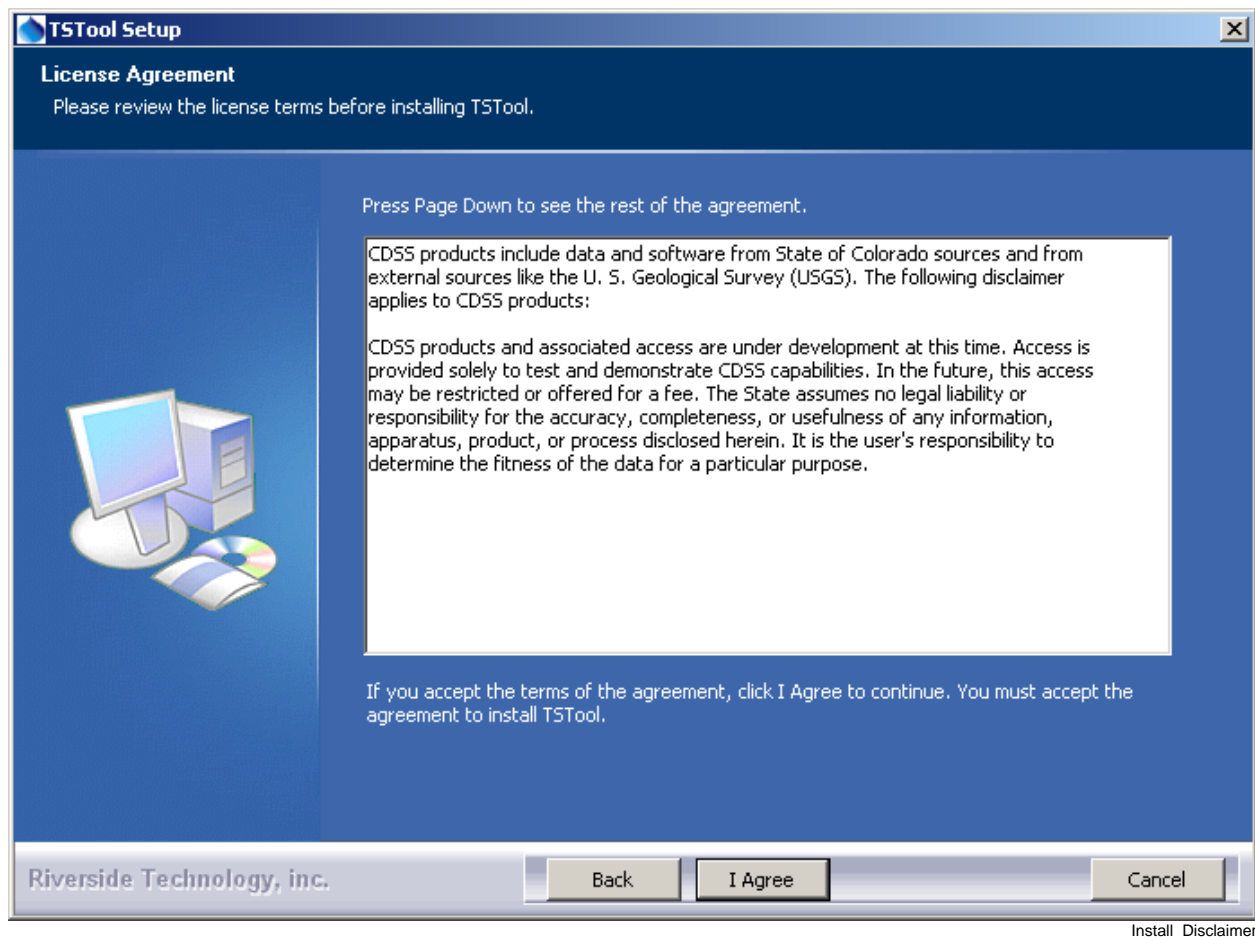
Install_AdministratorWarning

If you have administrative privileges, the following welcome will be displayed, and the installation can continue:



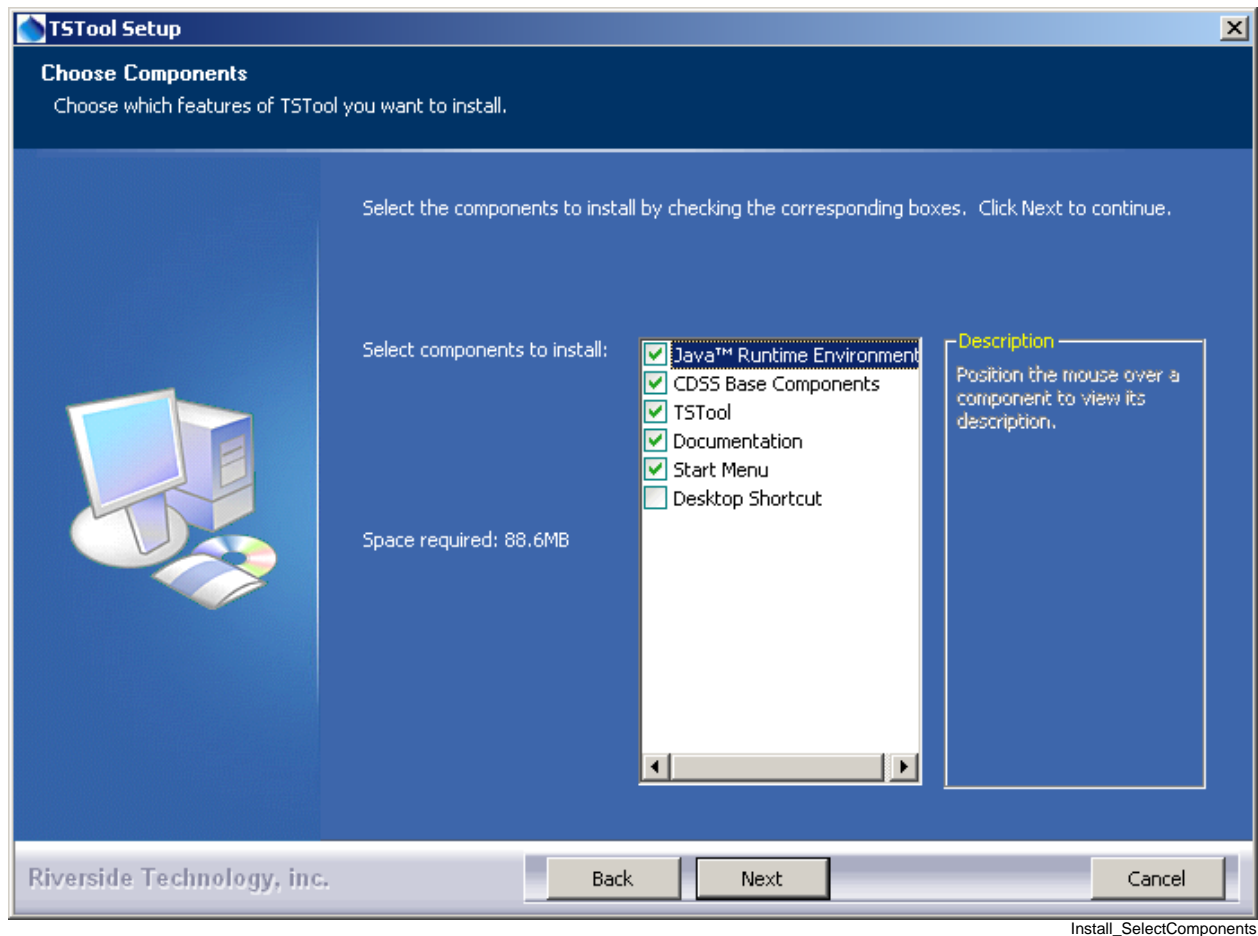
Install_Welcome

Press **Next** to continue with the installation.



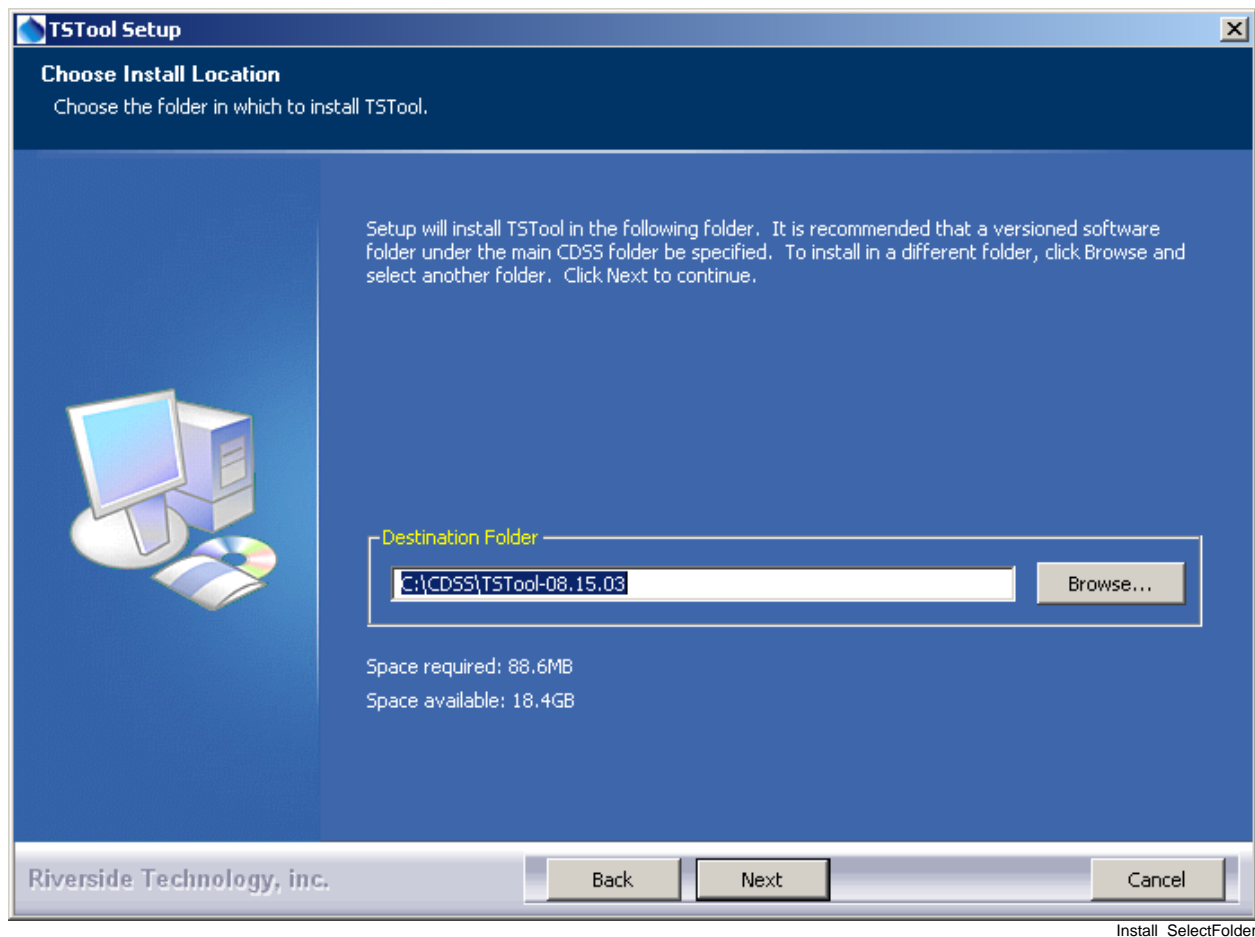
TSTool is distributed with CDSS with no license restrictions. However the disclaimer must be acknowledged. Press **I Agree** to continue with the installation.

2. Several components can be selected for the install as shown in the following dialog. Position the mouse over a component to see its description.



Select the components to install and press **Next**.

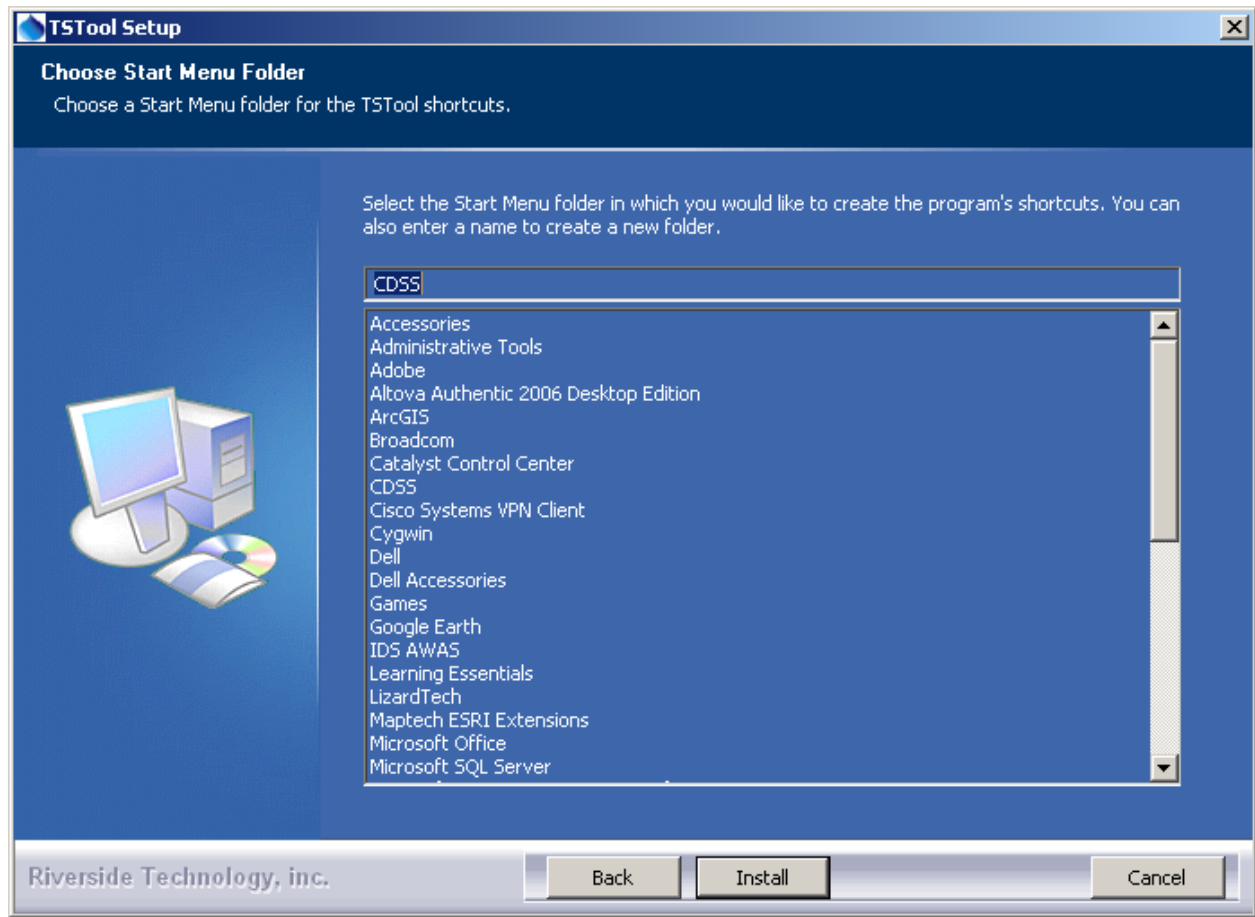
3. The following dialog is then shown and is used to select the installation location for TSTool. Multiple versions of TSTool can be installed and there are no dependencies between the versions. It is recommended that the default install location shown is used.



After selecting the install location, press **Next**.

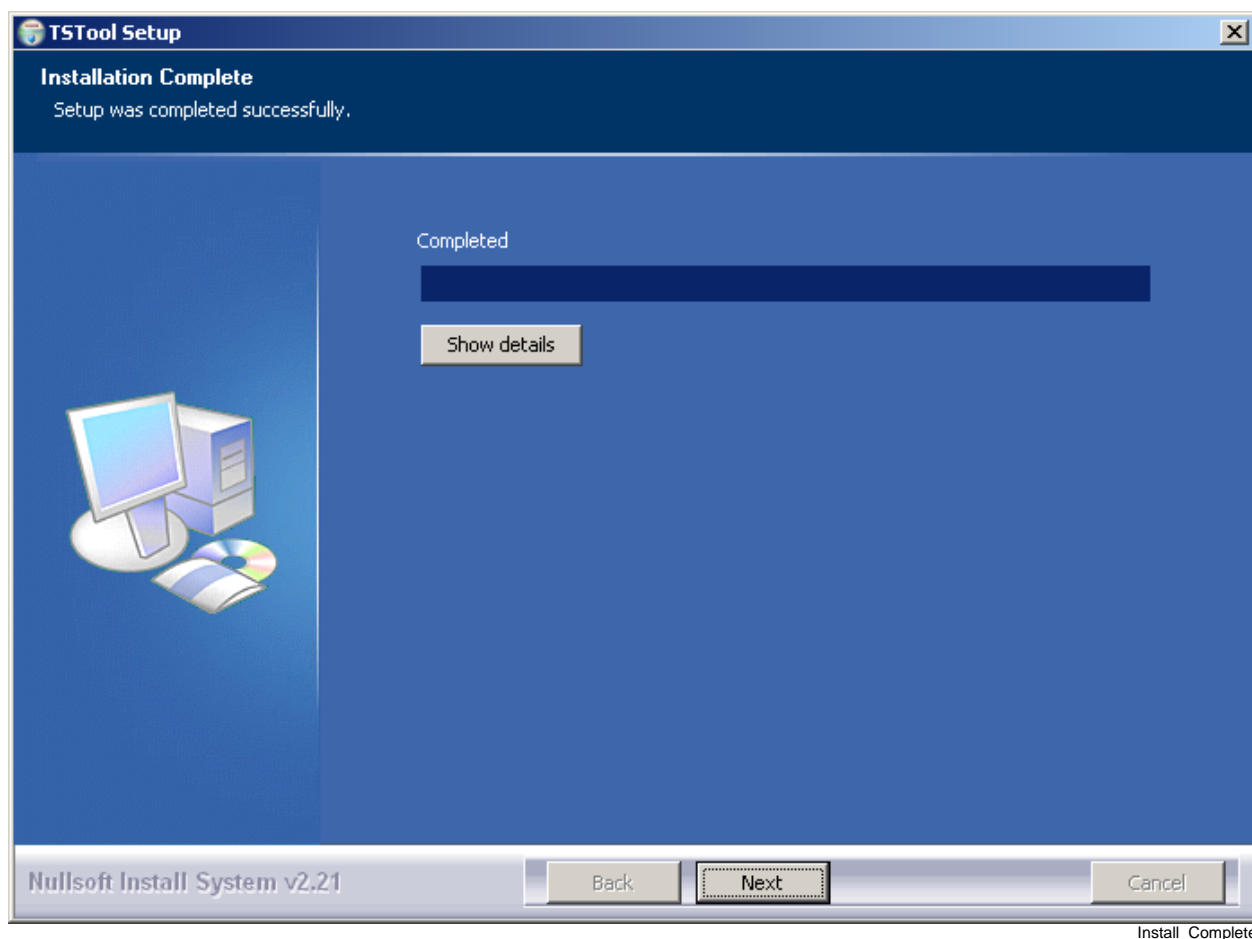
Note that this location will be saved as a Windows registry setting (*HKEY_LOCAL_MACHINE\Software\State of Colorado\TSTool-Version\Path*) to allow future updates to check for and default to the same install location, and to allow the standard software uninstall procedure to work correctly.

4. The following dialog will be shown to select the menu for the software:



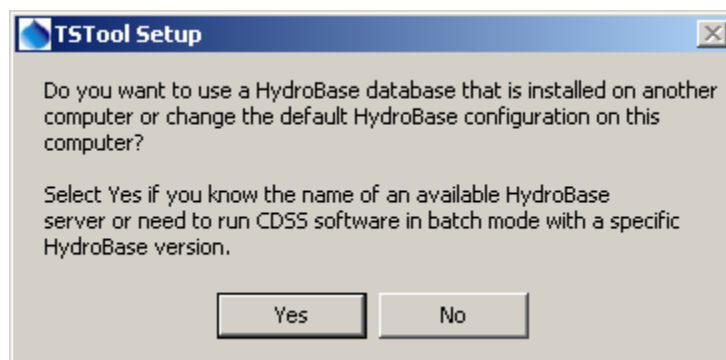
After selecting the folder, press ***Install***.

5. The following dialog will show the progress of the installation:



Press Show details to see the files that were installed or press **Next** to continue.

6. If the CDSS Base Components were selected for install, the following dialog will be displayed:

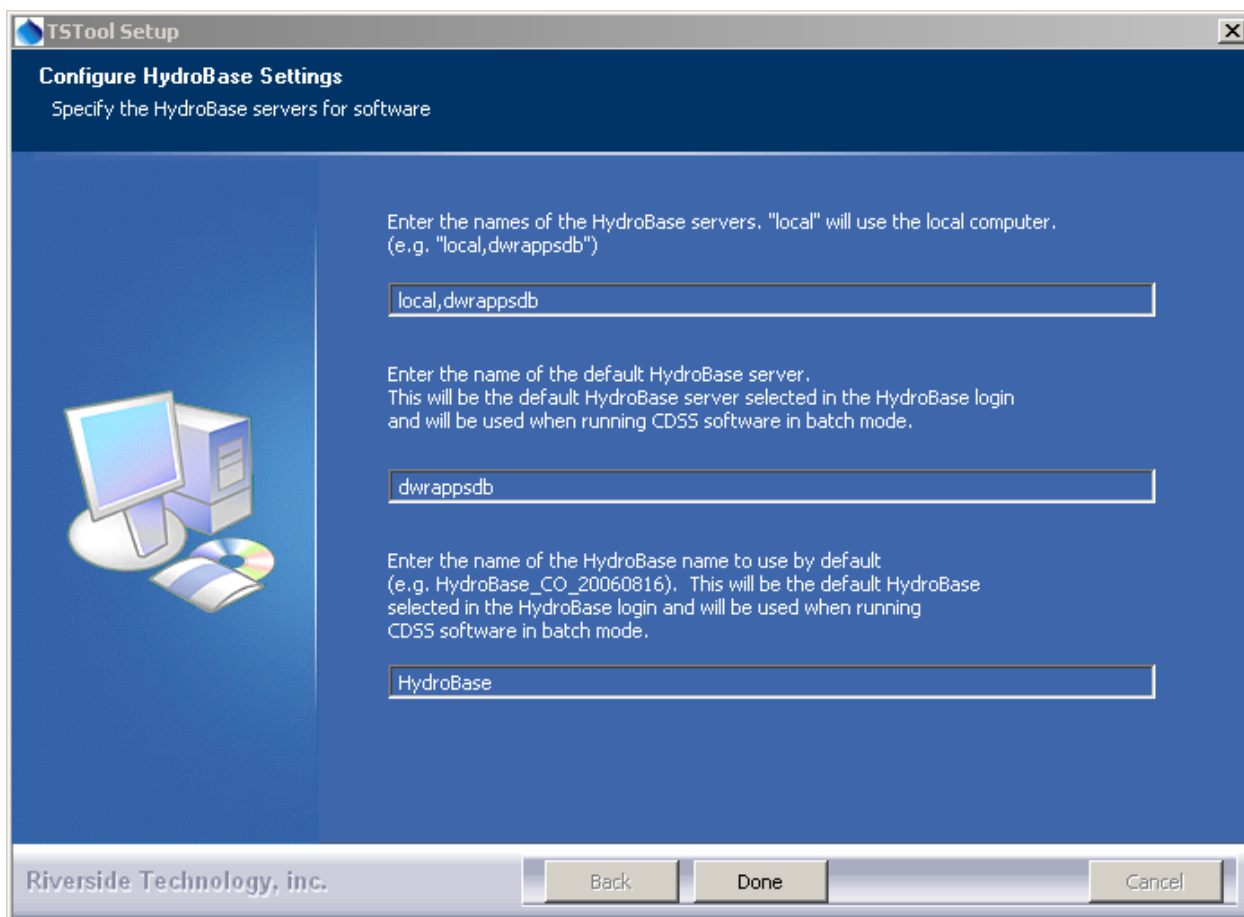


Install_HydroBaseQuestion

TSTool and other CDSS software can utilize HydroBase running on the local computer as well as other computers. Press **Yes** if HydroBase has been installed on another computer in the network environment and may be used by the software (then continue to the next step). Also press **Yes** if

TSTool will be run in batch mode because the specific HydroBase name must be specified in configuration files. Otherwise, press **No** (skip to step 8).

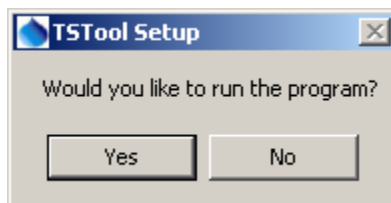
- The following dialog allows additional HydroBase servers to be specified for use by CDSS software (the example below configures CDSS software to list the dwrappsdb HydroBase server in choices and defaults to HydroBase on the local computer). The dialog will initially show previous settings from the `\CDSS\TSTool-Version\system\CDSS.cfg` file and settings typically only need to be changed after installing a new HydroBase version.



Install_HydroBaseConfiguration

After entering the name of a HydroBase server and the default server to use, press **Done**.

- The following dialog will then be shown asking whether the TSTool software should be run:



Install_RunTSToolQuestion

Press **Yes** to run the software or **No** to exit the installation procedure.

9. TSTool is distributed with a default configuration for CDSS. If you have edited the configuration properties, you can import the old configuration file using the **Help...Import Configuration...** menu'.

3.3 Installing TSTool on a File Server

TSTool can be installed on a file server, which allows software updates to be made in one location, thereby eliminating the need to install software on individual machines. For this type of installation, all computers that access the software should typically have similar configuration, including network configuration. The standard installer described in this documentation focuses on individual installs on user computers. To make TSTool software installed on a server available to other computers, perform the following (this is typically performed by system administrators):

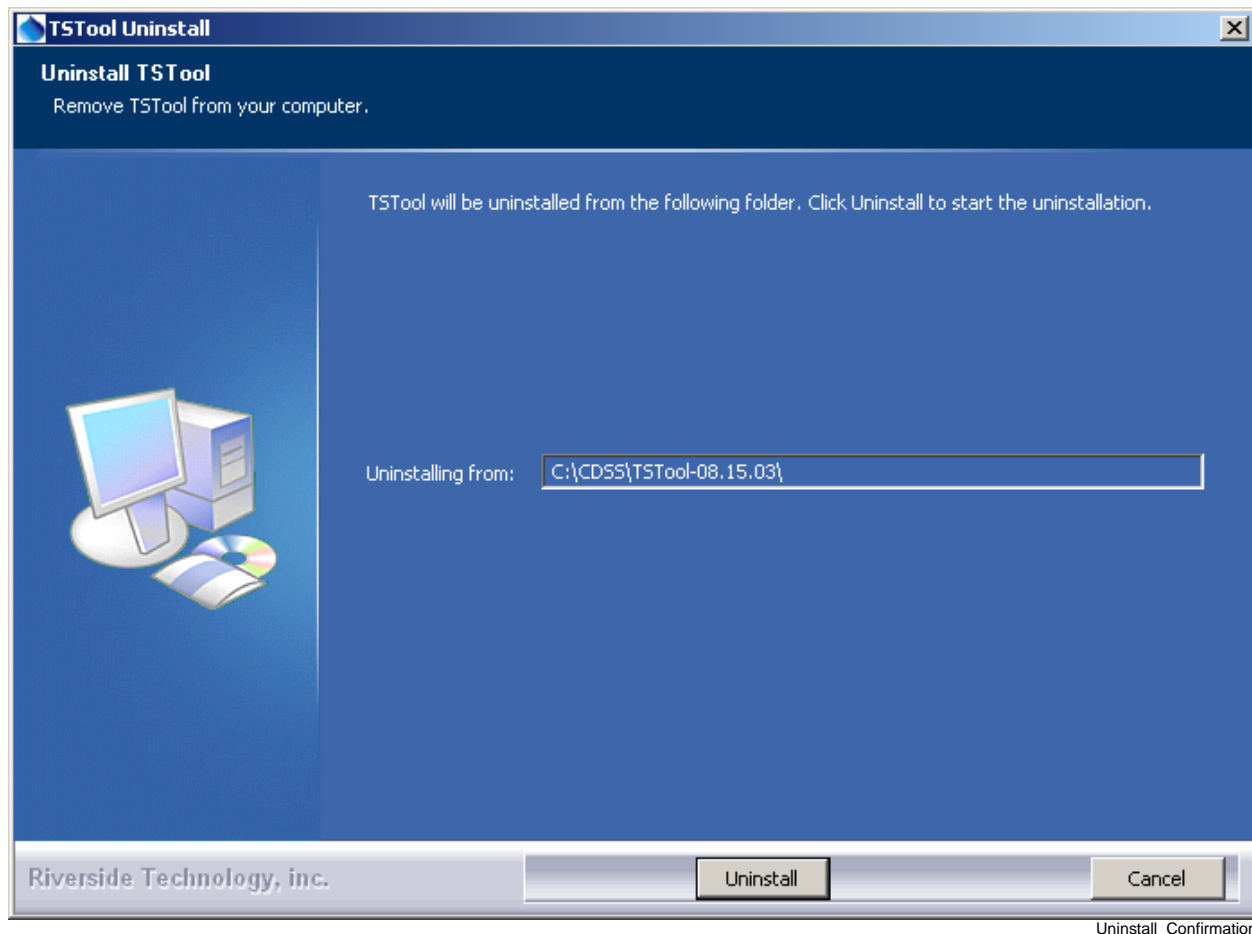
1. Run the *TSTool_CDSS_Version_Setup.exe* installer as described above. During installation specify the TSTool installation home using a drive letter and path for the server or specify a Universal Naming Convention (UNC) path (e.g., `\\ServerName\CDSS\TSTool-Version`).
2. Or....Copy the files from a local installation to a network location. The TSTool software will detect the file location when run using the *TSTool.exe* file. If the *TSTool.bat* file is used to run the software, it may need to be modified to specify the location of files on the server.

The menus and shortcuts will only be configured for the computer from which the installation was run. Therefore, menus and shortcuts for other computers will need to be manually configured.

If TSTool has been installed on a local computer and it is also available on the network, the network version can be run by running the software in the `ServerName\CDSS\TSTool-Version\bin` folder. The software will expect that file locations use the same drives as when the software was installed.

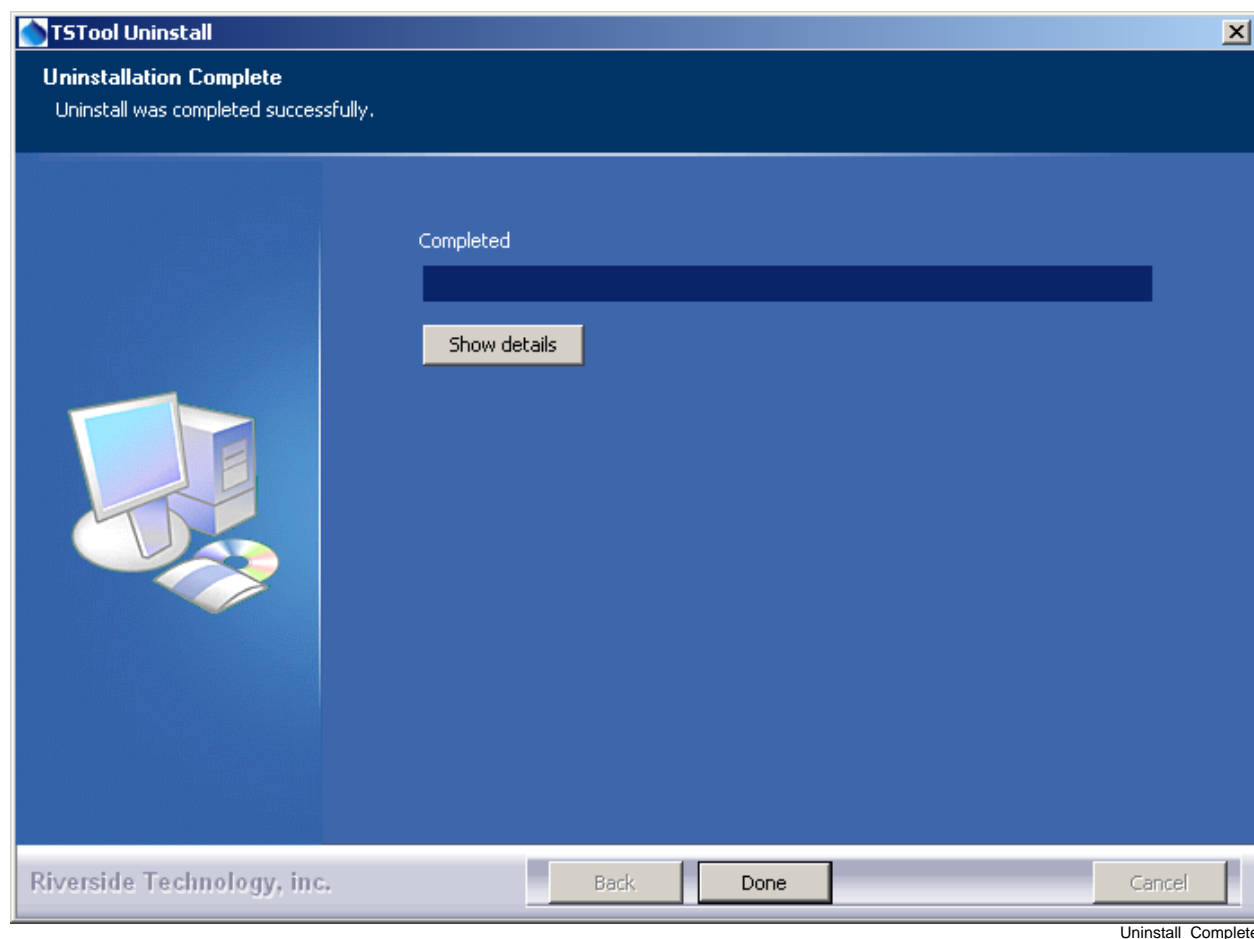
4. Uninstalling TSTool Software

To uninstall TSTool software, select the **CDSS...Uninstall...TSTool** from the **Start** menu and confirm the uninstall. CDSS components that are used by other software (e.g., CDSS Base component software) as well as user data will remain installed.



Press **Uninstall** to uninstall the software.

The following dialog shows the status of the uninstall.



Press **Show details** to see the list of files that were removed. Press **Done** to exit the uninstall.

5. Running TSTool

TSTool can be started in several ways as described below.

5.1 CDSS Menu

The **Start...All Programs...CDSS...TSTool-Version** (or **Start... Programs... CDSS... TSTool-Version**) menu can be used to start the software. This runs the `TSToolInstallHome\bin\TSTool.exe` software.

5.2 Command Line Executable

The installation process does NOT add the `TSToolInstallHome\bin` folder to the path; however, this addition can be made by the user, allowing the TSTool software to be started anywhere by running `TSTool`. Running TSTool from any location will result in the software being run in the installation location. Specifying a command file on the command line or interactively will reset the working directory to that of the command file.

5.3 TSTool Batch File – Windows

A batch file can be used to run the *TSTool.exe* program, for example using the `-commands` command line parameter to specify a command file. In this case it may be necessary to specify the absolute path to the command file to ensure that the software can locate related files.

6. TSTool Configuration

TSTool requires minimal configuration after installation. This section describes TSTool configuration files that can be customized for a system. Configuration is specified for each TSTool installation. Installations on a server use one configuration for all users.

6.1 TSTool Configuration File

The *system\TSTool.cfg* file under the main installation directory contains top-level configuration information for TSTool. The format of the file is as follows:

```
#
# Configuration file for TSTool

[TSTool]

ColoradoSMSEnabled = true
DateValueEnabled = true
HydroBaseEnabled = true
RiverWareEnabled = true
StateCUEnabled = true
StateModEnabled = true

MapLayerLookupFile = "\cdss\gis\co\TimeSeriesMapLookup.csv"

LicenseOwner = "CDSS"
LicenseType = CDSS
LicenseCount = NoLimit
LicenseExpires = Never
LicenseKey = 00-77960bdfb1dde707-1dd052fe0327a332-a07266ee645e8845-7560192d374235c5-1dd052fe0327a332
```

Example TSTool Configuration File

The example illustrates the format of the file. The `*Enabled` properties can be used to enable/disable input types. Common formats are enabled by default and more specialized formats are disabled by default, if not specified in the file. For example, use `HydroBaseEnabled = false` to disable the automatic HydroBase login that occurs with the HydroBase input type (e.g., if HydroBase is unavailable for some reason). The license properties are assigned by developers and should not normally be changed by TSTool users. Each input type can have additional properties, although only a few currently do, as described below.

The optional `MapLayerLookupFile` property indicates the name of the time series to map layer lookup file. See the **Map Configuration** section below.

6.2 Data Units File

The *system\DATAUNIT* file under the main installation directory contains data unit information that defines conversions and output precision. In most cases the default file can be used but additional units may need to be added for a user's needs (in this case please notify the developers so the units can be added to the default file distributed with installations). Currently, the *DATAUNIT* file is the only source for units information – in the future units may be determined from the various input sources.

6.3 HydroBase Configuration

The following properties can be defined in the *TSTool.cfg* file in a [HydroBase] section to control how TSTool interacts with HydroBase. See also the **CDSS Configuration File** section below.

TSTool HydroBase Configuration Properties

Property	Description	Default
AutoConnect	If False, a HydroBase login dialog will be shown at startup. If True, the default database information in the CDSS configuration file (see next section) will be used to automatically connect to the database, and the login dialog will not be shown.	False
WDIDLength	Indicates the length of water district identifiers (WDIDs) constructed from separate WD and ID data, when creating time series identifiers. Because time series identifier strings are compared literally, it is important that the WDIDs are consistent within a commands file.	7

6.4 CDSS Configuration File

By default, TSTool will automatically look for HydroBase databases on the current (local) machine and the State servers. State server databases are typically only accessible to State of Colorado computers. If SQL Server or MSDE HydroBase versions have been installed on a different machine, the *\cdss\TSTool-Version\system\CDSS.cfg* file can be used to indicate the database servers. An example of the configuration file is as follows:

```
[HydroBase]

ServerNames="ServerName,local"
DefaultServerName="ServerName"
DefaultDatabaseName="HydroBase_CO_20080730"

[ColoradoSMS]

ServerNames="ServerName,local"
DefaultServerName="ServerName"
DefaultDatabaseName="RealtimeStreamflow"
UserLogin="UserLogin"
```

The ColoradoSMS input type is being used to support annotation of real-time data graphs with alert information, within the State of Colorado's offices.

Properties can be specified on the TSTool command line using the notation “Property=Value” and will in some cases override the values in the configuration file. These features are under development as necessary.

The CDSS configuration properties are described in the following tables:

CDSS HydroBase Database Configuration Properties

Property	Description	Default
ServerNames	A comma-separated list of server names to list in the HydroBase login dialog.	The state server is listed.
Default ServerName	The default HydroBase server name to use. This allows the HydroBase login dialog to preselect a default that applies to most users in the system. If TSTool is run in batch mode and the HydroBase input type is enabled, use this property to make a default connection to HydroBase, for use with other commands in the batch run.	greenmtn. state.co.us
Default DatabaseName	The default HydroBase database name to use. This allows the HydroBase login dialog to preselect a default that applies to most users in the system. If TSTool is run in batch mode and the HydroBase input type is enabled, use this property to make a default connection to HydroBase, for use with other commands in the batch run.	
Database Engine	Reserved for internal use.	
DatabaseName	The database name to use for the initial connection. This overrides the default server.	
Database Server	The server name to use for the initial connection. This overrides the default server.	
SystemLogin	Reserved for internal use.	
SystemPassword	Reserved for internal use.	
UserLogin	Reserved for internal use.	

CDSS Satellite Monitoring System (ColoradoSMS) Database Configuration Properties

Property	Description	Default
ServerNames	A comma-separated list of server names to list in the SMS login dialog.	The state server is listed.
Default ServerName	The default SMS database server name to use. This allows the SMS login dialog to preselect a default that applies to most users in the system. If TSTool is run in batch mode and the ColoradoSMS input type is enabled, use this property to make a default connection to the SMS database, for use with other commands in the batch run.	greenmtn. state.co.us
Default DatabaseName	The default SMS database name to use. This allows the SMS login dialog to preselect a default that applies to most users in the system. If TSTool is run in batch mode and the ColoradoSMS input type is enabled, use this property to make a default connection to the SMS database, for use with other commands in the batch run.	
Database Engine	Reserved for internal use.	
DatabaseName	The database name to use for the initial connection. This overrides the default server.	
Database Server	The server name to use for the initial connection. This overrides the default server.	
SystemLogin	Reserved for internal use.	
SystemPassword	Reserved for internal use.	
UserLogin	The user login, for use with TSTool batch runs. The ColoradoSMS.UserLogin parameter can be specified on the command line and will be used when making the initial SMS database connection.	

The SMS database cannot currently be opened with a login dialog. Therefore, correct information must be specified in the CDSS configuration file and the TSTool command line.

6.4 Map Configuration

TSTool can display maps configured as GeoView project files. See the **GeoView Mapping Tools Appendix** for more information about these files. To allow a link between time series and map layers, use the `TimeSeriesMapLayerLook` property in the `TSTool.cfg` file to specify a time series to map layer lookup file (see the **TSTool Configuration File** section above). The following example file illustrates the contents of the lookup file:

```
# This file allows time series in TSTool to be linked to stations in spatial
# data layers. The columns are used as appropriate, depending on the direction
# of the select (from time series list or from the map).
#
# This file has been tested with the \CDSS\GIS\CO\co_TSTool.gvp file. Not all
# possible combinations of time series and map layers have been defined - only
# enough to illustrate the configuration.
# Additional attributes need to be added to the point files to allow more
# extensive functionality. For example, if attributes for data interval (time
# step) and data source are added to the attributes, then a definition query
# can be defined on the layer for displays to use the same data file. The
# configuration below can then use the different names to configure the link
# to time series.
#
# TS_InputType - the time series input type, as used in TSTool
# TS_DataType - the data type shown in TSTool, specific to an input type
#               For example, TSTool uses "Streamflow" for HydroBase, whereas
#               for other input types a different data type string may be used.
# TS_Interval - time series interval of interest (e.g., "Month", "Day", "1Hour"
#               "Irregular")
# Layer_Name - the layer name used in the map layer list
# Layer_Location - the attribute that is used to identify a location, to be
#                  matched against the time series data location
# Layer_DataType - the attribute that is used to indicate the data type for a
#                  station's time series (CURRENTLY NOT USED - UNDER EVALUATION)
# Layer_Interval - the attribute that is used to indicate the interval for a
#                  station's time series
# Layer_DataSource - the attribute that is used to indicate the data source for
#                    a station's time series.
#
# When matching time series in the TSTool time series query list with features
# on the map, the TS_* values are matched with the time series identifier
# values and the Layer_* attributes are matched against specific time series.
#
# Data layers are listed from largest interval to smallest.
"TS_InputType","TS_DataType","TS_Interval","Layer_Name","Layer_Location","Layer_DataSource"
HydroBase,DivTotal,Day,"Diversions",id_label_7,""
HydroBase,DivTotal,Month,"Diversions",id_label_7,""
HydroBase,EvapPan,Day,"Evaporation Stations",station_id,""
HydroBase,EvapPan,Month,"Evaporation Stations",station_id,""
HydroBase,Precip,Irregular,"Precipitation Stations",station_id,""
HydroBase,Precip,Day,"Precipitation Stations",station_id,""
HydroBase,Precip,Month,"Precipitation Stations",station_id,""
HydroBase,RelTotal,Day,"Reservoirs",id_label_7,""
HydroBase,RelTotal,Month,"Reservoirs",id_label_7,""
HydroBase,Streamflow-DISCHRG,Irregular,"Streamflow Gages - Real-time",station_id,""
HydroBase,Streamflow,Day,"Streamflow Gages - Historical",station_id,""
HydroBase,Streamflow,Month,"Streamflow Gages - Historical",station_id,""
```

Example Time Series Map Layer Lookup File

The columns in the lookup file indicate how information in the time series input/query list can be matched against time series in map layers. In particular, the `TS*` columns define values that are seen in the TSTool interface and the `Layer*` columns define the layer and attribute names for map layers. The `Layer_Interval` and `Layer_DataSource` are optional but if specified result in more specific links between time series and map layers.

This page is intentionally blank.

Appendix: TSTool Release Notes

Version 9.09.00, 2010-09-30

This appendix provides information about changes that have occurred in TSTool versions.

1. TSTool Version History

The following table summarizes the TSTool release history. See the following sections for more detailed information about each version. Only recent versions are documented in detail. Comments for minor versions may be listed under a version that is publicly released. Recent release note items are categorized as follows:

Bug Fix – A bug has been fixed. Users should evaluate whether their work is impacted.

Known Limitation – A known limitation has been documented and may impact the user. The limitation will be addressed in a future release.

Change – An existing feature has been changed.

Remove – A feature has been removed.

New Feature – A new feature has been added, with functionality that was not previously available.

TSTool Version History Summary (most current at top)

TSTool Version	Summary of Changes in Version	Release Date
9.09.00	Add additional commands for table processing. Improve template integration with processor properties and tables.	2010-09-30
9.08.00 – 9.08.01	Support connecting to more than one RiversideDB and introduce the concept of named data stores as an alternative to input type/name. Add <code>TableMath()</code> and <code>TableTimeSeriesMath()</code> commands.	2010-09-15
9.07.00 – 9.07.02	Add HTML summary, improve data flag handling, improve Python integration, initial support for ColoradoWaterHBGuest web service, include training materials, other maintenance.	2010-08-20
9.06.00 – 9.06.04	Initial support for ColoradoWaterSMS web service, enhance RiversideDB support, various improvements.	2010-05-25
9.05.00 – 9.05.03	Enhancements to support additional time series and ensemble processing, in particular to compute statistics for drought studies.	2009-11-17
9.04.00 – 9.04.02	The following features are now at production level: <code>ReadTableFromDelimitedFile()</code> , <code>WriteTableToDelimitedFile()</code> , <code>ResequenceTimeSeriesData()</code> . The <code>CalculateTimeSeriesStatistic()</code> command and additional table processing features have been added.	2009-07-28

TSTool Version	Summary of Changes in Version	Release Date
9.01.00 – 9.03.06	Add <code>VariableLagK()</code> and <code>RunDSSUTL()</code> commands, fix several bugs, and enhance several commands. Add preliminary <code>CheckTimeSeries()</code> , <code>WriteCheckFile()</code> commands. Enhance the <code>ChangeInterval()</code> command and documentation.	2009-04-29
9.00.00 – 9.00.05	Update from Java 1.4.2 to Java 1.6, various bug fixes.	2009-02-05
8.18.00 – 8.18.02	Initial HEC-DSS support. Improved RiversideDB support.	2008-11-24
8.17.01 – 8.17.02	Bug fixes for 8.17.00. See below. New features include File...New to open a new command file and add support for new StateMod 12.29 binary file format.	2008-10-29
8.17.00	All commands are updated to the new error handling and named parameter notation. Many other minor changes have been made for consistency. Many minor user-requested enhancements have been implemented. Several minor bugs reported by users have been fixed. The StateCUB (StateCU binary output file) has been enabled.	2008-10-06
8.16.00 – 8.16.02	Migrate additional commands to new error-handling and named parameter notation. Add <code>RunPython()</code> and <code>FTPGet()</code> commands.	2008-07-22
8.15.01 – 8.15.03	Fix a number of problems where migration of commands from fixed parameter to named parameter syntax resulted in some old command files not being handled. The command file is also now marked as modified if any commands are automatically updated. Added more error checks, such as in <code>DateValue</code> file reading to help provide better feedback to users.	2008-06-11
8.13.00 – 8.14.02	Add commands to set properties, for use by other commands (e.g., to configure file names). Continue updating commands to utilize the new error handling.	2008-02-20
8.03.00 – 8.12.06	Update many commands to utilize new error handling and consistently handle the <code>TSList</code> parameter. Add ensemble processing to many commands. Enable ability for read commands to run in discovery mode to let other commands know time series identifiers. Add more commands to compute statistics time series.	2008-01-14
8.00.00 – 8.02.00	Update main interface to use new error-handling visualization features. Add several commands to allow TSTool to perform regression tests on itself.	2007-12-03
7.04.00	Various updates for HydroBase including adding support for administrative flow station. Allow reading StateMod rights files and handle new StateCU file formats.	2007-06-22
7.01.00	Support new SFUT(G) coding for HydroBase diversion classes, and allow CIU when filling diversion data. Fix a number of bugs in the <code>analyzePattern()</code> , <code>fillInterpolate()</code> , and <code>cumulate()</code> commands	2007-03-02
7.00.00	Begin distributing software using a new installer. Add CASS livestock data and human population data.	2006-10-31
6.19.00	Update to extend period when filling with diversion comments.	2006-05-19

TSTool Version	Summary of Changes in Version	Release Date
6.18.00	Add the <code>runCommands()</code> command to facilitate data processing.	2006-05-02
6.17.00	Add the <code>compareFiles()</code> command to facilitate testing.	2006-04-17
6.16.02	Begin adding commands to test data, for alarms.	2006-04-17
6.16.01	Time series to map link is enabled. Improve UNC support. Improve startup performance in batch mode.	2006-02-16
6.16.00	Begin adding support for NDFD (National Digital Forecast Database) input type, and maintenance.	2006-01-31
6.15.00	Begin adding time series to map link.	2006-01-16
6.14.00	Update some commands to named parameter notation, and maintenance.	2005-12-14
6.13.00	Internal release.	2005-11-13
6.12.00	Improve error handling when running in batch mode with graphs.	2005-10-05
6.11.00	Enable the ColoradoSMS input type for hydrograph annotations and update batch mode features to better utilize the CDSS configuration file.	2005-10-05
6.10.09	Maintenance release – convert some commands to use named parameters.	2005-09-28
6.10.08	Maintenance release – convert some commands to use named parameters. Add the <code>newStatisticYearTS()</code> command.	2005-09-22
6.10.07	Maintenance release – convert some commands to use named parameters.	2005-08-24
6.10.06	Release corresponding to the CDSS CD release.	2005-08-04
6.10.05	Respond to CDSS testing feedback.	2005-08-01
6.10.04	Respond to CDSS testing feedback. Add additional query filters for HydroBase stations and structures.	2005-07-20
6.10.03 BETA	Begin phasing in saving time series products to HydroBase and RiversideDB.	2005-07-08
6.10.02 BETA	Update the <code>openHydroBase()</code> command to use free-format parameters.	2005-06-28
6.10.01 BETA	Begin enabling data flags for time series to support enhancements to fill commands.	2005-06-03
6.10.00 BETA	Initial release supporting HydroBase stored procedures with initial prototypes of Mixed Station Analysis and related features. Implement new message log viewer and commands to simplify comparison of time series.	2005-06-01
6.09.03	Maintenance release.	2004-12-21
6.09.02	Maintenance release.	2004-10-05
6.09.01	Add NWSRFS FS5Files input type.	2004-09-01
6.09.00	Add <code>readHydroBase()</code> commands.	2004-08-27
6.08.02	Documentation made current to include all version 6 changes.	2004-07-27
6.08.01	Allow HydroBase connection to be made at startup.	2004-07-20
6.08.00	Allow wildcards in commands that read from StateCU and StateModB input types.	2004-07-11
	Initial Java version.	1997-10-23

Known Limitation When saving time series product (*.tsp) files, the absolute path of files is saved. This is not as portable as saving a path relative to the command file. It may be necessary to edit the product file manually to change file paths from absolute to relative – the relative path will then be converted to absolute when processed and time series files will be found, assuming that the locations are consistent.

Known Limitation The ReadStateCUB() command, unlike other read commands, does not provide a discovery mode. Consequently, other commands will not be provided with a list of time series identifiers for the binary file. The reason for this is that StateMod and StateCU binary files can contain a huge number of time series and providing a list could be overwhelming and slow. Alternatives are being evaluated. Currently, commands that reference time series in the binary files must use more generic selection methods such as TSLIST=AllMatchingTSID and TSID with wildcards.

Known Limitation Plotting features do not know understand the concept of instantaneous, mean, and accumulated time series (referred to as the time scale). All values are plotted at data value date/time. In the future, features may be implemented to automatically determine from the time scale whether to adjust the visual representation based on the time scale.

Changes in Version 9.09.00

- **Bug Fix** [09.00.00] The RunCommands() command was not passing data stores to the processor used for the command file being run. This is now the default and a parameter has been added to not pass the data stores.
- **Bug Fix** [09.00.00] Fix bug where **File...Open HydroBase** was not showing the HydroBase login.
- **Change** [09.09.00] Several commands have been updated to have Optional/Required language in editors – this will continue until all commands are updated.
- **Change** [09.09.00] The ExpandTemplateFile() command now exposes processor properties set with SetProperty() to the template expansion tool. One-column tables are also exposed as lists. This allows template processing to be dynamically controlled.
- **Change** [09.09.00] The FillRepeat() command now accepts a FillFlag parameter.
- **Change** [09.09.00] The SetTimeSeriesProperty() command now allows a user-defined property to be set.
- **New Feature** [09.09.00] Add the CopyTable() command, useful for creating one-column tables for lists that can be used to expand templates.
- **New Feature** [09.09.00] Add the ManipulateTableString() command.
- **New Feature** [09.09.00] Add the SetTimeSeriesPropertiesFromTable() command, which can be used to set user-defined properties for a time series.
- **New Feature** [09.09.00] Add the ReadTableFromDBF() command, which reads a table from a dBASE file (e.g., associated with an ESRI GIS shapefile).

Changes in Versions 9.08.00 – 9.08.01

- **Bug Fix** [09.08.01] HydroBase AutoConnect property in TSTool configuration file was not being recognized for non-CDSS configurations. This has been fixed.

- **Bug Fix** [09.08.01] The `CalculateTimeSeriesStatistic()` command now properly matches time series identifiers in existing records rather than adding new records for output. The statistic column also is automatically added if it does not exist.
- **Bug Fix** [09.08.00] Data units for HydroBase data were shown as blank in the time series list for many data types – this has been fixed. Units have always been properly set in time series results.
- **Bug Fix** [09.08.00] Better handle time series with no data in graphs – time series are ignored and warnings are not shown (see also new feature below that highlights such time series in the time series list).
- **Change** [09.08.01] The RiversideDB query panel now has 6 input filters and choices are editable to allow matching substrings.
- **Change** [09.08.01] The `CalculateTimeSeriesStatistic()` command now allows the TSID column format to be specified, to allow more control over linking data.
- **Change** [09.08.00] The **File...Open...RiversideDB** functionality now reads a RiversideDB configuration file rather than the full TSTool or RiverTrak® configuration file and does not prompt for a login (by default data can be read but not written to the database). See the **RiversideDB Data Store** appendix for more information.
- **New Feature** [09.08.01] Add `TableMath()` command to perform simple math on table columns.
- **New Feature** [09.08.01] Add `TableTimeSeriesMath()` command to perform simple math on time series using input from a table.
- **New Feature** [09.08.00] Multiple RiversideDB databases can be opened using data store names. Data stores are suitable for databases and binary files and are an alternative to the input type/name convention. Data store names are now listed above the input types in the **Input/Query** area if data stores are available.
- **New Feature** [09.08.00] Time series that do not have data are now indicated with red text in the time series results list and are handled better in the graphing tool.

Changes in Versions 9.07.00 – 9.07.02

- **Bug Fix** [09.07.02] The `RunningAverage()` command was generating errors trying to compute N-year running average values on Feb 29 for daily and finer data. The values are now set to missing.
- **Bug Fix** [09.07.00] The table display for time series now shows numbers right-justified. The display had been left-justified for awhile.
- **Change** [09.07.02] The `ReadStateCU(...,AutoAdjust=True,...)` value is now the default to help ensure that TSTool can properly handle StateCU data types that include periods.
- **Change** [09.07.01] The `ReplaceValue()` command now provides an `Action` parameter to allow setting values to missing (or removing in irregular time series), and an analysis window can be specified to process data in a part of the year.
- **Change** [09.07.01] The `CheckTimeSeries()` command now provides an `Action` parameter to allow setting values to missing (or removing in irregular time series).
- **Change** [09.07.00] Period and monthly time series limits now include median, standard deviation, and skew statistics to facilitate additional analysis.
- **Change** [09.07.00] Status messages now indicate the command being run during processing, in addition to the progress percent estimate.

- **Change** [09.07.00] The `WriteSummary()` command now outputs an HTML file if the output file extension is “html”, and allows the output year type to be specified in the command. An HTML report is also available from the main window results menu. The HTML report color-codes missing and flagged values and provides notes explaining flags. Additional enhancements to output will be added in the future.
- **Change** [09.07.00] The `CompareFiles()` command now has an `AllowedDiff` parameter to indicate that a certain number of lines are allowed to be different, which is useful, for example, for comparing files that have a date/time or software version in output.
- **Change** [09.07.00] The `ReadDelimitedFile()` command has improved error handling when invalid column names are specified in parameters.
- **Change** [09.07.00] The `FillHistMonthAverage()` command now accepts `FillFlag=Auto` and `FillFlagDesc` to better control flagging of filled values.
- **Change** [09.07.00] The `CheckTimeSeries()` command now accepts `Flag` and `FlagDesc` parameters to annotate values that are detected during the check, and the `Change>` and `Change<` check criteria have been added.
- **Change** [09.07.00] The `#` comment command now automatically has a status of success after editing, which avoids the “unknown status” indicator next to the command.
- **Change** [09.07.00] The `RunPython()` command now uses Jython 2.5.1 (when running the Jython embedded interpreter). Support has also been added for IronPython (the .NET implementation of Python) and additional parameters have been added to facilitate integration in various environments).
- **New Feature** [09.07.02] Training materials are included in the *doc/Training* folder under the installation. Additional examples will be added in the future.
- **New Feature** [09.07.00] An initial implementation of the ColoradoWaterHBGuest web services has been added, which allows accessing HydroBase via web services (no need for local database install). Initial work focuses on the `DivTotal` data type. Other data types will be handled in the future.
- **New Feature** [09.07.00] Flags associated with time series are now handled better. The 1-character limitation has been removed internally and restrictions imposed by commands will be removed over time. Flagged values are automatically noted on the HTML summary report.
- **New Feature** [09.07.00] The results area now provides **Views**, which allow more customized ways of listing, viewing time series. An initial version of the `NewTreeView()` command has been implemented to create a tree view. Additional views will be added in the future.

Changes in Versions 9.06.00 – 9.06.05

- **Bug Fix** [09.06.02] The `CalculateTimeSeriesStatistic()` command was reporting fraction for the missing and non-missing percent statistics – it has been fixed to report percent.
- **Bug Fix** [09.06.02] Running commands with `SetOutputPeriod()` and then loading a command file might display warnings for time series read commands because an attempt was made to change the period even though data values are not available. Running the commands would clear the warnings. This has been fixed so that warnings are not generated when loading the command file.
- **Bug Fix** [09.06.02] Commands read from a command file that have invalid parameters were not always generating a visible warning for the user – this has been fixed.

- **Bug Fix** [09.06.02] The `NewStatisticYearTS(..., SearchStart...)` parameter was disabled in the 9.05.x release but has now been restored. The bug resulted in major errors in calculating frost dates (such as time series having mostly very low or high days in year).
- **Bug Fix** [09.06.00] Copying a block of time series from the query results area to the command list when a command was selected resulted in the query results order being reversed – this has been fixed.
- **Change** [09.06.04] The `ReadDelimitedFile()` command has been updated to support reading column headings from the delimited file.
- **Change** [09.06.02] The `ChangeInterval()` command now includes a `Statistic` parameter that supports computing MAX and MIN statistics for INST (small) to INST (large) interval conversions. For example, use this feature to convert instantaneous temperature data to day maximum and minimum temperatures. Additional statistic support will be added in the future.
- **Change** [09.06.02] Opening a new HydroBase or RiversideDB database with **File...Open** now refreshes the input filters for the new connection, rather than just relying on startup configuration. A warning is now displayed when the HydroBase or RiversideDB input types are selected but no database connection is available.
- **Change** [09.06.01] The `WriteCheckFile()` command now includes the execution time for each command – this facilitates evaluation of software performance.
- **New Feature** [09.06.05] Add viewing capabilities for PNG and JPG output files.
- **New Feature** [09.06.04] Add `${InstallDir}` global property for processor to facilitate locating supporting files (e.g., Python scripts) in the installed environment. This property is recognized by commands that expand processor properties (see documentation).
- **New Feature** [09.06.04] Initial support for ColoradoIPP input type in main interface and `ReadColoradoIPP()` command.
- **New Feature** [09.06.03] The `WebGet()` command has been added to allow downloading content from a website.
- **New Feature** [09.06.03] The `ReadFromDelimitedFile()` command functionality has been fully enabled and documented.
- **New Feature** [09.06.02] Querying the time series list from a RiversideDB database now displays a join of time series, station, and location data, and the query can be filtered by the values.
- **New Feature** [09.06.01] If the TSTool configuration file indicates that the HydroBase input type is enabled and the `HydroBase.AutoConnect=True` property is set, then the HydroBase dialog will not be shown and the information in the CDSS configuration file will be used to make the HydroBase connection. This is useful when TSTool is installed in a server environment and everyone will use the same HydroBase connection.
- **New Feature** [09.06.00] The ColoradoWaterSMS input type has been added for interactive queries and TSID commands (specialized read commands have not been implemented). This allows TSTool users to access Colorado's real-time data via internet web services and then analyze it with TSTool features. The web services DO NOT provide access to data from external data providers such as the USGS. Additional enhancements will be made in future releases.
- **New Feature** [09.06.00] The data units that are globally used by TSTool can now be viewed using the **View... Data Units** menu. Data units are important for units conversion and when displaying data.

Changes in Versions 9.05.00 – 9.05.03

- **Bug Fix** [09.05.01] The `AnalyzePattern()` command was miscalculating positions of cutoff values, which, depending on the number of values in a sample, sometimes resulted in an edge pattern value being determined as one position too low. For example, a value near one of the percentile cutoffs would be reported as AVG when it should have been WET. This behavior resulted in a slight bias towards lower categories having higher values due to the extra value. This has been fixed; however, a `Legacy` parameter has been added to duplicate old behavior, in cases where old behavior needs to be retained.
- **Bug Fix** [09.05.00] Time series identifier commands that have invalid time series (e.g., not connected to database or using an invalid file name) generate an error when the command file is loaded. The “discovery” mode would not pass on the identifier to other commands and editors might fail when an empty identifier list is encountered. The identifiers are now passed on to other commands.
- **Change** [09.05.03] Update the `ReadStateMod()` and `ReadStateModB()` commands to allow an alias to be assigned time series that are read, and recognize `${property}` values in the input filename. Also update the `ReadStateMod()` command editor to better handle water right files.
- **Change** [09.05.03] Update the `NewStatisticTimeSeries()` command to handle year, hour, and minute data interval in addition to previous support for month and day.
- **Change** [09.05.02] Update the `NewStatisticYearTS()` to generate the output time series in year type other than calendar and handle other than daily time series (previous limitation).
- **Change** [09.05.02] Update the `ChangeInterval()` command to create year interval time series from daily and monthly data, where the output year type is other than calendar year.
- **Change** [09.05.02] Update the `ResequenceTimeSeriesData()` command to process output year types other than calendar year.
- **Change** [09.05.01] Update the `NewStatisticTimeSeries()` command to include Min, Max, and Median statistics, output period (in particular to allow output to be shortened to one year), and add a parameter to require a minimum sample size for computations.
- **Change** [09.05.01] Update the `NewStatisticTimeSeriesFromEnsemble()` command to include Min, Max, and Median statistics, output period (in particular to allow output to be shortened to one year), and add a parameter to require a minimum sample size for computations.
- **Change** [09.05.01] Update the `CalculateTimeSeriesStatistic()` command to calculate the following statistics: `DeficitMax`, `DeficitMean`, `DeficitMin`, `DeficitSeqLengthMax`, `DeficitSeqLengthMean`, `DeficitSeqLengthMin`, `DeficitSeqMax`, `DeficitSeqMean`, `DeficitSeqMin`, `Lag-1AutoCorrelation`, `Skew`, `StdDev`, `SurplusMax`, `SurplusMean`, `SurplusMin`, `SurplusSeqLengthMax`, `SurplusSeqLengthMean`, `SurplusSeqLengthMin`, `SurplusSeqMax`, `SurplusSeqMean`, `SurplusSeqMin`, `Variance`.
- **Change** [09.05.01] Update the `AnalyzePattern()` command to allow saving output statistics to a new table, which can then be written to a file with another command.
- **Change** [09.05.00] Rename the `CreateEnsemble()` command to `CreateEnsembleFromOneTimeSeries()` to reflect the command’s specific functionality and to avoid confusion with related commands.

- **Change** [09.05.00] Allow DateValue format files to be written with no time series. This facilitates software testing and helps troubleshoot production command files. Previously an error was generated.
- **New Feature** [09.05.01] Add table display to ensemble results – all time series in the ensemble can therefore quickly be displayed.
- **New Feature** [09.05.00] Add NewEnsemble() command to create a new ensemble and optionally insert 1+ time series into the ensemble.
- **New Feature** [09.05.00] Add InsertTimeSeriesIntoEnsemble() command to insert time series into an existing ensemble.
- **New Feature** [09.05.00] Add TimeSeriesToTable() command to copy time series data to a table.
- **New Feature** [09.05.00] Add ExpandTemplateFile() command to implement templates using FreeMarker (<http://freemarker.org>). This facilitates adding conditional logic, loops, etc., to command files.

Changes in Versions 9.04.00 – 9.04.03

- **Bug Fix** [09.04.03] Fix bug in ResequenceTimeSeriesData() command where the last year in the resequenced time series contained missing values.
- **Change** [09.04.00] Finalize ReadTableFromDelimitedFile() command features for production use.
- **Change** [09.04.01] Finalize ResequenceTimeSeriesData() command for initial production use.
- **New Feature** [09.04.02] Add NewTable() command to create an empty table that can receive output from other commands.
- **New Feature** [09.04.02] Add CalculateTimeSeriesStatistic() command to compute statistics and optionally save in a table.
- **New Feature** [09.04.02] Add initial Principal Component Analysis (PCA) tool and FillPrincipalComponentAnalysis() command – the command will be finalized after additional testing and review.
- **New Feature** [09.04.01] Enable ability to read RiversideDB information from TSTool configuration file for batch runs.
- **New Feature** [09.04.00] Add WriteTableToDelimitedFile() command. This command is initially being used to test the read command but can be utilized as more table features are enabled.

Changes in Versions 9.01.00 – 9.03.04

- **Bug Fix** [09.03.05] Update the ChangeInterval() command to better handle negative values in some computations.
- **Bug Fix** [09.03.04] The SetTimeSeriesProperty() command was not allowing wildcards in the TSID parameter – this has been fixed.
- **Bug Fix** [09.03.00] The CreateFromList() command now ignores lines in the input that result in empty location identifiers – this was causing unexpected warnings.

- **Bug Fix** [09.01.01] The `FillRegression()` command was not recognizing the `AnalysisStart` and `AnalysisEnd` parameters – this has been fixed.
- **Bug Fix** [09.01.00] Fix `WriteSummary()` to output in water year if the year type has been set with `SetOutputYearType()`.
- **Bug Fix** [09.03.00] Fix several issues with the `ReadHecDss()` and `WriteHecDss()` commands related to hour 23/24 conversions and address feedback about the previous release.
- **Change** [09.03.06] Update `ChangeInterval()` command documentation to reflect current software capabilities. Also update the dialog to clarify notes about some parameters.
- **Change** [09.03.04] Update `WriteSHEF()` to provide more override parameters and allow appending to the output file.
- **Change** [09.03.02] Update `VariableLagK()` to allow negative lag.
- **Change** [09.03.00] Update `RemoveFile()` to fail if the file was not removed – users will need to check file permissions if the remove did not occur.
- **Change** [09.03.00] Finalize the `VariableLagK()` command features for release. The `DataUnits` parameter has been changed to `FlowUnits` and comments and command messages now also use “flow”.
- **Change** [09.03.00] Update `RunCommand()` to provide parameters to specify the program name and each command line argument, in addition to the previous single command line – this facilitates handling of spaces in program name and arguments. Add the `ExitStatusIndicator` parameter to allow specification of a string to detect the exit status in output. Allow double quotes to be “escaped” in the program name and arguments by using `\`. Add the `UseCommandShell` parameter to indicate whether the command shell should be used – disabling the command shell for simple executable calls can increase performance and simplify error handling.
- **Change** [09.02.00] Change `Lag()` to append “routed” to the scenario, instead of setting the data sub-type – this more cleanly ensures that distinct yet similar time series result from the command.
- **Change** [09.01.00] Update `WriteSummary()` to offer full `TSList` parameter options similar to other commands.
- **New Feature** [09.03.06] Use default HTML viewing program for user environment when viewing HTML files (such as check files) and add documentation as **Help... View Documentation** menu.
- **New Feature** [09.03.04] Add preliminary `CheckTimeSeries()` command to test time series for invalid values, perform quality control, etc.
- **New Feature** [09.03.04] Add preliminary `WriteCheckFile()` command to write a summary of command processing warnings and failures.
- **New Feature** [09.03.00] Add `RunDSSUTL()` command to run the Army Corps of Engineers’ HEC DSSUTL program and other utility programs.
- **New Feature** [09.01.00] Add output year type `NovToOct`, similar to `WaterYear`, suitable for use with some systems. `WriteStateMod()` and `WriteSummary()` now recognize this year type.
- **New Feature** [09.01.00] Add the **Problems** tab to the results to list all warning/failure messages from running the commands. Additional features will be implemented to facilitate viewing. The listing can be sorted by right-clicking on the column heading and can be copied and saved to a file.

Changes in Versions 9.00.00 – 9.00.05

- **Bug Fix** [09.00.05] Using the `Exit()` command would not display the results generated prior to the command – this has been fixed.
- **Bug Fix** [09.00.05] The `ReadHydroBase()` command allowed too many where clauses in queries. A maximum of 6 criteria can be queried based on the current HydroBase interface design, and criteria beyond 6 were being ignored. The command and its editor now only allow up to 6 criteria.
- **Bug Fix** [09.00.04] Writing time series with missing values to NWS Card format could result in values inappropriate for Card files – the software now converts internal missing data values (e.g., NaN) to -999 when writing Card files.
- **Bug Fix** [09.00.03] The `Copy()` command was generating an error when operating on time series with hour interval data and data flags – this has been fixed.
- **Bug Fix** [09.00.03] The `Multiply()` and `Divide()` commands' default behavior is to reset the data units on the modified time series to `units*units` or `units/units`, respectively. However, if the second time series has blank units then `**`, etc. could result – this has been corrected. Additionally, the `NewUnits` parameter has been added to both commands to allow the units to be reset to appropriate values.
- **Bug Fix** [09.00.00] Reading USGS NWIS time series using a TSID command resulted in null dates in the period – this has been fixed.
- **Change** [09.00.03] Upgrade the editor for TSID (time series identifier commands) to allow removing/adding the working directory from file names in the identifiers.
- **Change** [09.00.00] Upgrade Java from version 1.4.2 to 1.6, allowing use of updated third-party components and resulting in an increase in performance.

Changes in Versions 8.18.00 – 8.18.02

- **Bug Fix** [08.18.02] Fix limitation where the cell selection behavior in many tabular displays was not correct when running with Java 5+. TSTool will continue to be distributed with Java 1.4.2 in the short term but Java 6 will be phased in when tests are complete.
- **Bug Fix** [08.18.01] Fix the `SetInputPeriod()` and `SetOutputPeriod()` commands – spaces between parameters were not being handled.
- **Bug Fix** [08.18.01] Fix the uninstaller to remove the python folder used for utility scripts, which results in a complete uninstall.
- **Change** [08.18.01] Add the `MissingValue` parameter to the `WriteDateValue()` command, in particular to support time series read from formats with very large or small values used for missing data.
- **Change** [08.18.01] Improve support for the RiversideDB database – all standard time series data tables are now supported.
- **New Feature** [08.18.02] Begin distributing example data with the installer, starting with `DateValue` examples. See the *TSTool-Version/examples/data/DateValue* folder.
- **New Feature** [08.18.02] Add the **Help...Import Configuration** menu item, which allows a TSTool configuration file to be merged with the current file (e.g., for use after a new software install).

- **New Feature** [08.18.00] Add preliminary support for reading HEC-DSS files in the main interface and the `ReadHecDss()` command. Irregular time series are not supported and by default only the first data block is read – use the `ReadHecDSS()` command with a period to read the full period.

Changes in Versions 8.17.01 – 8.17.02

- **Bug Fix** [08.17.02] When opening HydroBase with **File...Open HydroBase** more than one time, the **Where** filters were not being reset for the new database connection – this has been fixed.
- **Bug Fix** [08.17.02] When loading command files that had time series identifier commands with extra spaces, the user may have seen an error. The error goes away when running the commands. The software now removes unneeded spaces at load so that they are not considered part of the time series identifiers, and the errors consequently do not occur at load.
- **Bug Fix** [08.17.01] When run in batch mode, TSTool was not recognizing the default HydroBase connection information in the *CDSS.cfg* configuration file – this has been fixed, allowing TSTool to access HydroBase in batch mode.
- **Bug Fix** [08.17.01] Fix the bug where no `TSList` parameter for `RunningAverage()` caused an error when running.
- **Bug Fix** [08.17.01] The following commands were not properly transitioning the `TSID` parameter for older command files to new syntax. The behavior is now to set `TSList=AllMatchingTSID` if the older command `TSID` parameter includes `*` and `TSList=LastMatchingTSID` if no wildcard is used. This matches legacy functionality and also supports current conventions. Problems may have occurred if the same `TSID` was reused in the command file because all `TSList=AllMatchingTSID` was used and more time series would have been operated on than desired. The updated commands are: `AddConstant()`, `AdjustExtremes()`, `ARMA()`, `ConvertDataUnits()`, `FillConstant()`, `FillFromTS()`, `FillInterpolate()`, `FillPattern()`, `FillRepeat()`, `Free()`, `ReplaceValue()`, `RunningAverage()`, `Scale()`, `SetConstant()`, `SetDataValue()`, `SetFromTS()`, `ShiftTimeByInterval()`.
- **Bug Fix** [08.17.02] Similar to the previous item, the following commands were not properly transitioning the `IndependentTSID` parameter for older command files to new syntax and have been updated: `FillFromTS()`, `SetFromTS()`, `SetToMax()`, `SetToMin()`.
- **Change** [08.17.02] The `Add()` and `Subtract()` commands now automatically update old syntax to the current syntax – previously a message would be displayed indicating that the command had to be recreated.
- **Change** [08.17.02] Previously, time series aliases with periods would be treated as full time series identifiers and could only be matched with other full time series identifiers during processing. This may have resulted in no match. Aliases with periods are now allowed to be specified and will result in a match with similar aliases when compared with parameters that use the alias. Care must be taken to NOT specify an alias with periods that is the same as a full time series identifier, and which is not intended to be a match. In general, aliases that use periods should either match the full time series identifier or be different enough to not result in an unintended match.
- **Change** [08.17.02] When opening a command file, read commands are run in “discovery” mode in order to determine time series identifiers for command editors. Previous versions would do a full read of the data at this point, which was slow – this has been fixed so that only time series metadata are read when loading command files.

- **New Feature** [08.17.02] Add the **File...New** menu to allow clearing the current commands and starting a new command file.
- **New Feature** [08.17.02] StateMod binary output files as of version 12.29 had a change in the file header – this version of TSTool is able to read the new format while being backward compatible with old formats.

Changes in Versions 8.16.03 – 8.17.00

- **Bug Fix** [08.17.00] Fix RiverWare file reading. Because RiverWare dates always include 24:00, even when not needed, parsing some dates was causing roll-over into the next month. The 24:00 is now ignored for day, month, and year interval time series.
- **Bug Fix** [08.17.00] Fix StateModB file reading – previously an error was occurring when no reservoirs were in the data set.
- **Bug Fix** [08.16.03] Re-enable the general `ReadTimeSeries()` command in the GUI. It was thought that this command would be phased out in favor of specific read commands. However, it is useful in some cases and provides a companion to the `CreateFromList()` command. Also update the command to allow more control over handling missing time series with the `IfNotFound` parameter.
- **Bug Fix** [08.17.00] Fix the `FillRepeat()` command – the `MaxIntervals` parameter could not be set in the command editor.
- **Bug Fix** [08.17.00] Fix many editor dialogs – the `TSID` entry field was disabled for `TSLIST=FirstMatchingTSID` and `TSLIST=LastMatchingTSID`. These parameter values were added for specific commands but became available globally for other commands.
- **Bug Fix** [08.16.03] When running in batch mode on Linux the menu bar graphic was loaded at startup. This causes an error when an X11 connection is not configured (e.g., for cron jobs). This error may still result if processing graphical products in batch mode – more will be done later including updating the Java version used by TSTool.
- **Bug Fix** [08.16.03] Fix the `ReadNwsCard()` command to once again enable the `NewUnits` parameter – this bug was introduced in version 08.03.00.
- **Bug Fix** [08.16.03] Fix so that reading an NWS Card file that is not 24Hour will generate an error if `Read24HourAsDay=True` is specified.
- **Change** [08.17.00] Update the following commands to have new error handling and convert to named parameter notation (if not previously converted): `AdjustExtremes()`, `ARMA()`, `CreateFromList()`, `Disaggregate()`, `Divide()`, `Exit()`, `FillDayTSFrom2MonthTSAnd1DayTS()`, `FillInterpolate()`, `FillPattern()`, `FillProrate()`, `Multiply()`, `NewDayTSFromMonthAndDayTS()`, `NewEndOfMonthTSFromDayTS()`, `Normalize()`, `ReadDateValue()`, `ReadMODSIM()`, `ReadNwsrfsFS5Files()`, `ReadPatternFile()`, `ReadRiverWare()`, `ReadTimeSeries()`, `ReadUsgsNwis()`, `RelativeDiff()`, `ReplaceValue()`, `SetDataValue()`, `SetToMax()`, `SetToMin()`, `StateModMax()`, `WriteStateCU()`. All commands are now updated to the new error handling and named parameter notation.
- **Change** [08.17.00] Disable hiding of problem gutter in main GUI. The problem icons will always be shown and mouse over will popup the command status.
- **Change** [08.17.00] `/*`, `*/` and `Exit()` commands now have command editors even though these commands have no parameters – this provides a consistent handling of all commands.

- **Change** [08.17.00] Change `SetPatternFile()` to `ReadPatternFile()`. The command will automatically be converted when a command file is read.
- **Change** [08.17.00] Change `SetMax()` to `SetToMax()`. The command will automatically be converted when a command file is read.
- **Change** [08.17.00] Change `RemoveFile(WarnIfMissing=...)` to `RemoveFile(IfNotFound=...)` to be consistent with other commands. The command will automatically be converted when a command file is read.
- **Change** [08.17.00] Update the `FillInterpolate()` command to have the `FillStart`, `FillEnd`, and `FillFlag` parameters.
- **Change** [08.17.00] Update the `CreateFromList()` command to change the `HandleMissingHow` parameter to `IfNotFound` and change the default to `Warn`. Users can then decide whether missing time series should be a fatal problem, should be ignored, or should result in default empty time series. Also change the default delimiter to comma (was comma and space) to more explicitly handle comma separated value files.
- **Change** [08.17.00] Update the `ReadNwsrfsFS5Files()` command to allow a relative path for the file.
- **Change** [08.17.00] Update the `WriteSHEF()` command to include the `DataTypePELookup` parameter, to allow assigning the PE code when running in environments when such information is not automatically initialized.
- **Change** [08.17.00] Update the `CompareFiles()` command to include the `CommentLineChar` parameter, to allow setting the comment line character to other than the default (#).
- **Change** [08.17.00] Add full command editor for the `LagK()` command.
- **Change** [08.16.03] Update the `ReadHydroBase()` commands to have the `IfMissing` parameter, to indicate how to handle missing time series. See also the information about the `OpenCheckFile()` command below.
- **Change** [08.16.03] Update the `FillFromTS()` and `SetFromTS()` commands to have the `RecalcLimits` parameter, to recalculate the historical data limits as if all the data were observed in the merged time series. This facilitates combining time series from different sources to create one observed time series.
- **Change** [08.16.03] Update the `SetFromTS()` command to have the `HandleMissingHow` parameter, to allow missing data to be ignored during the transfer.
- **New Feature** [08.16.03] Add the `ReadStateCUB()` command and ability to read StateCU (State of Colorado Consumptive Use model) binary output files in the main interface.
- **New Feature** [08.16.03] Add the initial version of the `OpenCheckFile()` command, to facilitate checking results. `ReadHydroBase()` commands that fail will be listed in the check file. Additional checks will be enabled in the future as the command is enhanced. The check file is viewable in the results area. It is expected that formatting of the output file will change.
- **New Feature** [08.16.03] Add the `WriteTimeSeriesProperty()` command, to facilitate software testing, in particular to write the data limits to test new `FillFromTS()` and `SetFromTS()` command features. In the future this may also be used to save time series information, such as statistics. Additional time series properties will be added over time.

Changes in Versions 8.16.00 – 8.16.02

- **Bug Fix** [08.16.00] TSTool running in batch mode was always exiting with status 0, even if errors occurred. It will now exit with status 1 if any warnings or errors occurred in processing. Refer to the log file for problems or run interactively to fix command input errors.
- **Bug Fix** [08.16.00] In the `Free()` command, the matched time series are now also freed in reverse order from the list in memory – previously the logic may have freed the wrong time series if multiple time series were matched in a pattern.
- **Bug Fix** [08.16.00] The `FillStart` and `FillEnd` parameters were not being recognized by the `FillFromTS()` command – this has been fixed.
- **Change** [08.16.02] Update the `CopyEnsemble()` command to have the `NewAlias` parameter, to allow more flexibility in identifying time series in the copy.
- **Change** [08.16.02] Update the `CreateRegressionTestCommandFile()` command to recognize `@os` and `@testSuite` tags in command file comments, to control collection of test command files.
- **Change** [08.16.00] Reset global properties (except logging levels) to defaults at the start of command processor runs. Previously this was not done and global properties like output period could still be in effect if rerunning commands interactively.
- **Change** [08.16.02] Update the following command to have new error handling and convert to named parameter notation (if not previously converted): `SetAutoExtendPeriod()`, `SetAveragePeriod()`, `SetWorkingDir()`.
- **Change** [08.16.00] Update the following command to have new error handling and convert to named parameter notation (if not previously converted): `DeselectTimeSeries()`, `SelectTimeSeries()`, `SetDebugLevel()`, `SetIgnoreLEZero()`, `SetIncludeMissingTS()`, `SetOutputYearType()`, `SetWarningLevel()`.
- **Change** [08.16.00] Update the `CreateRegressionTestCommandFile()` and `RunCommands()` command to better support testing. The expected status for a command file can now be indicated in a comment. The output report now indicates the expected and actual status and whether the test had an overall pass/fail. See examples of how to use these commands in the documentation.
- **Change** [08.16.00] Update the `Free()` command to use the `TSList` parameter, to allow more flexibility in selecting time series. Also add the `FreeEnsembleIfEmpty` parameter to remove empty ensembles.
- **Change** [08.16.00] Update the `WriteDateValue()` command to have the `Precision` parameter, to allow more flexibility in formatting output. The default is still 4 digits after the decimal.
- **New Feature** [08.16.02] Begin adding Python example scripts to the distribution, located in the *python* folder. Additional scripts will be added over time.
- **New Feature** [08.16.00] Add the `FTPGet()` command to retrieve files from remote systems using file transfer protocol (FTP).
- **New Feature** [08.16.00] Add the `RunPython()` command to run Python/Jython scripts.
- **Remove** [08.16.00] Remove the `SetMissingDataValue()` command, which has not been supported in the GUI for some time. The `SetTimeSeriesProperty()` or another command may be updated to specify the missing data value for the time series.

- **Remove** [08.16.00] Remove the `SetRegressionPeriod()` command, which has not been supported in the GUI for some time. The regression analysis period can be set in the `FillRegression()` command parameters.

Changes in Versions 8.15.00 – 8.15.03

- **Bug Fix** [08.15.03] Re-enable the ability to read default HydroBase connection information from the *system/CDSS.cfg* file when running in batch mode. This allows the user to configure HydroBase once and use with any command file that is run.
- **Bug Fix** [08.15.03] Re-enable the ability to run TSTool in batch mode with `-nomaingui` and have plot windows display until the Close button is pressed. This had been broken in version 8.00.00+.
- **Bug Fix** [08.15.00] Fix a bug in the `Add()` and `Subtract()` commands introduced after 08.02.00. Additional flexibility was enabled to specify the time series list but the new features were not backward compatible with old command files in all cases, in particular when a list of specified time series identifiers was used. Version 08.15.00 is backward compatible and translates old commands on the fly. A workaround is to use version 08.02.00 and change the command parameters to use `TSList=SelectedTS` (instead of `AddTSList=SelectedTSID` or `TSList=SelectedTSID`).
- **Bug Fix** [08.15.00] Fix a bug in the `SetConstant()` command introduced after 08.02.00. Additional flexibility was enabled to specify the time series list but the new features were not backward compatible in all cases. In particular the `TSList` parameter default is now `LastMatchingTSID` when updating old command files (was mistakenly defaulted to `AllMatchingTSID`).
- **Bug Fix** [08.15.00] The ability to right-click on the command list and search for a command was recently broken and has been fixed.
- **Bug Fix** [08.15.00] Printing the Analysis Details from an XY-scatter plot was broken and has been fixed.
- **Bug Fix** [08.15.00] Fix so that the obsolete `SetConstantBefore()` command is treated as an unknown command and verify that all unknown commands are loaded, to allow editing and correction. Previously some obsolete commands might be skipped when loading command files.
- **Bug Fix** [08.15.00] Fix the `ReadNwsCard()` command for ensemble files to handle leap years in the ESP run period (case where ESP run start is Feb 29 is still not handled). Also handle the nonstandard period header produced by the NWS ESPADP software – previously this format error had to be corrected outside of TSTool.
- **Bug Fix** [08.15.00] Fix the `ReadNwsCard()` command to handle reading ensemble files where ESP was run on the last day of the year. The conversion of 1-24 hour to 0-23 hour was causing the data to be shifted by one full month in this case. Also allow an optional ensemble identifier and name to be specified, which will create an ensemble recognized by TSTool.
- **Bug Fix** [08.15.00] Fix the `FillUsingDiversionComments()` command (used when processing HydroBase diversions). A bug was present that caused the filling to not occur when operating on only one time series (filling worked when operating on all time series).
- **Bug Fix** [08.15.00] Fix the `FillMOVE2()` command to properly handle legacy command parameters (prior to named parameter syntax) – this problem only occurred for old command files.
- **Bug Fix** [08.15.00] Fix the `SetFromTS()` command to properly handle legacy command parameters (prior to named parameter syntax) – this problem only occurred for old command files.

- **Change** [08.15.03] Change the `Copy()` command to be more forgiving when reading old command files. The required `NewTSID` parameter will now be defaulted to a copy of `TSID` with scenario “copy”. Using an alias for `TSID` will still require updating the command to specify appropriate `NewTSID` parameter information.
- **Change** [08.15.02] Change the `ChangePeriod()` command to also operate on ensembles.
- **Change** [08.15.00] Change `DateValue` time series file reading to NOT allow multiple adjacent delimiters and do not allow mixing of space and tabs for delimiters. For example, when using commas as the delimiter, “,” would not result in a missing value. The updated software is more strict in order to prevent inadvertent data errors. The default delimiter is a space. If for example, columns are being pasted from Excel using tabs as the delimiter, make sure to add the following line at the top of the `DateValue` file:
`Delimiter = " "`
where a tab character is inside the quotes.
- **Change** [08.15.00] Change `DateValue` time series reading to generate a more explicit error if the file does not exist, to facilitate error checks. Command files that reference invalid files may now generate errors at different processing steps.
- **Change** [08.15.00] Update the `WriteDateValue()` command to recognize ensembles.
- **Change** [08.15.00] Update the `Blend()` command to current error handling and parameter naming conventions. The old syntax is recognized and will be automatically updated.
- **Change** [08.15.00] Fix the `WeightTraces()` command – it had been disabled for some time and has now been updated with command parameters and error handling consistent with current standards. The old syntax is no longer recognized because the command now operates on an ensemble identifier (old depended on less robust time series identifier conventions).
- **Change** [08.15.00] The “REF TS” label shown in the legend for plots, indicating which time series is used in the overview (reference) window under the main plot has been removed. On-screen, saved images, and printed plots should now look the same.
- **Change** [08.15.00] Improve the startup so that database queries for choices do not cause user interface problems.
- **Change** [08.15.00] Software is now distributed with installers that install to a versioned folder and indicate the software version in menus. This allows multiple versions of the software to be installed at the same time. Previous versions evaluated this approach without full installers.
- **New Feature** [08.15.00] Indicate that the command file is modified when reading a command file and changes to command syntax are automatically applied. This will occur with commands that have been fully updated to the new error handling (you are not required to edit the command for its syntax to be updated). The command file can then be saved to accept the automatic changes.

Changes in Versions 8.13.00 – 08.14.02

- **Bug Fix** Warning dialogs in command editors were inadvertently turned off in a previous release and have been enabled again.
- **Bug Fix** Fix so that the `TSAlias` is used if specified in time series product files (used with `ProcessTSProduct()`). This allows aliases to be configured in commands and passed to pre-generated product files, to streamline product processing.

- **Change** The `WriteDateValue()` command has been updated to include a `Delimiter` parameter (e.g., to allow comma to be specified) and the output period can be set in the command. The alias is also now printed in column headings if it has been specified.
- **New Feature** Continue updating commands to have new error handling and to enable ensemble processing for many commands.
- **New Feature** Add `SetProperty()` and `SetPropertyFromNwsrfsAppDefault()` commands to set controlling information for processing. In particular, it is envisioned that this capability will be used to set date/time and filename information at the top of a command file, for use in other commands throughout the command file.
- **New Feature** Add ability to recognize `${Property}` in read/write commands for `DateValue`, `NwsCard`, and `NwsrfsEspTraceEnsemble` commands. This capability will be added to other commands in future releases.
- **New Feature** Add the ability to set the time series alias dynamically in the `ReadNwsrfsEspTraceEnsemble()` command.
- **New Feature** Add preliminary capability in the `ReadDelimitedFile()` command – additional work will be completed to fully enable this command.
- **New Feature** Add the `ComputeErrorTimeSeries()` command, to create a time series indicating the difference between, for example, observed and simulated time series. Percent error is enabled and additional error measures may be added in the future.
- **New Feature** The `RunPython()` command has been enabled in preliminary fashion, with the goal of implementing full support for calling external Python processing scripts, to support more complex processing.
- **New Feature** Add the `ResequenceTimeSeriesData()` command to resequence years of data in a time series, given a list of years.

Changes in Versions 8.03.00 – 08.12.06

- **Bug Fix** Fix `NwsrfsEspTranceEnsemble` handling to handle leap year and correct bug where time zone was not being handled properly (one hour off).
- **Change** Many commands have been updated to use the `TSList` parameter, which indicates the time series to be processed by the command. Commands are backward compatible; however, the new parameter will not be recognized by older versions of TSTool. Once this parameter is enabled in a command, it will allow additional values to be recognized in the future (e.g., getting the list of time series from a table may be enabled). A consistent approach for the parameter also promotes consistency between commands.
- **Change** As much as possible, update commands that read time series to provide the list of time series identifiers to other commands. This facilitates command editing. For example, when a `Read*()` command is inserted, it will partially run (discovery mode) to read time series information, but not the full data. The time series information is then made available to later commands to facilitate editing the commands.
- **Change** Expand the capabilities of the `SetTimeSeriesProperty()` to include setting whether editable – editable time series will enable editing capabilities in the graph view. Add the `DefaultSaveFile` parameter to the `ProcessTSProduct()` command to help automate saving edited time series.

- **Change** Change all results to a tabbed panel of lists, with appropriate mouse actions. For example, a variety of actions can be taken by right-clicking on the time series results. However, for output files, a single click on a file will result in the file being displayed.
- **Change** Include most output files in the results tab. Some secondary files are not yet included but will be as additional commands are updated with improved error handling.
- **Change** Reorganize general command menus to group related commands and avoid a long list of general commands.
- **Change** Reorganize into a separate command menu commands that only apply to ensembles.
- **Change** The performance of the `ShiftTimeByInterval()` command has been greatly improved.
- **Change** Running “TSTool File.TSTool” will cause the command file to be loaded, but not run. To run in batch mode, continue to run with “TSTool –commands File.TSTool”.
- **Remove** Remove obsolete commands from menus. Running old command files will warn about the obsolete commands and recommend new commands. Most of these commands have not been used for a long time: `SetConstantBefore()` was previously replaced with `SetConstant()`. `FillCarryForward()` was previously replaced with `FillRepeat()`.
- **New Feature** Add the `ReadTableFromDelimitedFile()` and `ResequenceTimeSeriesData()` commands to facilitate generation of stochastic time series.
- **New Feature** Add the `CreateEnsemble()` command to create an ensemble of time series from a single time series (e.g., by shifting and overlapping each year of the time series).
- **New Feature** Add the `CopyEnsemble()` command, which copies each time series in an ensemble.
- **New Feature** Add the `NewStatisticTimeSeriesFromEnsemble()` command, which generates a statistic (e.g., “Mean”) time series from an ensemble.
- **New Feature** Add a command menu group and results tab for table processing. Add the `ReadTableFromDelimitedFile()` command, for example to read a CSV file. It is envisioned that table commands will be used to further automate and streamline processing.
- **New Feature** The `NewStatisticTimeSeries()` command has been added to generate a statistic time series determined from a time series. For example, for the “Mean” statistic, the mean of all Jan 1 daily values are repeated throughout the period for each Jan 1. This allows the mean to be graphed or otherwise used for analysis.

Changes in Versions 8.00.00 – 08.02.00

- **Change** The `Copy()` command now requires a new time series identifier to be specified, in order to avoid confusion with the original time series identifier. Old commands will fail if a valid new identifier is not specified. A simple workaround is to use the same location and interval as the original time series and “copy” for the scenario. Because an alias is assigned to the copy, this full time series identifier will likely only be used for displays about time series details.
- **Change** Begin distributing TSTool such that when installed the software lives in a separate versioned folder with a name similar to “TSTool-08.02.00”. This allows different versions of the software to be installed at the same time, in case a specific version must be used and to allow for transition to new versions without conflicts with other software that may share components. A zip file install is available and a full installer is being created, similar to previous versions.

- **New Feature** Initial implementation of new error-handling features, which display graphics to the left and right of the command area indicating warnings and failures. The intent is to provide users with more immediate and accessible feedback and minimize the need to review the log file. Black dots after running indicate commands that have not been updated to the new error handling. Right click on a command and select “Show Command Status” to see useful information about resolving a problem. A command has 3 phases: initialization, discovery, and run, each with a status of unknown, success, warning, or failure.
- **New Feature** Process commands on a separate thread. This allows the GUI to remain responsive and show command progress during running. Features are being implemented to cancel processing.
- **New Feature** Add `CreateRegressionTestCommandFile()`, `RemoveFile()`, and `StartRegressionTestResultsReport()` commands to facilitate creating command test suites, to allow regression testing. Use these commands to create test suites for testing, to automate testing for future releases.
- **New Feature** Add the `WriteProperty()` command to write a processor property (e.g., the output start date) to a file, primarily for use in testing.
- **New Feature** Add the `RemoveFile()` command for use in testing, and can also be used in normal processing.
- **New Feature** Add the `NewPatternTimeSeries()` command, which can be used to generate test data for other commands, and can also be used for normal processing.

Changes in Versions 7.02.00 – 07.04.00

- Remove checkbox for stored procedures from HydroBase login – the transition to stored procedures has been complete for some time.
- Allow the `readStateMod()` command to read water rights files – this was implemented to verify CDSS StateDMI software processing.
- Add support for HydroBase administrative flow stations.
- Add the `setToMin()` command similar to `setMax()`.
- Update the TSTool PDF documentation to include navigation.
- Update the HydroBase `fillUsingDiversionComments()` command to optionally fill with the CIU (currently in use) flag.
- Improve the sizing of the time series query list table.
- Change installer so that when TSTool is run in batch mode from the command line, the working directory is the starting location, rather than the software installation home.
- Update to allow the `readNWSRFSFS5Files()` command to work in batch mode.
- Update to handle new StateCU file formats.

Changes in Version 7.01.00

- HydroBase 20061003 and later has a G: at the end of the SFUT and the F: has been expanded to seven characters. This version of TSTool handles the new identifiers and is backward compatible with older databases and commands files. Old commands files using SFUT should return the same results as before.
- The time series list area now has a minimum height consistent with the HydroBase input type – lists of time series from StateMod or other files are now more readable.
- The `analyzePattern()` command dialog now correctly forces the user to use percentiles in the range of 0 to 1. The command has also been updated to use the output period from `setOutputPeriod()` and the year type from `setOutputYearType()` to write the pattern file.

Consequently, the input time series are no longer required to be the specific water year period to control output. The previous version added “_pattern” on the location part of the TSID, but the current version instead sets the data type to “Pattern” – this will allow the pattern file output to be directly used with `fillPattern()`, using standard locations.

- When saving commands files, the “TSTool” file extension is automatically added. This is compatible with the new installer, which lets the operating know that the extension should be associated with TSTool.
- Fix the `fillInterpolate()` command to allow time series identifiers with space.
- Fix the `cumulate()` command to allow the `HandleMissingHow` parameter to not be specified – it will default to `SetMissingIfMissing`.
- The `fillUsingDiversionComments()` command has been updated to use the CIU HydroBase data to provide more zeros.
- Update to support new StateCU file formats with longer crop names, consistent with similar StateDMI software updates.
- The installer includes several improvements, including more ability to configure the HydroBase information, and displaying previously set HydroBase configuration information as defaults.

Changes in Version 7.00.00

- Begin using the Nullsoft Scriptable Install System (NSIS) to build software installers.
- Begin distributing TSTool as an executable file *TSTool.exe*, which starts up the Java Runtime Environment. This allows for simpler configuration of the **Start** menu and gives users a more traditional executable to run.
- The software organization is slightly different from the previous releases in order to recognize clearer boundaries between components. Several new Jar files are provided, rather than being merged with other Jar files. The **Installation and Configuration Appendix** lists the files.
- Add support for Colorado Agricultural Livestock Statistics and human population time series in HydroBase.

Changes in Version 6.19.00

- Update `fillUsingDiversionComments()` to extend time series with diversion comments available outside the normal diversion records period, if no query or output period has been specified.

Changes in Version 6.18.00

- Add `runCommands()` to allow a controlling commands file to run other commands files.

Changes in Version 6.17.00

- Add `compareFiles()` to help with regression testing, to verify current and expected results.

Changes in Version 6.16.02

- Begin adding data test commands in development mode – these commands will evaluate time series for critical conditions.
- Reenabled `fillMove2()`, which was unintentionally disabled in a previous release.

Changes in Version 6.16.01

- First version that includes operational features to support link between time series and map interface.

- Increase performance at startup when no main GUI is shown, for cases when TSTool is being used to provide graphs for other software.
- Add support for Universal Naming Convention (UNC) for software home in startup files.
- Change **View...Map Interface** to **View...Map**.

Changes in Version 6.16.00

- Implement hooks for the NDFD input type.
- Improve handling of NWS Card file extensions in commands and **File...Save** menu choices.
- Add map interaction features. See the **Installation and Configuration Appendix** for more information about configuring links with maps.

Changes in Version 6.15.00

- Begin implementing link between time series and map interface.
- Reorder general command menus to be more consistent with other software.
- Add warning if time series cannot be retrieved from the RiversideDB input type.

Changes in Version 6.14.00

- Change the `setQueryPeriod()` command to `setInputPeriod()` to be consistent with other software nomenclature. The old command is still supported.
- The `readNwsCard()` and `TS Alias = readNwsCard()` commands both now use the named-parameter notation and have the new `Read24HourAsDay` parameter.
- Blank lines in commands files now display properly.
- Fix bug where time series table sometimes showed half-drawn rows.
- Fix bugs where StateMod binary and StateCU input type file chooser prompt would not allow a cancel of the file select to occur. Cancel now results in the previous file that was selected being displayed.

Changes in Version 6.12.00

- Improve error handling for processing time series products. In particular, TSTool now returns a non-zero exit status if there is an error processing a product. This can be detected by external software that is running TSTool.

Changes in Version 6.11.00

- Enable the ColoradoSMS input type and begin adding alert annotations for streamflow graphs.
- Fix bug so that if a commands file is specified using a relative path, the working directory is interpreted correctly to determine the full path to the commands file.
- Add the ability to accept `Parameter=Value` command line parameters. This will allow override of configuration file information.
- Convert `processTSProduct()` to use named parameters and ensure that output can be viewed even if in batch mode with no main GUI.
- Update so that for batch runs, the `CDSS.cfg` file information for HydroBase is used to make the initial connection. Phase out the HydroBase database properties in the `TSTool.cfg` file.

Changes in Version 6.10.09

- Convert `cumulate()` to use named parameters and begin development of a new Reset parameter.

- Convert `readStateModB()` to use named parameters and add the `Version` parameter to allow reading of old files. The features associated with the `Version` parameter are under development.
- Update the `newStatisticYearTS()` to support calculation of maximum and minimum values in a year and count of values in a year above/below a test value. Also update the command to better handle incomplete data at the end of the analysis period.
- Update the `openHydroBase()` command to check the *CDSS.cfg* information and provide database server and database name choices to the user, to minimize errors in use.

Changes in Version 6.10.08

- Convert `fillConstant()` to use named parameters.
- Convert `newTimeSeries()` to use named parameters.
- Add the `newStatisticYearTS()` command, in particular to support calculation of frost date time series.
- Update `openHydroBase()` to accept the database name parameter.
- Double-clicking on a command will now cause the editor for the command to be displayed.
- Add a Command Glossary to the documentation and begin to standardize command parameter names to be consistent.

Changes in Version 6.10.07

- Convert `scale()` to use named parameters.
- Change `TS X = ...` to `TS Alias = ...` in menus. Start to change notation in documentation and command dialogs.
- Convert `copy()` to use named parameters and add the ability to assign a new TSID to the copy.
- Convert `writeStateMod()` to use named parameters and add ability to select time series to write.
- Convert `readStateMod()` to use named parameters and add parameters for the input period..

Changes in Version 6.10.06

- Official release to support stored procedures.
- Documentation made current to reflect changes since the last documentation issue.
- Respond to feedback from previous 6.10.x incremental releases.
- Fix bug where XY-Scatter graph was not working due to changes in the 6.10.00 BETA release.

Changes in Version 6.10.05

- Add the `lagK()` command.
- Update the `fillProrate()` command `InitialValue` parameter to support `NearestForward` and `NearestBackward`.

Changes in Version 6.10.04

- Add additional input filter choices for `HydroBase` structures and stations, consistent with the `StateView` software.
- Update the `fillProrate()` command to include the `ComputeFactorHow` parameter to allow computing the proration factor based on an average of ratios. Update the command to support free-format parameters.
- Update the `selectTimeSeries()` command to allow combinations of selection filters, to allow more flexibility.
- Add the ability to query `HydroBase` infrequent diversion and reservoir release time series.

Changes in Version 6.10.03 BETA

- Input filters for HydroBase well structures and stations are now handled properly.
- Add initial support for saving time series products to HydroBase and RiversideDB.

Changes in Version 6.10.02 BETA

- Update the `openHydroBase()` command to use free-format parameters.

Changes in Version 6.10.01 BETA

- Enable ability to have data flags for daily and monthly data.
- Update the `writeRiverWare()` command to handle time steps other than hourly.

Changes in Version 6.10.00 BETA

- Begin releasing support for HydroBase stored procedures.
- Begin development of generic `changeInterval()` command and update to free-format parameters.
- Begin work on the Mixed Station Analysis tool and `fillMixedStation()` command.
- Update the `fillRegression()` command to support free-format parameters.
- Begin work on the `analyzePattern()` command.
- Add the **Commands...Analyze Time Series** menu for analysis commands.
- Add the **Commands...Models** menu for more complicated modeling commands.
- Add the **Tools...Analysis** menu for analysis tools.
- Begin implementing the generic log file viewer, which allows links between commands and log messages.
- Change defaults to NOT display messages to the console, to improve performance.
- Add the point graph type.
- Add the predicted value graph type.
- Add the predicted value residual graph type.
- Add the `sortTimeSeries()` command.
- Add the ability for the `readNWSCard()` command to read 1+ time series.
- Add the `startLog()` command.
- Add the `compareTimeSeries()` command.
- Update the `fillHistMonthAverage()` and `fillHistYearAverage()` commands to have fill flag and free-format parameters.
- Add a warning in the `add()` command when frost date time series are added and indicate more appropriate commands.

Changes in Version 6.09.03

- Fix bug where initial directory with spaces in name was causing errors.

Changes in Version 6.09.02

- Added release notes to documentation.
- Fix bug in NWSRFS FS5Files input type where identifiers with underscores were not being handled.
- StateModB input type reservoir data types (and some well types) had ? for data groups – this has been resolved using StateMod 10.27 HTML documentation.

Changes in Version 6.09.01

- Added NWSRFS FS5Files input type support, for use with the National Weather Service River Forecast System (NWSRFS).
- Fix summary reports (daily totals and means) to handle minute data.

Changes in Version 6.09.00

- Add the `readHydroBase()` command to read one or more HydroBase time series while filtering based on location, ID, etc.

Changes in Version 6.08.02

- Documentation updated to reflect all version 6 changes.
- Minor corrections to interface based on documentation review.

Changes in Version 6.08.01

- For the HydroBase input type, allow the ODBC DSN to be specified in the TSTool configuration file, to allow a HydroBase connection to be made at startup without prompting. This supports the CDSS CD distribution.

Changes in Version 6.08.00

- Allow StateCU input type time series read commands to allow wildcards.
- Allow StateMod input type time series read commands to allow wildcards.

This page is intentionally blank.

Appendix: ColoradoIPP Input Type

2010-05-20

Overview

The ColoradoIPP input type corresponds to the Identified Projects and Processes (IPP) database used by the Colorado Water Conservation Board (CWCB) to evaluate long-term water supply conditions in the State of Colorado. The database stores several primary object types (referred to as subject types), which have metadata and related time series. Subject types include:

- Basin (river basins, pending new database development)
- County
- Provider (entities that provide water to users)
- Project (water projects associated with providers)
- State (pending new database development)

ColoradoIPP and Standard Time Series Properties

The standard time series identifier for ColoradoIPP time series is of the form:

`Location.DataSource.DataType.Interval.Scenario~ColoradoIPP`

More specifically, the identifier follows the convention:

`SubjectType:SubjectID.DataSource.DataType-Subtype-Method-Submethod.Year.Scenario~ColoradoIPP`

where identifier parts are described as follows:

- `SubjectType` is `County`, `Project`, or `Provider`.
- `SubjectID` is the identifier for a `SubjectType` object, for example the county name (**currently Provider and Project identifiers are numbers; however, the database will be updated to use a unique, human-readable string**).
- `DataSource` is the data source for the time series (**currently this is a verbose string; however, the database will be updated to use a unique, human-readable string**).
- `DataType` is the time series data type (e.g., `demand`).
- `Subtype` is a sub-type for the time series (e.g., `percapita`).
- `Method` is the method by which the data were determined (e.g., `estimated` or `observed`).
- `Submethod` is a modifier for the method (e.g., if different approaches are used to estimate data).
- `Year` is always the 4-digit year for the data (only annual time series are currently saved in the database).
- `Scenario` is often blank but may indicate the scenario for data (e.g., `low`, `middle`, `high`).

Limitations

ColoradoIPP data are often sparse. Software such as TSTool can be used to fill or extend data. The database is under development and will evolve slightly as design changes are implemented and data are loaded.

This page is intentionally blank.

Appendix: Colorado Satellite Monitoring System (SMS) Input Type

2010-03-03

Overview

The State of Colorado's Satellite Monitoring System (SMS) database stores observations, configuration information, processed data, and alarms, related to field observations collected from real-time stations throughout the State of Colorado. This database has links to HydroBase (see the **HydroBase Input Type Appendix**).

The database is currently only used to provide alarm information, for use as annotations on real-time data from HydroBase. In the future, additional capability may be added to read raw and processed time series from the Colorado SMS database.

See also the **ColoradoWaterSMS Input Type** appendix, which uses a web service to provide access to real-time streamflow and other data types.

This page is intentionally blank.

Appendix: Colorado Water HydroBase Guest (ColoradoWaterHBGuest) Input Type

2010-08-20

Overview

The State of Colorado's HydroBase database is the primary database for water data in Colorado. However, using the HydroBase input type in TSTool (see the **HydroBase Input Type** appendix) requires a direct connection to the database, and a local installation of the database may not be available. The ColoradoWaterHBGuest input type provides internet web service access to historical data and is described here:

<http://cdss.state.co.us/DNN/ViewData/WebServices/tabid/90/Default.aspx>

Although the above web service provides many data types, the ColoradoWaterHBGuest input type currently only supports accessing diversion totals on day, month, and year interval. Support for additional data types is envisioned in the future.

The ColoradoWaterSMS input type (see the **Colorado Water SMS Input Type** appendix) provides access to real-time data using a web service.

ColoradoWaterHBGuest Web Service and Standard Time Series Properties

The standard time series identifier format for ColoradoWaterHBGuest time series is of the form:

`Location.DataSource.DataType.Interval~ColoradoWaterHBGuest`

The meaning of the parts is as follows:

- The `Location` is set to the State of Colorado's water district identifier (WDID) for structures.
- The `DataSource` is set to the providing agency (e.g., DWR for diversion data).
- The `DataType` is set to the "measurement type" described in the State's web service documentation (e.g., `DivTotal` for total diversions through a structure). Refer to the **HydroBase Input Type** appendix for a full list of time series data available in HydroBase.
- `Interval` is `Day`, `Month`, or `Year`, as requested. The interval string is converted from HydroBase conventions of `Daily` and `Annual` (monthly and annual diversion data are stored together in HydroBase and are identified as `Annual` data).
- The `ColoradoWaterHBGuest` input type indicates that the data are being read from the ColoradoWaterHBGuest web service.

Limitations

The following limitations of the web service may impact users of the data.

- Data type – only the `DivTotal` data type has been implemented. Additional data types will be supported in the future.
- Time series metadata – some metadata such as units and measurement counts currently are not available from the web service. This information will be displayed as blank in the time series listing.

- Performance – time series metadata (lists of location/data type/interval combinations) are retrievable from the web service by water district, water division, and single entry. In order for TSTool to provide the user with “drill down” capability starting with a full list of available data, it is necessary to request blocks of data from the web service. However, requesting too large a block results in performance problems due to the bandwidth necessary to transmit data across the network. Consequently, TSTool utilizes caching to store lists of time series metadata, grouped by water district, data type, and interval. The cache is populated based on user requests. Consequently, the first time that data are requested for a district, performance will be slower while the data are retrieved. Subsequent listing of the time series should be fast. Time series data are not currently cached and therefore there may be noticeable slowing for large queries. Additional optimization of data transfer will be evaluated as web service use increases.

Appendix: Colorado Water Satellite Monitoring System (ColoradoWaterSMS) Input Type

2010-08-20

Overview

The State of Colorado's Satellite Monitoring System (SMS) database stores observations, configuration information, processed data, and alarms, related to field observations collected from real-time stations throughout the State of Colorado. This database has links to HydroBase and HydroBase also includes real-time data (see the **HydroBase Input Type Appendix**). However, the HydroBase real-time time series are only available to State of Colorado staff that has access to the State's internal server. The ColoradoWaterSMS input type provides internet web services access to real-time data and as described here:

<http://www.dwr.state.co.us/SurfaceWater/help.aspx>

Raw observations as well as hourly and daily aggregations can be requested. Because the data are considered provisional, time series data collected from external providers are NOT available and must be retrieved from the original provider (e.g., USGS). Time series with `DataSource=DWR` and several other data cooperators are available from this web service.

ColoradoWaterSMS Web Services and Standard Time Series Properties

The standard time series identifier format for ColoradoWaterSMS time series is of the form:

`Location.DataSource.DataType.Interval~ColoradoWaterSMS`

The meaning of the parts is as follows:

- The `Location` is set to the State of Colorado's station abbreviation, `ABBREV` (e.g., `PLAKERCO`), which typically is formed from the river/basin name (`PLA`=Platte River), station location (`KER`=Kersey), and state (`CO`=Colorado). Note that currently the web service does not provide the original identifier and therefore the abbreviation is always used.
- The `DataSource` is set to the providing agency. For example, although data are stored in the State's database, the data provider may be USGS or another agency. If different from DWR, then the data may also be available directly from the provider agency using the agency's station identifier. In fact, the State's web services only provide DWR data and do not pass through provisional data from other agencies.
- The `DataType` is set to the "variable" described in the State's web service documentation. Streamflow is the primary data type, indicated by `DISCHRG`. Reservoir/lake level is indicated by `ELEV`, and storage by `STORAGE`.
- `Interval` is `Irregular` for raw data, which is generally evenly spaced but may include more frequent observations during events. `Hour` and `Day` interval are also available as aggregated values.
- The `ColoradoWaterSMS` input type indicates that the data are being read from the ColoradoWaterSMS web service.

Limitations

The following limitations of the web service may impact users of the data.

- **Data type** – a description of data types is not available from the web service and therefore cannot automatically be displayed by software. A query to determine a list of data types from time series is performed once and will result in a noticeable delay when the ColoradoWaterSMS input type is initially selected. Subsequent listing of time series will be relatively fast.
- **Interval** – the interval for raw observations is not provided by the web service and therefore real-time data interval is shown in TSTool as `Irregular`. Stations generally report at regular N-minute intervals; however, some also provide additional event-triggered values that result in more observations in a shorter timeframe. The web service does allow requesting hour and day interval aggregations, which are computed from the raw data.
- **Station** – provider identifiers, spatial data (county, state, HUC, lat/long, UTM) are not currently available for time series lists. `ABBREV` may not be assigned for all stations, in particular for external providers.
- **Data units** – are currently not available from the web service. Units are hard-coded for some data types, and are left blank for other data types, pending clarification from the State.
- **Data** – web service requests must be made with a query period. A default query period of the most recent 14 days is used to ensure that some data are returned. Optionally, use the `SetInputPeriod()` command in TSTool to specify a longer period. Some stations may not report data in off-season periods.

Appendix: DateValue Input Type

2008-05-13,

Overview

The DateValue time series file format can be used to store one or more time series of consistent time interval. The format has been developed by Riverside Technology, inc. The example below shows the format of the file. Important comments about the file format are:

- The file is divided into a header section (top) and data section (bottom). Comments can occur anywhere in the file and are lines that start with #.
- The default delimiter between property columns and data columns is a space. Use the `Delimiter` property to reset the delimiter (e.g., to a tab). Adjacent delimiters WILL NOT be merged into one column.
- If not specified, many of the header properties will be set to reasonable defaults as data are read by software such as TSTool. However, as much information as possible should be specified to allow complete time series handling. Header information is displayed by applications like TSTool to allow selection of time series before the data section is read.
- Properties are checked in a case-independent fashion.
- The `TSID`, `Start`, `End`, and `Units` properties are important for basic time series handling.
- The interval part of the `TSID` is used to determine how memory should be allocated for data.
- The `Start` and `End` values are used to allocate memory for regular interval time series. Dates associated with data values are used to allocate memory for irregular interval time series.
- For regular interval time series, if data lines between the start and end dates are omitted, the unspecified values are set to the missing data value for the time series (default is -999).

```
# DateValueTS 1.1 file
#
# This is a sample of a typical DateValue minute time series. This format
# was developed by Riverside Technology, inc. to store time series data. An
# example file is as follows and conforms to the following guidelines:
#
# * Comments are lines that start with #.
# * Applications often add a comments section at the top indicating how the
#   file was created
# * Any line that starts with a number is assumed to be a data line.
# * Date hours should be in the range 0 to 23 (an hour of 24 will be
#   converted to hour 0 of the next day).
# * If a time is necessary, the date/time may be separated by a space, T, :, or
#   @. If a space is used, use a date and time column headings,
#   if headings are used.
# * The same general format is used for year, month, day, hour, and minute
#   data, except the format of the date is adjusted accordingly.
# * If multiple time series are written, header variables are delimited with
#   space or tab characters. Data are delimited by tab or space (or use the
#   Delimiter property to set the delimiters used for data lines)
# * Internally, the time series identifier may initially be set using the file
#   name. For example, a file name of XXX.USGS.Streamflow.MONTH will result
#   in the location being set to "XXX", the data source to "USGS", the data
#   type to "Streamflow", and the interval to 1 month. The identifier
#   information is reset if individual properties are specified in the file.
# * This format is free-format and additional information may be added in
#   future (e.g., data quality strings).
```

```

# * For portability, data in a DateValue file should have compatible intervals.
# * Header variables and column headers can be enclosed in double quotes if
#   the data contain spaces.
# * Missing data can either be coded as the missing data value or no value
# * Missing records will result in missing data being used when read.
#
# The following header variables are recognized.  This information can be
# used by software.
Version = 1.1          # Optional.  File format version
                        # (to handle format changes)
Delimiter = " "        # Optional.  Delimiter for property and data lines
                        # (default is space)
NumTS = 2              # Optional.  Number of time series in file
                        # (default is 1)
TSID = "XXX.USGS.Streamflow.15MINUTE" "YYY.USGS.Streamflow.15Minute"
                        # Required.
                        # List of time series identifiers in file
                        # Location.Source.DataType.Interval.Scenario
                        # Do not include input type and name in identifier
SequenceNum = 1950 1951 # Optional - used with time series traces.
                        # Indicates the year for the trace
Description = "Flow at XXX" "Flow at Y"
                        # Optional.  Description for each time series.
DataFlags = true,1 false # Optional.  Indicates whether data flags (e.g.,
                        # character data quality) are provided.  If true,
                        # specify the maximum number of characters that
                        # will be be used in any flag (the default is 2 if
                        # not specified).  The data value column for each
                        # time series with data flags is followed by a
                        # column for the data flag.  Surround the flags by
                        # "" if a flag is not specified or is a space.
DataType = Streamflow Streamflow
                        # Optional.  Data types for each time series
                        # (consistent with TSID if specified).
                        # The default is to use the data type in the TSID
                        # Supplied to simplify use by other programs.
Units = CFS CFS        # Optional.  Units for each time series
                        # (default is no units).
MissingVal = -999 -999 # Optional.  Missing data value for each
                        # time series (default is -999).
IncludeCount = true    # Optional.  If true, column after date/time
                        # is record count (1...) (default is false).
IncludeTotalTime = true # Optional.  If true, column after date
                        # is cumulative time (0...) (default is false).
# Both of above can be true, and both columns will be added after the date
Start = 1996-10-18:00:00 # Required.  Start date for time series
End = 1997-06-14:00:00  # Required.  End date for time series
                        # Period dates should be of a precision consistent
                        # with the dates used in the data section below.
# Optional.  The following line can be read into a spreadsheet or database for
# headers.  The lines above this line can be ignored in a spreadsheet import.
# The number of headings should agree with the number of columns.
Date "Time" "Count" "TotalTime" "Description 1" "DataFlag1" "Description 2"
1996-10-18 00:00 1 0 110.74 "m" 14.2
1996-10-18 00:15 2 15 113.24 "" 13.7
...

```

DateValue Files and Standard Time Series Properties

The standard time series identifier for DateValue files is of the form:

```
Location.DataSource.DataType.Interval.Scenario~DateValue~PathToFile
```

Because DateValue time series files are a persistent storage format for in-memory time series objects that have been developed by RTi, the properties stored in the file closely match the standard time series properties of the objects. In particular, the time series data type, units, and missing data value are consistent with time series header information. The TSID property in a DateValue file is directly applied to time series objects read from the file, allowing explicit identification of the time series in the file, regardless of the name of the file. This allows multiple time series to be saved in a single file. The data source typically agrees with that determined from a data-supplying agency or model that generates the data.

Limitations

DateValue files have the following limitations:

- The header information in DateValue files may be too technical for some general tools. However, simple delimited files cannot be handled as rigorously by some applications, like TSTool. Spreadsheets can import DateValue files easily by ignoring the header lines.
- Because date/time values are included on every data line, processing DateValue time series files requires more disk space and processing time. However, using the dates on each line also allows gaps in data to be omitted from the file. Inclusion of the date/times for each data point is considered a reasonable trade-off to ensure data quality and readability. Many other time series file formats also include the date/time on each line.

This page is intentionally blank.

Appendix: HEC-DSS Input Type

2009-01-14

Overview

HEC-DSS input type refers to the United States Army Corps of Engineers' Hydrologic Engineering Center (HEC) Data Storage System (DSS). Refer to the following web sites for more information:

<http://www.hec.usace.army.mil> (main web site)

http://www.hec.usace.army.mil/software/legacysoftware/hec-dss_pc-dos/documentation/overview.pdf
(HEC-DSS User's Guide and Utility Program Manuals overview)

HEC-DSS Files and Standard Time Series Properties

The standard time series identifier used with TSTool and other software is of the form:

```
Location.DataSource.DataType.Interval.Scenario~InputType~PathToFile
```

The implementation of the identifier for HEC-DSS files is of the form:

```
Apart:Bpart.HEC-DSS.Cpart.Epart.Fpart~HEC-DSS~PathToFile
```

HEC-DSS time series identifier information is taken from the A-F "pathname parts" used to identify HEC-DSS time series. The following assignments are made:

- The location part of the identifier is set to the A-part, a colon, and the B-part. This retains the original HEC-DSS location information. **The colon and period characters cannot be used in the original HEC-DSS A- and B- parts because they conflict with the identifier implementation described above. Instead, it is recommended that dashes, underscores, and other delimiting characters are used within the parts if the HEC-DSS data will be used extensively with TSTool.**
- The data source part of the identifier by default is set to HEC-DSS to indicate that "HEC-DSS" is the provider of the data. In the future this could be used to store the data source (data provider) such as "USGS", "NWS", etc., if such information could be obtained from the HEC-DSS time series pathname or supplemental data.
- The data type is set to the C-part. In HEC-DSS, this is referred to as the "parameter". In HEC-DSS the term "type" is used to indicate whether a time series is instantaneous, mean, or accumulated. **The period character cannot be used in the original HEC-DSS C-part when used with TSTool.**
- The data interval is set to the E-part. **The period character cannot be used in the original HEC-DSS E-part when used with TSTool. Currently the irregular interval is not supported, but will be added in the future.**
- The scenario is set to the F-part. **The period character cannot be used in the original HEC-DSS E-part when used with TSTool.**
- The data units are determined from time series information.
- The D-part is initially used to assign the time series period, but is reset with information from the time series data records, if available. The D-part dates are of the form DDMonYYYY or DDMMYYYY – DDMonYYYY. A single date indicates the start of a data block in the HEC-DSS data management scheme (see the HEC-DSS documentation referenced above). Two dates indicates the starting and ending data blocks for a condensed catalog (list of time series); however, in this case the ending date

is actually the start of the last data block. The size of each data block depends on the time series interval (e.g., year interval data are stored in blocks of centuries). The HEC-DSS standards for block size are used in conjunction with the time series interval to set the time series end date as appropriate.

- The missing data value is assigned to the internal representation for HEC-DSS files (a large negative number).
- The description is by default set to the location, a comma, followed by the data type.

Limitations

The following limitations are known with the HEC-DSS input type:

- A- and B- parts that include colons will cause an error when converting HEC-DSS identifiers to/from the convention described above – avoid using colons in the A- and B- parts.
- A-, B-, C-, E-, and F- parts that include periods will cause an error when converting to/from HEC-DSS identifiers to the convention described above – avoid using colons in the location identifier. If encountered, the alias for the time series is set to the full identifier with periods. The periods are then removed before setting the full identifier.
- Irregular data are not fully implemented but are supported by the software architecture – they will be enabled in the future.
- Paired data are not currently read – the data are envisioned to be read into the table objects supported by TSTool.
- Access to supplemental data such as station comments is not implemented but is supported by the software architecture.
- Data units from HEC-DSS time series are read and used with the time series. However, conversion between units may not be supported if the units are not included in the data units file used by software. Additional data unit definitions can be added if necessary to facilitate conversions.
- HEC-DSS can store the observation date/time for data to a higher precision than the time series data interval. For example, the observation date/time for an annual value may be July 30, 2400 of the year. The additional precision is ignored when read. An observation time at the end of an interval that might result in a different base date/time (e.g., hour 2400 causing the day to increment for daily interval data), is handled to prevent the rollover.
- Plotting capabilities do not recognize the data scale (instantaneous, mean, accumulated) – line graphs are always drawn as if data were instantaneous. This could be addressed by enhancing the software to utilize a data type file and display styles to determine when data types are instantaneous, mean, or accumulated.

Appendix: HydroBase Input Type

2008-09-22

Overview

The State of Colorado's HydroBase database stores a variety of time series data. The time series conventions described here, in particular for time series identifiers, are consistent for major CDSS software components including TSTool, StateView/CWRAT, StateDMI, and StateMod GUI. This allows for consistent features and sharing of data between software tools.

The current database design splits time series into three main categories:

1. Data related to structures or administrative data maintained by the State of Colorado (e.g., diversions, reservoirs). Structure locations are typically identified using a water district identifier (WDID), consisting of a two digit State of Colorado water district number and a trailing structure identifier (which in the past was four digits but has been increased to five or more digits to support longer identifiers). Although a single WDID identifier is used when identifying time series, the separate WD and ID fields are generally needed to find information in HydroBase.
2. Data for stations, consisting mainly of location information and time series (e.g., NOAA precipitation data, USGS streamflow). Station locations are typically identified using a station identifier from the data source. For example, stations can use a USGS identifier, a State of Colorado Satellite Monitoring System abbreviation, or other identifier.
3. Data recorded at locations that are not stations or structures. For example, Water Information Sheet (WIS) are daily spreadsheets used to administer water. Although WIS contain data values for structures and stations, the time series are extracted from database tables that are not directly associated with structure or station database tables. Other examples include Colorado and national agricultural crop statistics.

A structure or station may have more than one identifiers depending on the number of agencies involved with data collection, etc. For example, a reservoir may have a State of Colorado WDID because it has water rights, a Bureau of Reclamation identifier, a US Geological Survey identifier, and a second State of Colorado identifier because real-time data are collected. HydroBase collects data from many sources; however, the State has not attempted in all cases to cross-reference the identifiers. For example, a streamflow station may have a partial time series record with a "USGS" data source and identifier and a partial time series record with a "DWR" (Division of Water Resources) data source and identifier – the user must recognize that this may be the same station, under different management at different times.

HydroBase is updated for release to the public approximately once per year, although internal updates may occur year-round. Time series are used with CDSS (Colorado's Decision Support Systems) applications and follow basic time series standards when used by TSTool and other software.

HydroBase and Standard Time Series Properties

The standard time series identifier format for HydroBase time series is of the form:

`Location.DataSource.DataType.Interval~HydroBase`

Due to the variety of data types, sources, and formats in HydroBase, time series properties can be set a number of ways. General guidelines are as follows:

- The location part of the time series identifier is set to a station or structure identifier, which is typically the identifier used by the managing agency. For example, USGS stream gages will use the 8-digit USGS identifier and State of Colorado diversions will use a structure WDID.
- The source part of the time series identifier corresponds to the current source of the data. For example, if the current provider for a time series is the USGS, then the data source will be USGS. If the State of Colorado has at some point taken over maintenance of a station from the USGS, then the data source will be DWR. Individual data records may indicate a variety of data sources. The convention in HydroBase is to store the data records under the current data source, rather than force the user to query more than one time series and merge the time series. If, however, a station has moved, then separate time series will typically be stored, likely under different identifiers.
- The data type part of the time series identifier as much as possible uses the “measurement type” information in HydroBase or a readable and reasonable data type phrase. For example “Precip” is a measurement type for station data and “DivTotal” (diversion total) is a measurement type for diversion data. In some cases, especially with real-time data, the data type may not exactly match HydroBase. For example, HydroBase uses a measurement type “RT_Rate” for multiple stream related data types. TSTool uses a data type of “Streamflow”. In the past, TSTool and other software used data types that did not as closely match the measurement types in HydroBase. For example, daily streamflow was identified as QME (a National Weather Service notation) because that is how it was defined in CRDSS modeling efforts. The table at the bottom of this appendix describes all available HydroBase data types and provides guidance for upgrading from old data types.
- Data intervals are set based on the tables that are being queried. In most cases, a regular interval like DAY or MONTH is used. IRREGULAR is used for real-time data because there is currently no way to know without doubt what the regular data interval is (e.g., 15MIN). Data that are measured infrequently (e.g., reservoir field measurements) are typically stored as a regular interval time series with interval DAY. This allows more flexibility in data processing and filling.
- In older versions of TSTool, the scenario part of the identifier was sometimes used to supplement the data type information. For example, real-time flow data in the database has a number of attributes (Streamflow, RT_Rate, DISCHRG) that cannot easily fit into the standard time series identifier. The current version of TSTool uses datatype-subdatatype where necessary and generally does not use the scenario for normal time series identifiers (WIS time series are an exception) and this field is being reserved to possibly indicate historical data, filled data, etc.
- Units are set based on the database table definitions.
- Period of record is set based on the available database contents. Periods are typically not determined by checking the data because this would require querying large amounts of data. When listing time series, periods are normally determined from summary information available in the database. In some cases, the period of record information is not saved at a precision sufficient to accurately represent the true period (e.g., the database may indicate data for years but not months). Therefore, the true period will only be available when data are actually queried.
- Missing data are typically set to -999 in time series but are typically stored as nulls in the database.
- The input type of the time series identifier may not be used for older applications. The new convention is being phased in and uses an input type of HydroBase (e.g.,

12345678.USGS.QME.DAY~HydroBase). If multiple HydroBase connections are needed, the input name may also be added (e.g., 12345678.USGS.QME.DAY~HydroBase~ServerName), although this capability is only in the evaluation stage.

- The time scale for data (whether accumulated [ACCM], instantaneous [INST], or mean [MEAN]) is not automatically determined from the data type and interval.

Diversion data may be retrieved from several tables in HydroBase, including daily and monthly detailed records, infrequent values, diversion comments, and currently in use values. The TSTool `ReadHydroBase()` command gives several options for handling data and the `FillUsingDiversionComments()` command can be used to fill with additional zero values. When using time series identifiers to read time series, the following defaults are used:

- Daily `DivClass` and `DivTotal` time series are filled using the carry-forward technique implemented by the State of Colorado. Missing irrigation years remain missing. Years with data are filled with zeros at the start and values are carried forward until another observation is found, or to the end of the irrigation year.
- Diversion comments and “currently in use” flag are NOT automatically applied. This default may change in the future but is retained for historical data processing reasons.

The following tables present a summary of time series identifier fields for the HydroBase data types. Data sources may be added and/or removed with data updates. Data types are listed by major group and are alphabetized by the data type description within the group. The time scale is provided to facilitate data use, in particular when changing the time interval.

**HydroBase Time Series Types and Standard Time Series Identifier Fields
Agricultural Crop and Livestock Data**

Data Group	Data Type Description	Location	Data Source	Data Type	Available Intervals and Time Scale	Comments
Agricultural/ CASS	Colorado Agricultural Statistics Service crop area harvested	County Name	CASS	CropAreaHarvested-Commodity_Practice Commodity and practice are from available values in HydroBase.	Year INST	See NASS data for orchards, pasture, and vegetables. Perennial crops usually have only harvested value.
	CASS area planted	County Name	CASS	CropAreaPlanted-Commodity_Practice Commodity and practice are from available values in HydroBase.	Year INST	Annual crops should have planted value but use maximum of planted and harvested if necessary.
	CASS livestock head	County Name	CASS	LivestockHead-Commodity_Type Commodity and type are from available values in HydroBase.	Year INST	For each commodity (e.g., sheep), multiple types (e.g. sheep at various maturity levels).
Agricultural/ GIS	CDSS irrigated lands assessment result. See also Diversion Comments below.	WDID	CDSSGIS	CropAreaAllIrrigation-CropType CropAreaDrip-CropType CropAreaFlood-CropType CropAreaFurrow-CropType CropAreaSprinkler-CropType CropType is taken from available values in HydroBase.	Year INST	Data are only available for years where DSS projects or data refreshes have occurred. Partial data for intermediate years may be available in spatial data layer attributes but not HydroBase. Data are available for lands served by surface water structures, listed by crop/year/irrigation type.
Agricultural/ NASS	CropArea	County Name	NASS	CropArea-Commodity Commodity is taken from available values in HydroBase.	Year INST	See CASS data where available. NASS does not distinguish between planted and harvested. NASS data are useful for orchards, pasture, and vegetables, which may not be reported in CASS.

HydroBase Time Series Types and Standard Time Series Identifier Fields (Climate Data)
Climate Group Table 1 of 2

Data Group	Data Type Description	Location	Data Source	Data Type	Available Intervals and Time Scale	Comments
Climate	Evaporation (Pan)	Station ID	NOAA	EvapPan Old (obsolete) data type was EPAN.	Day ACCM, Month ACCM	
	Frost Dates (derived from temperatures)	Station ID	COAGM, NOAA	FrostDateL28S, FrostDateL32S, FrostDateF28F, FrostDateF32F Old (obsolete) data type was FrostDate or FrostDates.	Year INST	Time series in software are the Julian day of the year (1-366) to allow graphing, filling, and manipulation.
	Precipitation	Station ID	COAGM, NOAA	Precip Old (obsolete) data type was PTPX.	Day ACCM, Month ACCM, Irregular ACCM	Irregular data are real-time increments.
	Snow (accumulation on ground during interval).	Station ID	NOAA	Snow Old (obsolete) data type was SNOG.	Day ACCM, Month ACCM	
	Snow course depth and snow water equivalent	Station ID	SCS	SnowCourseDepth, SnowCourseSWE Old (obsolete) data type was SnowCrse, SNWE.	Day INST	Values are recorded on a day, with one or more times a month.
	Solar radiation	Station ID	COAGM	Solar Old (obsolete) data type was RADS.	Day ACCM	
	Temperature (instantaneous)	Station ID	various	Temp	Irregular INST	
	Temperature (maximum)	Station ID	COAGM, NOAA	TempMax Old (obsolete) data type was MaxTemp, TAMN.	Day INST	
	Temperature (mean of maximum daily values)	Station ID	COAGM, NOAA	TempMeanMax Old (obsolete) data type was MaxTemp, TAMX with monthly interval.	Month MEAN	
	Temperature (mean)	Station ID	COAGM, NOAA	TempMean Old (obsolete) data type was MeanTemp, TAVG.	Month MEAN	
	Temperature (minimum)	Station ID	COAGM, NOAA	TempMin Old (obsolete) data type was MinTemp, TAMN.	Day INST	
	Temperature (mean of minimum daily values)	Station ID	COAGM, NOAA	TempMeanMin Old (obsolete) data type was MinTemp, TAMN with monthly interval.	Month MEAN	

HydroBase Time Series Types and Standard Time Series Identifier Fields (Climate Data)
Climate Group Table 2 of 2

Data Group	Data Type Description	Location	Data Source	Data Type	Available Intervals and Time Scale	Comments
Climate	Vapor pressure (mean daily)	StationID	COAGM	VaporPressure Old (obsolete) data type was VP, MVP.	Day MEAN	
	Wind run	Station ID	AGRO, COAGM	Wind Old (obsolete) data type was UDIS.	Day ACCM	

HydroBase Time Series Types and Standard Time Series Identifier Fields (Demographic Data)

Demographic data are related to human population. See the Agricultural Data above for livestock population.

Data Group	Data Type Description	Location	Data Source	Data Type(s)	Available Intervals and Time Scale	Comments
Demographics	Human population (persons)	Area_type- Area_name The type indicates whether a county, municipality, state, etc. The name agrees with the type. The combination defines a unique location.	(blank) This could be assumed from the Pop_type part of the data type; however, the data source is not readily available in HydroBase.	HumanPopulation- Pop_type The population type is Census, Estimated, etc.	Year INST	See CDSS documents for information on how population estimates are determined.

HydroBase Time Series Types and Standard Time Series Identifier Fields (Diversion Data)

Data Group	Data Type Description	Location	Data Source	Data Type	Available Intervals and Time Scale	Comments
Diversion May include records for reservoir and well structures, as per State of Colorado administration practices. See also reservoir data.	Diversion Class (showing water color)	WDID	DWR	DivClass-SFUT Old (obsolete) data type was DQME, Div, or Diversion.	Day MEAN, Month INST or ACCM, Year INST or ACCM	SFUT is encoded as: S:s F:f U:u T:t s = source f = from u = use t = type Annual values are for irrigation year (Nov-Oct).
	Diversion Comment (the acreage for a diversion and string data flag indicating whether a structure irrigated in a year)	WDID	DWR	DivComment	Year INST or ACCM	The numerical time series value is set to the acreage for the year. The data quality flag is set to the HydroBase <i>diversion_comment.not_used</i> flag. Therefore, this time series can be used to extract total acreage for a structure and determine if diversions should be zero for a year. Annual values are for irrigation year (Nov-Oct).
	Diversion Total (sum of all DivClass records for a structure).	WDID	DWR	DivTotal Old (obsolete) data type was DQME, Div, or Diversion.	Day MEAN, Month INST or ACCM, Year INST or ACCM	Annual values are for irrigation (Nov-Oct) year.

The above table summarizes how diversion records are available as time series. However, to determine a complete diversion time series, it is necessary to understand the various ways that diversion records can be stored. See also the **State of Colorado's Water Commissioner Manual**.

Raw data observations for a diversion structure are stored as one or more of the following forms in HydroBase:

- Daily water class time series. These data are recorded using irrigation year (November to October). If one or more values have been entered in a month, then HydroBase will include a full month of data. Days at the beginning of the irrigation year that have no observed values at the start of the year should be considered to be zero, regardless of values found in previous irrigation years. Once an observation occurs, then days within the month where an observation was not recorded are set to the last observed value. Therefore, if an irrigation year contains at least one value, that irrigation year will have at least one month of values (with no missing in the month). To preserve space in HydroBase, months with no observations are not included in the daily data in the database. If a year has no observation, then no data are available in HydroBase for the year and a determination of whether the data values should be zero or other must be determined using other data (see below) or engineering judgment. **TSTool and StateView by default implement the carry-forward procedure within irrigation years.**
- Diversion comments. Diversion comments may be included for an irrigation year. The not_used flag indicates if a diversion was not used in a year. If this is the case, then daily diversion records should not be available and a zero value can be assumed for the water year. **TSTool and StateView DO NOT by default use diversion comments when providing daily or monthly time series.**
- Infrequent water class. Infrequent water class values can be entered as an annual value for the irrigation year, or as a monthly value. The data can be accessed as time series in TSTool, although no specific capabilities have been implemented to supplement the daily or monthly time series.

Processed (derived) data records are created as follows:

- Daily total diversion. Daily water class values are accumulated to daily total records. Similar to the daily water class, any month that has at least one value will result in a month with no missing data. To preserve space in HydroBase, only months that include an observation are included in HydroBase. Other months in the same irrigation year should be carried forward. Irrigation years with no observation have no records in HydroBase and a determination of whether the data values should be zero or other must be made using other data (see below) or engineering judgment. **TSTool and StateView by default implement the carry-forward procedure within irrigation years.**
- Monthly water class. Monthly water class is computed by converting the daily water class values (average CFS) to ACFT for each day of the month, and adding the values. Because of the way that daily data are treated, a month will either have all daily values or none. A month with no data will have its value set to missing in the database. Full irrigation years with no observation will result in a full year of missing values, and a determination of whether the data values should be zero or other must be determined using other data (see below) or engineering judgment. Unlike daily data, monthly diversion records are included in HydroBase for the full data period. Full years of missing values may be included in the database.
- Monthly total diversion. This is derived using the same procedure as monthly water class; however, the daily total diversion is used as input.
- Infrequent data are not considered when producing the monthly total time series.

Therefore, to determine a complete time series, the following must be performed, using TSTool or other software:

Daily time series:

1. Read the daily time series from HydroBase. The default in TSTool and StateView is now to carry forward daily diversion time series within the irrigation year.
2. Utilize the diversion comments to set additional years of data to zero. Using diversion comments is an option with TSTool and StateDMI time series read commands.
3. For years with no data, use an appropriate fill technique. If it is known that the ditch did not operate, then zeros should be used. If it is known that the ditch did operate, use historical averages or some other method to fill the data.
4. HydroBase infrequent diversions could be used to supplement the data, but currently there is no software to help users with this process.

Monthly time series:

1. Read the monthly time series from HydroBase. Any irrigation year with at least one daily observation results in 12 monthly time series values.
2. Utilize the diversion comments to set additional years of data to zero. Using diversion comments is an option with TSTool and StateDMI time series read commands.
3. For years with no data, use an appropriate fill technique. If it is known that the ditch did not operate, then zeros should be used. If it is known that the ditch did operate, use historical averages or some other method to fill the data.
4. HydroBase infrequent diversions could be used to supplement the data, but currently there is no software to help users with this process.

Yearly time series:

1. Infrequent time series can be read by TSTool and can supplement the above data. However, currently there is no software to help users with this process. General TSTool commands must be used as appropriate.

HydroBase Time Series Types and Standard Time Series Identifier Fields (Hardware Data)

Data Group	Data Type Description	Location	Data Source	Data Type(s)	Available Intervals and Time Scale	Comments
Hardware	Battery voltage	Station ID	DWR	Battery	Irregular INST	Limited data are available. This data type allows remote system maintenance checks.

Hardware data types are not commonly available have been implemented as a test and to allow for greater future use.

HydroBase Time Series Types and Standard Time Series Identifier Fields (Reservoir Data)
Reservoir Group Table 1 of 2

Data Group	Data Type Description	Location	Data Source	Data Type	Available Intervals and Time Scale	Comments
Reservoir	Field Measurements	WDID	DWR, other	ResMeasElev, ResMeasEvap, ResMeasFill, ResMeasRelease, ResMeasStorage Old (obsolete) data type was RSTO.	Day INST, Day ACCM, Day ACCM, Day ACCM, Day ACCM	Reservoir measurements are often recorded at the beginning or end of the month.
	Pool Elevation	Station ID or State of CO Abbrev.	DWR, other	PoolElev	Irregular INST	Real-time data for reservoirs are recorded using a station abbreviation that does not match a WDID.
	Release Class (showing water color)	WDID	DWR	RelClass-SFUT	Day MEAN, Month INST or ACCM, Year INST or ACCM	SFUT is encoded as: S:s F:f U:u T:t s = source f = from u = use t = type Annual values are for irrigation year (Nov-Oct).
	Release Comment (the acreage for a release and string data flag)	WDID	DWR	RelComment	Year INST or ACCM	See DivComment comments. Sometimes acreage is associated with reservoirs. Annual values are for irrigation year (Nov-Oct).
	Release Total (sum of all RelClass records for a structure).	WDID	DWR	RelTotal	Day MEAN, Month INST or ACCM, Year INST or ACCM	Annual values are for irrigation year (Nov – Oct).

HydroBase Time Series Types and Standard Time Series Identifier Fields (Reservoir Data)
Reservoir Group Table 2 of 2

Data Group	Data Type Description	Location	Data Source	Data Type	Available Intervals and Time Scale	Comments
Reservoir	Release (instantaneous)	Station ID	DWR, other	Release	Irregular INST	Real-time data for reservoirs are recorded using a station abbreviation that does not match a WDID.
	Reservoir Storage (end of month).	WDID	USBR, DWR, other	ResEOM Old (obsolete) data type was RSTO.	Month INST	Few time series are available.
	Reservoir Storage (end of year).	WDID	USBR, DWR, other	ResEOY	Year INST	From <i>annual_res</i> table. Annual value is for irrigation year (Nov-Oct).
	Storage (instantaneous)	Station ID or State of CO Abbrev.	DWR, other	Storage	Irregular INST	Real-time data for reservoirs are recorded using a station abbreviation that does not match a WDID.

HydroBase Time Series Types and Standard Time Series Identifier Fields (Stream Data)

Data Group	Data Type Description	Location	Data Source	Data Type	Available Intervals and Time Scale	Comments
Stream	Natural Flow	Station ID	USBR	NaturalFlow Old (obsolete) data type was Nat_flow, NQME	Month INST or ACCM	
	Stage	Station ID	DWR, other	Stage	Irregular INST	Real-time data.
	Streamflow	DWR Abbrev. or USGS station ID	DWR, USGS, other	Streamflow Old (obsolete) daily, monthly data type was QME. Old real-time data type used RT_rate and scenario DISCHRG or other VAXfield to indicate channel.	Day MEAN, Month INST or ACCM, Irregular INST	Real-time data use Irregular time interval.
	Streamflow (maximum of daily mean)	Station ID	DWR, USGS	StreamflowMax Old (obsolete) data type was Maxq, Maxflow.	Month INST	
	Streamflow (minimum of daily mean)	Station ID	DWR, USGS	StreamflowMin Old (obsolete) data type was Minq, Minflow.	Month INST	
	Water temperature (instantaneous)	Station ID, State of CO Abbrev.	DWR, other	WatTemp	Irregular INST	Real-time data, using identifier that does not match USGS or other identifier for historical data.

**HydroBase Time Series Types and Standard Time Series Identifier Fields
(Water Information Sheet Data)**

Data Group	Data Type Description	Location	Data Source	Data Type	Available Interval and Time Scale	Comments
WIS	Water Information Sheet (WIS) cell values, over time	WIS row identifier. For example, structures have an identifier wdid:NNNN NNN, where the leading "wdid:" is a literal string and the following information is the actual WDID. Similarly, stations start with "stat:", followed by a station ID; confluences with "conf:", followed by the HydroBase <i>wd_water</i> numbers for the tributary and the larger stream; other row types with "othr:", followed by a sequential number in the WIS.	DWR	Data types match the WIS columns, as follows: WISPointFlow, WISNaturalFlow, WISDeliveryFlow, WISGainLoss, WISRelease, WISPriorityDiversion, WISDeliveryDiversion, WIS TribNaturalFlow, WIS TribDeliveryFlow, WISDryRiver (not currently implemented – may be implemented as a data flag in the future).	Day MEAN	<p>The scenario part of the time series identifier is set to the sheet name. Over time, WIS with a particular sheet name may be modified in format. The combination of sheet name and row identifier can be used to find data.</p> <p>The time series description is set to the row label.</p> <p>Data values are as stored for the WIS, which reflect the gain method used when the sheet was stored.</p>

HydroBase Time Series Types and Standard Time Series Identifier Fields (Well Data)

Data Group	Data Type Description	Location	Data Source*	Data Type	Available Intervals and Time Scale	Comments
Well	Well level (elevation)	Location identifier, based on the current data source. For example, if the data source is USGS, the location identifier will be the USGS identifier.	BJORKLUND CH2MHILL CSU CWSO DWR FOX HALAPASKA HILLIER MCCONAGHY NELSON ROBSON ROBSONBANT SCHNEIDER SEO SMITH SOUTHMETRO SPDSS USGS USGS_NAWQA WILSON *as of 2005-06-16	WellLevel	Day INST, Irregular INST	Daily data are historical measurements, often at the ends of a month. A well may have multiple identifiers. However, the identifier presented in TSTool is that corresponding to the current data source. Use StateView to see alternate identifiers for the location, to cross-reference with data outside of HydroBase. Irregular data are real-time using state station abbreviations, which do not match the identifier for historical data.

Limitations

HydroBase has the following limitations related to time series storage:

- The station and structure measurement types and time series tables defined in HydroBase do not always allow information to be determined from database records. Instead, some time series properties must be hard-coded based on the table design. For example, the *meas_type* table has a MeanTemp, MaxTemp, MinTemp types defined, but these refer primarily to the separate daily tables for such data. The *monthly_temp* table includes *avg_max_t*, *avg_min_t*, and *mean_t* fields that do not correspond one-to-one with *meas_type* values. Therefore, applications like TSTool use data types that are not specifically defined as strings in HydroBase, which have consequently been hard-coded. This is an issue with station and structure time series.
- Real-time data types in HydroBase do not directly translate to time series data types used in TSTool. An effort has been made to be as consistent as possible while using data types that can be understood by users.
- Data units are not defined consistently in tables. Some tables have a units string and others do not and the units abbreviations are not always consistent (units of “A” are often used for acre-feet and “C” for CFS). A master units table is not used in HydroBase to enforce data units consistency throughout the database.
- The time scale for time series (whether accumulated, instantaneous, or mean) is not automatically determined from the data type and interval. Users must understand how to interpret the data, in particular when changing the data interval.

Appendix: RiverWare Input Type

2008-09-03

Overview

RiverWare is a river and reservoir model developed by the Center for Advanced Decision Support for Water and Environmental Systems (CADSWES) at the University of Colorado. RiverWare uses data management interfaces (DMIs) to read time series data from various formats at run-time. The format described in this appendix is a standard time series format that is imported into the RiverWare data sets and can be output during runs. The example below shows the format of a file. Refer to the **RiverWare Data Management Interface** documentation for more information. Important comments about the file format are:

- The file is divided into a header section (top) and data section (bottom). Comments can occur anywhere in the file and are lines starting with #.
- The data period is defined by the `start_date` and `end_date` keywords. Date/times must include hours and minutes regardless of the date/time precision (the more precise information is ignored if not needed). For day, month, and year interval data, specify 24 : 00 at the end of the line.
- The data interval is defined by the `timestep` keyword and consists of an integer multiplier and a base interval string, separated by a space. Recognized intervals are HOUR, DAY, WEEK, MONTH, and YEAR.
- The data units are specified using the `units` keyword and are the units after the scale is applied. The `scale` keyword indicates a value that should be applied to the data values to result in the specified units. For example, a data value of 1.5 with units of cfs and a scale of 1000 will result of a value of 1500 cfs in memory.
- Optional `set_units` and `set_scale` keywords may be used similar to `units` and `scale` to indicate the units and scale to be converted to when data are read. These properties can be written by TSTool's `writeRiverWare()` command but currently are not evaluated by TSTool when reading data.

The following example illustrates the format of a RiverWare file.

```
# Comments
start_date: 1903-01-01 06:00
end_date: 2001-12-31 24:00
timestep: 6 HOUR
scale: 1
set_scale: 1
units: ft
set_units: ft
1356.00
1356.00
1356.00
1356.00
1356.00
NaN
NaN
...
```

RiverWare Files and Standard Time Series Properties

The standard time series identifier for RiverWare time series files is of the form:

`Location..DataType.Interval~RiverWare~PathToFile`

RiverWare time series files contain limited information to assign to standard time series properties. The following assignments are made:

- The location part of the identifier is taken from the first part of the file name. It is assumed that the file name is of the form *ObjectName.SlotName*.
- The data source part of the time identifier is left blank.
- The data type is taken from the second part of the file name. It is assumed that the file name is of the form *ObjectName.SlotName*.
- The data interval is determined from the timestep property in the file.
- The data units are determined from the `units` property in the file. Currently the `set_units` property is not evaluated when reading data.
- The missing data value is assigned to NaN (not a number).
- The description is set to the location, a comma, followed by the data type.

Limitations

RiverWare files have the following limitations:

- RiverWare time series files require that units be spelled exactly as required by RiverWare, including upper/lower case. TSTool currently does not know about RiverWare units and therefore commands like `writeRiverWare()` must be used to verify that the units are correct for RiverWare.
- Only one time series can be saved in a file (other RiverWare files support multiple time series and may be supported in the future).
- RiverWare files do not store the data type or location information for the time series. These values are assigned from the file name, as described above. Relying on a file name convention may cause errors if the convention is not followed.
- Data lines do not contain the date. Therefore, it is difficult to use the files in other applications without first assigning dates for all the values.

Appendix: StateCU Input Type

2004-05-27, Acrobat Distiller

Overview

The StateCU time series input type corresponds to the file formats used by the State of Colorado's StateCU consumptive use model, including:

- Crop pattern time series file, yearly (*.cds)
- Irrigation water requirement, formatted for StateMod (*.ddc)
- Historical direct diversions, monthly (*.ddh) and daily (*.ddd) (in StateMod format but treated as StateCU input when used with a StateCU data set)
- Irrigation practice time series, yearly (*.ipy)
- Irrigation water requirement (IWR, *.iwr) and water supply limited (WSL, *.wsl) output report files, including monthly and yearly values
- Frost dates, yearly
- Precipitation, monthly (in StateMod format)
- Temperature, monthly (in StateMod format)

See also the StateMod input type, which corresponds to StateMod input files, and the StateModB input type, which corresponds to StateMod binary output files.

See the StateCU Documentation for a complete description of StateCU input files. Refer to the **StateMod Documentation** for files that use the StateMod file format.

Important comments about the StateCU file formats are:

- Input files are divided into a header section (top) and data section (bottom). Comments can occur only at the top and are lines that begin with #.
- One or more time series can be stored in a file.
- Consistency in the order and number of the stations is required for each year of data, within the file.
- Other than comments, input files are fixed-format, compatible with FORTRAN applications. See the **StateCU Documentation** and **StateMod Documentation** for field specifications.
- Input file formats are optimized to allow a full year of data to be read for the entire data set. Reading a time series for a single location for the full period requires reading through the entire file.
- The precision of data values may be controlled by software, resulting in more or fewer fractional digits. This may lead to round-off differences when comparing raw data values output by the software.

StateCU Files and Standard Time Series Properties

The standard time series identifier for StateCU files is of the form:

`Location.StateCU..Month~StateCU~PathToDDCFile` (for *DDC* file)
`Location.StateCU.CropArea-AllCrops.Year~StateCU~PathToIWRReport` (for IWR report acreage)
`Location.StateCU.IWR.Month~StateCU~PathToIWRReport` (for IWR report monthly IWR)
`Location.StateCU.IWR.Year~StateCU~PathToIWRReport` (for IWR report yearly IWR)
`Location.StateCU.IWR_Depth.Year~StateCU~PathToIWRReport` (for IWR report IWR depth)
`Location.StateCU.CropArea-AllCrops.Year~StateCU~PathToWSLReport` (for WSL report acreage)
`Location.StateCU.WSL.Month~StateCU~PathToWSLReport` (for WSL report monthly WSL)
`Location.StateCU.WSL.Year~StateCU~PathToWSLReport` (for WSL report yearly WSL)
`Location.StateCU.WSL_Depth.Year~StateCU~PathToWSLReport` (for WSL report WSL depth)
`Location.StateCU.DataType.Interval~StateCU~PathToFile` (historical time series data files)

StateCU files contain limited header information (e.g., period of record but no data type). Time series properties are set using the following guidelines:

- For input files, the location part of the time series identifier is taken from the identifier field in the data records (from the first year of data). A change in the year indicates that all time series have been identified. For output files, the location is identified by lines that start with an underscore (this allows the StateCU interface to search for identifiers in output).
- The source part of the time series identifier is set to `StateCU` or blank.
- The data type may not be assigned because it is not defined in the file (e.g., temperature and precipitation time series). Currently no interpretation of the file name extension occurs. Some specific applications may set the data type, based on reading a StateCU data set response file (and therefore knowing the specific contents of the file).
- The data interval is assigned as `Day`, `Month`, or `Year` based on the file format (determined automatically).
- The scenario is typically not assigned.
- The input type part of the time series identifier is set to `StateCU`, indicating the file format. Software will use the interval and/or examine the file contents to verify whether the data are in daily or monthly format.
- The input name part of the time series identifier is set to the file name, either as the full path or a relative path to the working directory.
- The units are assigned to those indicated in the file header or based on the file type.
- The missing data value is assigned to `-999.0`.
- The description is set to the same value as the location. A verbose description can typically be determined by cross-referencing the identifier with another StateCU data file (e.g., `CU Locations, *.str`).
- The period is set based on the header information.

Limitations

StateCU files have the following limitations:

- The formats of the files do not facilitate extracting one time series from the file. Software has been optimized to perform this within current constraints.
- Some time series properties are not explicitly included in StateCU files (e.g., data type). Therefore, general software like TSTool may not be able to provide default information. For example, a graph may show multiple time series with nearly the same legend text because more detailed information cannot be defaulted. The following has not been implemented but may be in the future: DDC file data type = IWR.
- Although the complete output report files contain all values needed to evaluate water balance, these values are not available in files that can be easily read as time series. Currently the verbose reports are not available for reading as part of the StateCU input type.

This page is intentionally blank.

Appendix: StateCUB Input Type

(StateCU Binary Output Files)

2008-08-27

Overview

The StateCUB time series input type corresponds to the file format used by the State of Colorado's StateCU consumptive use model, in particular the binary output file. These files contain important water balance information for every location in the model. The following table summarizes the contents of the binary files and corresponding text report files (all files can be large for large data sets).

Node Type	Monthly Binary File	Monthly Report File
CU locations	*.bd1	*.dwb

The following documentation describes the format of the BD1 binary file. See the **StateCU Documentation** for a complete description of StateCU output files. The following is a summary to explain how TSTool handles the format.

Unlike the StateMod binary files, the StateCU BD1 file does not have a fixed length record throughout the file. Different sections of the file have fixed length segments, depending on the contents of the section. The main sections and their format are described below, using terminology consistent with StateCU.

Header Records (File Metadata)

Field	Data	Type	Description
1	NumStr	integer	Number of structures (CU locations).
2	NumTS	integer	Number of time steps. The data period for the file is determined from first time series record plus (NumTS – 1).
3	NumStrVar	integer	Number of variables associated with each structure.
4	NumTSVar	integer	Number of time series variables (parameters) associated with each structure.
5	NumTSA	integer	Number of time steps in a year (1, 12, 365), although 12 (monthly data) is currently the only supported value.

Structure Header Records (Structure Metadata)

Repeat the following for NumStrVar:

Field	Data	Type	Description
1	StructureVarType	char(1)	The type of the structure variable: R for floating point number, I for integer, C for character string.
2	StructureVarLen	integer	Length of structure variable in bytes.
3	StructureVarName	char(24)	Structure variable name.
4	StructureVarInReport	integer	Whether the structure is in the DWB report file.
5	StructureVarReportHeader	char(60)	The report header to use for the structure variable.

Time Series Header Records (Time Series Metadata)

Repeat the following for NumTSVar:

Field	Data	Type	Description
1	TimeSeriesVarType	char(1)	The type of the time series variable: R for floating point number, I for integer, C for character string.
2	TimeSeriesVarLen	integer	Length of time series variable in bytes.
3	TimeSeriesVarName	char(24)	Time series variable name.
4	TimeSeriesVarInReport	integer	Whether the structure is in the DWB report file.
5	TimeSeriesVarUnits	char(10)	The units for the time series.

Structure Variable Data Records

Repeat the following for NumStr, and for each NumStrVar. The order of the variables is not fixed; however, the “Structure Index” variable contains a numeric identifier that is used to sort the structures to lookup structures when reading the time series.

Field	Data	Type	Description
1	StructureVarValue	As per metadata.	The values of the structure variables, including “Structure Index” (1+), “Structure ID”, “Structure Name”. Additional variables may be added later. The index is used to create a sorted list of structure identifiers and names for applications like TSTool.

Time Series Data Records

Repeat the following looping on NumStr, then NumTimeSteps, and then NumTimeSeriesVar. The order of time series variables is the same for all structures and throughout the entire file (variable “X” will always be in the same position in the inner loop). The order of the structures may not agree with the order of the metadata from above. The “Structure Index” variable in the time series records is used to map the time series to the structure identified in the metadata above.

Field	Data	Type	Description
1	TimeSeriesVarValue	As per metadata.	The values of the time series variables, including “Structure Index” (matching “Structure Index” from the structure data), “Year” (Calendar 4-digit), “Month Index” (1-12), and variable names for values at each time step. Additional variables may be added later.

A visual representation of data is as follows (note that Structure 1 is an internal looping representation and the actual structure is identified by the “Structure Index” variable for the time series):

```
Structure 1
  Timestep 1
    Variable 1
    ...
    Variable NumStrVar
  Timestep 2
    Variable 1
    ...
```

```

        Variable NumStrVar
    ...
    TimeStep NumTS
Structure 2
    Timestep 1
        Variable 1
    ...
        Variable NumStrVar
    Timestep 2
        Variable 1
    ...
        Variable NumStrVar
    ...
    TimeStep NumTS
...
Structure NumStr

```

The order of the structures in the time series data block may not be the same as that in the header metadata due to the constraints of the StateCU model and how it writes each section during different phases of execution. Therefore, at initialization, the “Structure Index” variable value for each time series is read for the first timestep of each structure to determine the mapping of the structure in the time series data block with that in the main header.

Some time series variables are integers (e.g., the year and month) and some are characters (e.g., the month name and model flags). The integer variable “Year” has the same value for 12 monthly time steps and then increases by one. The variable “Month Index” repeats the values 1 – 12 through the period of the time series. Only floating point parameters are read by default. In the future, integer and character time series may be allowed or the character values may be translated to a lookup table of numbers.

StateCU BD1 Files and Standard Time Series Properties

The standard time series identifier for StateCUB binary time series is of the form:

```
Location.StateCU.DataType.Interval~StateCUB~PathToFile
```

Time series properties are set using the following guidelines:

- The location part of the time series identifier is taken from the structure identifier field in the data.
- The data source part of the time series identifier is set to `StateCU`, because StateCU has created the output time series.
- The data type is assigned as the variable (parameter) name described above – See the StateCU documentation for more information.
- The data interval is assigned as `Month`.
- The scenario is set to blank (not used).
- The input type is set to `StateCUB`.
- The input name is set to the name of the file.
- The units are determined from the time series variable metadata.
- The missing data value is assigned to `-999.0`.
- The description is set to the structure name.
- The period is set to the information in the first time series record incremented by the number of timesteps in the file (minus one). Current the file only contains calendar year data (January to December).

Limitations

StateCU binary files have the following limitations:

- The file does not contain a format version; therefore, it is difficult for software to handle changes in the file format. However, the current format is designed to allow for changing structure and time series parameters without changing the file format.
- The file does not contain header information indicating the source of the file (e.g., the creation date, user, directory, StateCU response file, command line). Therefore, it is difficult to know with certainty how a file was created.
- Leap years are not explicitly handled with 29 days during model calculations. Therefore there may be some loss of precision as data are processed through the model. Refer to the StateCU documentation for more information on how values are calculated.

Appendix: StateMod Input Type

2004-07-27, Acrobat Distiller

Overview

The StateMod time series input type corresponds to the file format used by the State of Colorado's StateMod model, including standard daily, monthly, average monthly (referred to as annual in the StateMod documentation) file formats. See also the StateModB input type, which corresponds to StateMod binary output files and the StateCU input type, which corresponds to the State of Colorado's StateCU consumptive use model.

The following example illustrates the format of the three main file formats. See the **StateMod Documentation** for a complete description of StateMod input files. Important comments about the file format are:

- The file is divided into a header section (top) and data section (bottom). Comments can occur only at the top and are lines that begin with #.
- One or more time series can be stored in a file.
- Consistency in the order and number of the stations is required for each year of data, within the file.
- Other than comments, the file is fixed-format, compatible with FORTRAN applications. See the **StateMod Documentation** for field specifications.
- The format is optimized to allow a full year of data to be read for the entire data set. Reading a time series for a single location for the full period requires reading through the entire file.
- In addition to the required values, a total/average value is accepted as the far-right value on each data line. This value may be ignored by applications (it can be computed from the data values on the line if necessary).
- The precision of data values may be controlled by software, resulting in more or fewer fractional digits. This may lead to round-off differences when comparing raw data values with the total/average in the optional end column.

```
# StateMod time series files can have 3 main forms (monthly, average monthly, daily) as
# described below. The order of time series is important for
# some files (e.g., order of diversion time series should match order of
# diversion stations in .dds file); however, StateMod is being updated over
# time to remove this requirement). Different StateMod input files have
# slight variations on the general format (e.g., the reservoir target file
# has two time series for each reservoir for minimum and maximum targets).
# Missing data are typically indicated by -999.
# The generic extension for StateMod time series files is .stm, although specific
# extensions are used in a StateMod data set.
#
# 1) This is an example of a StateMod monthly time series for water year data:
#
# Comments are lines at the top of the file starting with the # character.
# The header may contain software-generated comments about the time series.
# The remainder of the file is fixed format, with the first non-comment
# line being a header with the following elements (i5,1x,i4,5x,i5,1x,i4,a5,a5):
#
# Beginning month (1=Jan)
# Beginning year (4-digit)
# Ending month
# Ending year
# Data units (AF/M, ACFT, CFS or ""), where rates are for diversions and
# flow, and volume is for reservoir contents. Units are not used for
# dimensionless data (like weight or percent).
```

```

# Year type (CYR=calendar, WYR=water, IYR=irrigation)
#
# Data lines then follow with:
# Year Station 12-monthly-values year-total/average (i4, lx, a12, 12f8, f10)
# The year value is optional and is generally not read as input but is
# computed for output. The year in data lines corresponds to the calendar type.
# An example follows:
10/1926 - 9/1998 ACFT WYR
1927 08236000 1229.8 892.6 922.3 737.9 555.4 922.3 7049.4 32263.6
31000.1 14541.0 5662.9 8326.7 104104.0
1927 08235250 -999.0 -999.0 -999.0 -999.0 -999.0 -999.0 -999.0 -999.0 -
999.0 -999.0 -999.0 -999.0 0.0
1927 08235700 -999.0 -999.0 -999.0 -999.0 -999.0 -999.0 -999.0 -999.0 -
999.0 -999.0 -999.0 -999.0 0.0
1927 08236500 1047.3 595.1 614.9 614.9 555.4 1900.2 6769.7 31226.2
20338.8 14777.1 9465.3 4476.8 92381.5
...

#
# 2) This is an example of a StateMod average monthly time series for water year data:
#
# The average monthly time series is a pattern of twelve monthly values
# that are applied for each year in the period.
# The format is exactly the same as a monthly time series; however, the
# years in the header should be set to zero and year and month are ignored in data rows
# and can therefore be blank.
#
# An example follows:
10/ 0 - 9/ 0 ACFT WYR
08236000 1229.8 892.6 922.3 737.9 555.4 922.3 7049.4 32263.6
31000.1 14541.0 5662.9 8326.7 104104.0
08235250 -999.0 -999.0 -999.0 -999.0 -999.0 -999.0 -999.0 -999.0 -
999.0 -999.0 -999.0 -999.0 0.0
08235700 -999.0 -999.0 -999.0 -999.0 -999.0 -999.0 -999.0 -999.0 -
999.0 -999.0 -999.0 -999.0 0.0
08236500 1047.3 595.1 614.9 614.9 555.4 1900.2 6769.7 31226.2
20338.8 14777.1 9465.3 4476.8 92381.5
...

#
# 3) This is an example of a StateMod daily time series for water year data:
#
# The daily time series is similar to the monthly time series except that
# a year and month are included on the data lines and 28, 30, or 31 daily
# data values can occur on each line (end values ignored, depending on month).
# The data format is (i4, i4, lx, a12, 31f8, f8). The month total/average
# is optional and is generally read as input but is computed for output.
# Regardless of the calendar type in the header, the year and month in data records use
# calendar year (month 1 = January).
#
# An example follows:
10/1926 - 9/1998 ACFT WYR
1926 10 08236000 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -
999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -
999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -
0.00 0.00
...
1927 4 08236000 38.00 42.00 42.00 67.00 90.00 90.00 100.00 118.00
93.00 80.00 93.00 80.00 80.00 80.00 80.00 80.00 68.00 80.00 68.00
68.00 80.00 80.00 106.00 136.00 170.00 229.00 250.00 296.00 322.00 348.00
0.00 114.65
1927 4 08235250 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -
999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -
999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00 -
0.00 0.00
...

```


StateMod Files and Standard Time Series Properties

The standard time series identifier for StateMod files is of the format:

`Location...Interval~StateMod~PathToFile`

StateMod files contain limited header information. Time series properties are set using the following guidelines:

- The location part of the time series identifier is taken from the identifier field in the data records (from the first year of data). A change in the year indicates that all time series have been identified.
- The data source part of the time series identifier is set to `StateMod` or blank. In the past this information was used to indicate the input type (file format) in the time series identifier; however, the new input type notation has a specific field for the input type and therefore data source can be used more appropriately. In the future, it may be possible to pass along the original input source but this information cannot currently be saved in the StateMod file format.
- The data type is often not assigned because it is not defined in the file. Currently no interpretation of the file name extension occurs. Some specific applications (e.g., the StateMod GUI) may set the data type, based on reading a StateMod data set response file (and therefore knowing the specific contents of the file).
- The data interval is assigned as `Day` or `Month` based on the file format (determined automatically).
- The scenario is typically not assigned. Older software may use the scenario to store the file name; however, the new time series identifier notation stores the file name as the input name field (see below).
- The input type part of the time series identifier is set to `StateMod`, indicating the file format. Software will use the interval and/or examine the file contents to verify whether the data are in daily or monthly format.
- The input name part of the time series identifier is set to the file name, either as the full path or a relative path to the working directory.
- The units are assigned to those indicated in the file header.
- The missing data value is assigned to `-999.0`.
- The description is set to the same value as the location. A verbose description can typically be determined by cross-referencing the identifier with another StateMod data file (e.g., diversion stations).
- The period is set based on the header information.

Limitations

StateMod files have the following limitations:

- The format of the does not facilitate extracting one time series from the file. Software has been optimized to perform this within current constraints.
- Some time series properties are not explicitly included in StateMod files (e.g., data type). Therefore, general software like TSTool may not be able to provide default information. For example, a graph may show multiple time series with nearly the same legend text because more detailed information cannot be defaulted.
- If two time series for the same station are stored in the same file (e.g., reservoir maximum and minimum targets), there is no way to uniquely identify the two time series. The application or user must understand the file type and data organization. Some specific software (e.g., StateMod GUI) may be able to recognize the specific format.

This page is intentionally blank.

Appendix: StateModB Input Type

(StateMod Binary Output Files)

2004-07-27, Acrobat Distiller

Overview

The StateModB time series input type corresponds to the file format used by the State of Colorado's StateMod model, in particular the binary FORTRAN direct access output files. These files contain important water balance information for every node in the model network. The following table summarizes the contents of the binary files and corresponding text report files (all files can be large for large data sets):

Node Type	Monthly Binary File	Monthly Report File	Daily Binary File	Daily Report File
Diversion	*.b43	*.xdd	*.b49	*.xdy
Instream flow	*.b43	*.xdd	*.b49	*.xdy
Reservoir	*.b44	*.xre	*.b50	*.xry
Stream gage and Stream estimate	*.b43	*.xdd	*.b49	*.xdy
Well	*.b42	*.xwe	*.b65	*.xwy

The following documentation describes the format of the B43 binary file. Other files are similar. See the **StateMod Documentation** for a complete description of StateMod output files. Important comments about the file format are:

- The file is generated by StateMod as a direct access binary file with fixed-length records. The record length is 140 bytes.
- The file is divided into a header section (top) and data section (bottom).
- The format is optimized to allow a full year of data to be read for the entire data set. Efficiently reading a time series for a single location for the full period requires reading appropriate lines of the file using direct access. Because the file is binary and consistent for a given data set, file reads can be optimized.
- The data period and the calendar year type are consistent with the StateMod control file.
- All character strings are left justified and are padded with spaces. Therefore, software that reads the file should trim trailing spaces after reading the strings.
- River node identifiers in record 5 are included for all nodes in the network and data records (record 11) follow this order. Subsequent lists for various node types are a subset of the list in record 5 and have data items to reference the position in the river node list. Time series are queried using the identifiers in records 6+. However, the river node position is actually used to retrieve data in the file.

The B43 binary file contains the following records:

Record	Field	StateMod Variable	Type	Description
1	1	iystro	integer	Beginning year of simulation, for year type in StateMod control file.
	2	iyend0	integer	Ending year of simulation, for year type in StateMod control file.
2	1	numsta	integer	Number of river nodes.
	2	numdiv	integer	Number of direct diversions.
	3	numifr	integer	Number of instream flows.
	4	numres	integer	Number of reservoirs.
	5	numown	integer	Number of reservoir owners.
	6	nrsact	integer	Number of active reservoirs.
	7	numrun	integer	Number of base flow nodes.
	8	numdivw	integer	Number of diversion structures with wells.
	9	numdxw	integer	Number of well only structures.
3	1	xmonam(14)	Each is char(4).	Month names corresponding to the calendar type for the simulation. This information is provided as a convenience for data processing. For example, if the year type is WYR (water year), xmonam(1) is 'OCT'. The 13 th value is 'TOT' and the 14 th value is 'AVE'.
4	1	mthday(12)	Each is integer.	Number of days per month, corresponding to the calendar type for the simulation. This information is provided as a convenience for data processing and to convert daily data values to monthly. For example, if the year type is WYR (water year), mthday(1) is 31 for October. The number of days in February is typically 28 and is used for all data processing, regardless of whether a year is a leap year.
5 Repeat record for numsta	1	j	integer	Counter for record type 5.
	2	cstaid(j)	char(12)	River node identifiers.
	3	stanam(j)	real(6)	River node names.
6 Repeat record for numdiv	1	j	integer	Counter for record type 6.
	2	cdivid(j)	char(12)	Diversion identifier.
	3	divnam(j)	real(6)	Diversion name.
	4	idvsta(j)	integer	River node position (1+) to allow cross-reference with river nodes.
7 Repeat record for numifr	1	j	integer	Counter for record type 7.
	2	cifrid(j)	char(12)	Instream flow identifier.
	3	xfrnam(j)	real(6)	Instream flow name.
	4	ifrsta(j)	integer	River node position (1+) to allow cross-reference with river nodes.
8 Repeat record for numres	1	j	integer	Counter for record type 8.
	2	cresid(j)	char(12)	Reservoir identifier.
	3	resnam(j)	real(6)	Reservoir name.
	4	irssta	integer	River node position (1+) to allow cross-reference with river nodes.

Record	Field	StateMod Variable	Type	Description		
9 Repeat record for numrun	1	j	integer	Counter for record type 9.		
	2	crunid(j)	char(12)	Base flow node identifier.		
	3	runnam(j)	real(6)	Base flow node name.		
	4	irusta(j)	integer	River node position (1+) to allow cross-reference with river nodes.		
10 Repeat record for numdivw	1	j	integer	Counter for record type 10.		
	2	cdividw(j)	char(12)	Well identifier.		
	3	divnamw(j)	real(6)	Well name.		
	4	idvstw(j)	integer	River node position (1+) to allow cross-reference with river nodes.		
11 Repeat record for every river node numsta, for every month of the simulation. See the StateMod documentation for a full description of parameters. Parameters are grouped as shown in the *.xdd file.	1	dat(1)	real	Demand	Total_Demand	
	2	dat(2)	real	Demand	CU_Demand	
	3	dat(3)	real	Water Supply	From_River_By_Priority	
	4	dat(4)	real	Water Supply	From_River_By_Storage	
	5	dat(5)	real	Water Supply	From_River_By_Exchange	
	6	dat(6)	real	Water Supply	From_Well	
	7	dat(7)	real	Water Supply	From_Carrier_By_Priority	
	8	dat(8)	real	Water Supply	From_Carrier_By_Storage	
	9	dat(9)	real	Water Supply	Carried_Water	
	10	dat(10)	real	Water Supply	From_Soil	
	11	dat(11)	real	Water Supply	Total_Supply	
	12	dat(12)	real	Shortage	Total_Short	
	13	dat(13)	real	Shortage	CU_Short	
	14	dat(14)	real	Water Use	Consumptive_Use	
	15	dat(15)	real	Water Use	To_Soil	
	16	dat(16)	real	Water Use	Total_Return	
	17	dat(17)	real	Water Use	Loss	
	18	dat(18)	real	Station In/Out	Upstream_Inflow	
	19	dat(19)	real	Station In/Out	Reach_Gain	
	20	dat(20)	real	Station In/Out	Return_Flow	
	21	dat(21)	real	Station In/Out	Well_Depletion	
	22	dat(22)	real	Station In/Out	To_From_GW_Storage	
	23	dat(23)	real	Station Balance	River_Inflow	
	24	dat(24)	real	Station Balance	River_Divert	
	25	dat(25)	real	Station Balance	River_By_Well	
	26	dat(26)	real	Station Balance	River_Outflow	
	27	dat(27)	real	Available Flow	Available_Flow	
	28	dat(28)	real	Structure type (Na): <ul style="list-style-type: none">< 0 = Baseflow node (e.g., -10001 indicates a diversion that is a baseflow node).0 = Well only.1-5000 = Diversion5001 – 7500 = Instream flow7501 – 10000 = Reservoir		
	29	dat(29)	real	Number of structures at this node (typically 1).		

StateMod B43 Files and Standard Time Series Properties

The standard time series identifier for StateMod binary time series is of the form:

`Location.StateMod.DataType.Interval~StateModB~PathToFile`

Time series properties are set using the following guidelines:

- The location part of the time series identifier is taken from the identifier field in the data. The identifier for the specific node type (e.g., diversion) is used, not the river node identifier. The river node identifier is often the same as for the specific node type, but this is not a requirement within StateMod.
- The data source part of the time series identifier is set to `StateMod`, because StateMod has created the output time series.
- The data type is assigned as the parameter name (see record 11 above, without using the group).
- The data interval is assigned as `Month` or `Day`, depending on the file extension.
- The scenario is set to `blank`.
- The input type is set to `StateModB`.
- The input name is set to the name of the file.
- The units for daily data are assigned as `CFS`. The units for monthly data in the files are average `CFS` for the month and are converted to `ACFT`, assuming a constant number of days per month, as read from record 4. February normally has 28 days per month in the header and therefore leap years have one fewer days than actual.
- The missing data value is assigned to `-999.0`.
- The description is set to the node name.
- The period is set based on the header information in record 1 (for the year) and record 3 (to determine the start and end months, based on the calendar type).

Limitations

StateMod binary files have the following limitations:

- The file does not contain a format version; therefore, it is difficult for software to handle changes in the file format.
- The file does not contain header information indicating the source of the file (e.g., the creation date, user, directory, StateMod response file, command line). Therefore, it is difficult to know with certainty how a file was created.
- Leap years are not explicitly handled with 29 days.
- Baseflow nodes in record 9 may have the same identifier as other nodes because any node can be a baseflow node. This can be confusing since software may list the node in more than one list. The software that reads the file filters out duplicate time series identifiers to try to resolve this problem.
- This documentation is limited in that it presents the file format only for the `*.b43` file. Additional documentation may be added in the future.

Appendix: USGSNWIS Input Type

2004-07-27, Acrobat Distiller

Overview

The USGSNWIS time series input type corresponds to the United States Geological Survey (USGS) National Water Information System (NWIS) format. A number of formats are available but currently only the surface water daily format is supported. Data files can be created by saving USGS web site data to a text file. The example below shows the format of a daily surface water file. Important comments about the file format are:

- The file is divided into a header section (top) and data section (bottom). Comments can occur only at the top and are lines that begin with #.
- Optional data flags are saved with the data values, if available (e.g., e indicates estimated data). Applications like TSTool may include features to use the data flags.
- HTML remnants may be present at the end of the file. These lines are stripped out during time series processing.

The following example illustrates the format of a USGS NWIS file.

```
#
# U.S. Geological Survey
# National Water Information System
# Retrieved: 2002-01-28 13:35:25 EST
#
# This file contains published daily mean streamflow data.
#
# This information includes the following fields:
#
# agency_cd    Agency Code
# site_no      USGS station number
# dv_dt        date of daily mean streamflow
# dv_va        daily mean streamflow value, in cubic-feet per-second
# dv_cd        daily mean streamflow value qualification code
#
# Sites in this file include:
# USGS 03451500 FRENCH BROAD RIVER AT ASHEVILLE, NC
#
#
agency_cd      site_no dv_dt   dv_va   dv_cd
5s           15s      10d      12n      3s
USGS 03451500 1895-10-01 740
USGS 03451500 1895-10-02 740
...
USGS 03451500 1985-01-20 1100 e
USGS 03451500 1985-01-21 1100 e
USGS 03451500 1985-01-22 1100 e
...
USGS 03451500 2000-09-28 675
USGS 03451500 2000-09-29 597
USGS 03451500 2000-09-30 550
<font face="Arial" size=2>
<p>Microsoft VBScript runtime </font> <font face="Arial" size=2>error '800a01a8'</font>
<p>
<font face="Arial" size=2>Object required: 'db'</font>
<p>
<font face="Arial" size=2>/ctp_workgroup/cgi-bin/includes/Inc_htm_utils.asp</font>
<font face="Arial" size=2>, line 217</font> <font face="Arial" size=2>
<p>Microsoft VBScript runtime </font> <font face="Arial" size=2>error '800a01a8'</font>
<p><font face="Arial" size=2>Object
```

USGSNWIS Files and Standard Time Series Properties

The standard time series identifier for USGS NWIS time series is of the form:

```
Location.DataSource.DataType.Interval~USGSNWIS~PathToFile
```

It is difficult to automatically assign standard time series properties from a USGS NWIS file. The limited support of this file format assumes the following:

- The location part of the time series identifier is taken from the second field (`site_no`) in the data records.
- The source part of the time series identifier is taken from the first field (`agency_cd`) in the data records.
- The data type is assigned as `Streamflow` (interpretation of the verbose `dv_va` field in the header is not implemented).
- The data interval is assigned as `1Day` (interpretation of the verbose `dv_va` field in the header is not implemented).
- The input type is set to `USGSNWIS`, indicating the format of input.
- The input name is set to the absolute or relative path to the file.
- The Units are assigned as `CFS`.
- The missing data value is assigned to `-999.0` (gaps in data records will result in this value).
- The description is set to the information after the `Sites in this file include:` line. It is assumed that only one time series per file is used.

Limitations

USGSNWIS files have the following limitations:

- Riverside Technology, inc. is working to support the standard USGS file format(s). Limited information is available for the file specifications. Currently only the daily surface water format has been tested.
- Additional specific limitation will be listed when file format specifications are fully determined.
- The period for the data is not available in the file header. Therefore the period is determined from the first and last dates in the data records. This introduces a slight performance penalty.
- Although data flags are read in for use by applications, no standard flag values are enforced (the end user will need to know the meaning of the flags to use them properly).

Appendix: TSView - Time Series Viewing Tools

Color, 2006-09-28, Original Maintained with TSTool, Acrobat Distiller

Overview

Time Series Terminology

Time Series Properties Interface

- Time Series Properties – General

- Time Series Properties – Comments

- Time Series Properties – Period

- Time Series Properties – Limits

- Time Series Properties – History

- Time Series Properties – Data Flags

Time Series Traces

Time Series Views

Time Series Graph View

- Line Graph

- Line Graph – Log Y Axis

- Bar Graph

- Double Mass Curve

- Duration Graph

- Period of Record Graph

- XY-Scatter Graph

Time Series Product Properties

- Product Properties – General

- Product Properties – Titles

- Product Properties – Layout

- Graph Properties – General

- Graph Properties – Graph Type

- Graph Properties – Titles

- Graph Properties – X Axis

- Graph Properties – Y Axis

- Graph Properties – Label

- Graph Properties – Legend

- Graph Properties – Zoom

- Graph Properties – Analysis

- Graph Properties – Annotations

- Time Series Properties – General

- Time Series Properties – Graph Type

- Time Series Properties – Axes

- Time Series Properties – Symbol

- Time Series Properties – Label

- Time Series Properties – Legend

- Time Series Properties - Analysis

Changing a Graph Page Layout

Time Series Summary View

Time Series Table View

Time Series Product Reference

Overview

The TSView package contains integrated software components that can be used with software applications to enable time series viewing capabilities. The main purpose of the TSView package is to provide simple, consistent, and flexible displays that can be used in a variety of applications with little or no reconfiguration. TSView also provides features to configure and process time series products (e.g., graphs), where the time series data are stored separately from the configuration information.

The TSView package has been developed by Riverside Technology, inc., using Java technology. TSView interfaces can be embedded in Java applications and can be used in web pages either as embedded applets or stand-alone windows. TSView tools operate similarly on Microsoft Windows and UNIX operating systems.

This appendix describes general TSView features and can be used as a reference for how to configure and use TSView components. Software program documentation may include specific information about using TSView features.

Time Series Terminology

The TSView package treats time series as objects that can be read, manipulated, and output in various formats. A time series is defined as having header information (attributes) and data, which usually consists of a series of date/time versus data pairs. Internally, time series are considered to have either regular interval (equal spacing of date/time) or irregular interval (e.g., occasional observations). Regular time series lend themselves to simpler storage and faster processing because date/time information can be stored only for the endpoints. The following basic attributes are stored for each time series:

- Data interval as an interval base (e.g., Month, Hour) and multiplier (e.g., 1 for month, or 24 for hour) - in many cases, the multiplier is 1 and is not shown in output (e.g., Month rather than 1Month),
- Data type (e.g., Streamflow), which ideally can be checked to determine if a time series contains mean, instantaneous, or accumulated values,
- Units (e.g., CFS), which ideally can be used to make units conversions and look up precision for output,
- Period of record, using dates that are of an appropriate precision for the interval,
- Data limits (the maximum, minimum, etc.),
- Description (generally a station, structure, or sensor name),
- Missing data value (used internally to mark missing data and trigger data filling, often -999),
- Comments (often station comments, if available),
- Genesis history (a list of comments about how the time series was created).

In order to uniquely and consistently identify time series, a multi-part *time series identifier* is employed, having the following parts:

- Location (or location-sublocation)
- Data source
- Data type (or datatype-subdatatype)
- Data interval (time step)
- Scenario

and optionally:

- Sequence number (currently being evaluated)
- Input type
- Input name

These time series attributes are typically concatenated into a time series identifier string. The following example illustrates how the basic identifier parts can be used (without input type and name):

```
12345678.USGS.Streamflow.DAY.HIST
```

The above example identifies a USGS streamflow gage identified as location 12345678, at which historic average daily flow data are available. If possible, data types appropriate for the input type should be used to avoid confusion; however, time series file input types often do not contain a simple data type abbreviation (see the input type appendices in the **TSTool Documentation** for more information). The above example illustrates that the scenario can be used to qualify the data (in this case as historic data, HIST). The scenario is often omitted. When the scenario is used, it often indicates some specific condition (e.g., FLOOD, DROUGHT, HIST, FILLED)

The optional input type and input name are used to specify the time series input format and storage location, especially in cases where the identifier is saved in a file and the input type is needed for later processing. For example:

```
12345678.USGS.Streamflow.DAY.HIST~USGSNWIS~C:\data\12345678.txt  
12345678.USGS.Streamflow.DAY~HydroBase
```

The first example illustrates a time series identifier for a USGS National Water Information System data file. The second example illustrates the identifier for the same time series, in the HydroBase database. Using the input parts of the identifier allows software to transparently locate the data, and for the above examples, would allow the time series to be read from each input source and compared.

The use of the input type and name is being phased into TSView and related components. Input types that have been added to software more recently (e.g., as of version 05.04.00 of the TSTool application) use the new convention and older input types are being updated accordingly. The TSTool appendices that describe each input type identify issues with compatibility.

Using the above time series identifier convention omits use of time series attributes like the period of record and the units, even though these attributes could conceivably be used to distinguish between time series that are otherwise the same. Instead, it is assumed that the period of record and units can be determined from the input and do not need to be part of the identifier. If necessary, different input files can be used to further differentiate time series.

The TSView components use the time series identifiers extensively to locate and manage time series. For example, graph properties for each time series are cross-referenced to time series by using the identifiers. Perhaps most importantly, the time series identifiers as simple strings can be stored in files and can be used by a variety of software to consistently and reliably locate data for processing.

The following table summarizes important time series terminology.

Time Series Terminology (listed alphabetically)

Term	Description
<i>Data Interval</i>	Time interval between time series data values. If a <i>regular</i> time series, the interval is constant. If an <i>irregular</i> time series, the interval can vary. Intervals are represented as an optional multiplier followed by a base interval string (e.g., 1MONTH, 24HOUR) or IRREGULAR for irregular time series.
<i>Data Source</i>	A string abbreviation for a data source, which is part of the time series identifier and typically indicates the origin of the data (e.g., an agency abbreviation, or a model name if the result of a simulation).
<i>Data Type</i>	A string abbreviation for a data type, which is part of the time series identifier (e.g., Streamflow).
<i>Date/Time Precision</i>	Date/time objects used with time series have a precision that corresponds to the time series data interval. The precision is typically handled transparently but it is important that the precision is consistent (e.g., monthly data should not use date/time objects with daily precision). Displaying time series with various precision usually results in the smallest time unit being used for labels.
<i>Input Name</i>	A string input name corresponding to an input type, which is part of a time series identifier. For database input types, the name may be omitted or may be the name of the database connection (e.g., ARCHIVE). For input files, the name is typically the name of the file.
<i>Input Type</i>	A string abbreviation that indicates the input type (persistent format) for a time series, and is part of a time series identifier. This is often the name of a database (e.g., HydroBase, RiversideDB) or a standard data file format type (e.g., StateMod, MODSIM, RiverWare).
<i>Location</i>	A string identifier that is part of a time series identifier and typically identifies a time series as being associated with a location (e.g., a stream gage or sensor identifier). The location may be used with certain input types to determine additional information (e.g., station characteristics may be requested from a database table using the location).
<i>Scenario</i>	A string label that is part of a time series identifier, and serves as a modifier for the identifier (e.g., HIST for historical).
<i>Sequence Number</i>	A number indicating the sequence position of a time series in a series. For example, possible time series traces may be identified with a sequence number matching the historical year for the data. The use of sequence numbers with traces is being evaluated.
<i>Time Series Product</i>	A graph or report that can be defined and reproduced. See the Time Series Product Reference section.
<i>Time Step</i>	See Data Interval.

Time Series Properties Interface

Time series properties are displayed in a tabbed panel as appropriate in applications (e.g., the TSTool application can display the properties after time series are read and listed in the TSTool interface). Differences between time series input types may result in variations in the properties (e.g., some input types do not have descriptions for time series). The following figures describe the properties tabs. The size of each tabbed panel is set to the size of the largest tab; therefore, some tabbed panels are not completely filled.

Time Series Properties - General

TSTool - 08236000.DWR.Streamflow.Month - Properties

General | Comments | Period | Limits | History | Data Flags

Identifier: 08236000.DWR.Streamflow.Month

Identifier (with input): 08236000.DWR.Streamflow.Month~HydroBase

Alias:

Sequence Number: -1

Description: ALAMOSA RIVER ABOVE TERRACE RESERVOIR

Units (Current): ACFT

Units (Original): ACFT

☐ Is Selected

☒ Is Dirty

Close Print

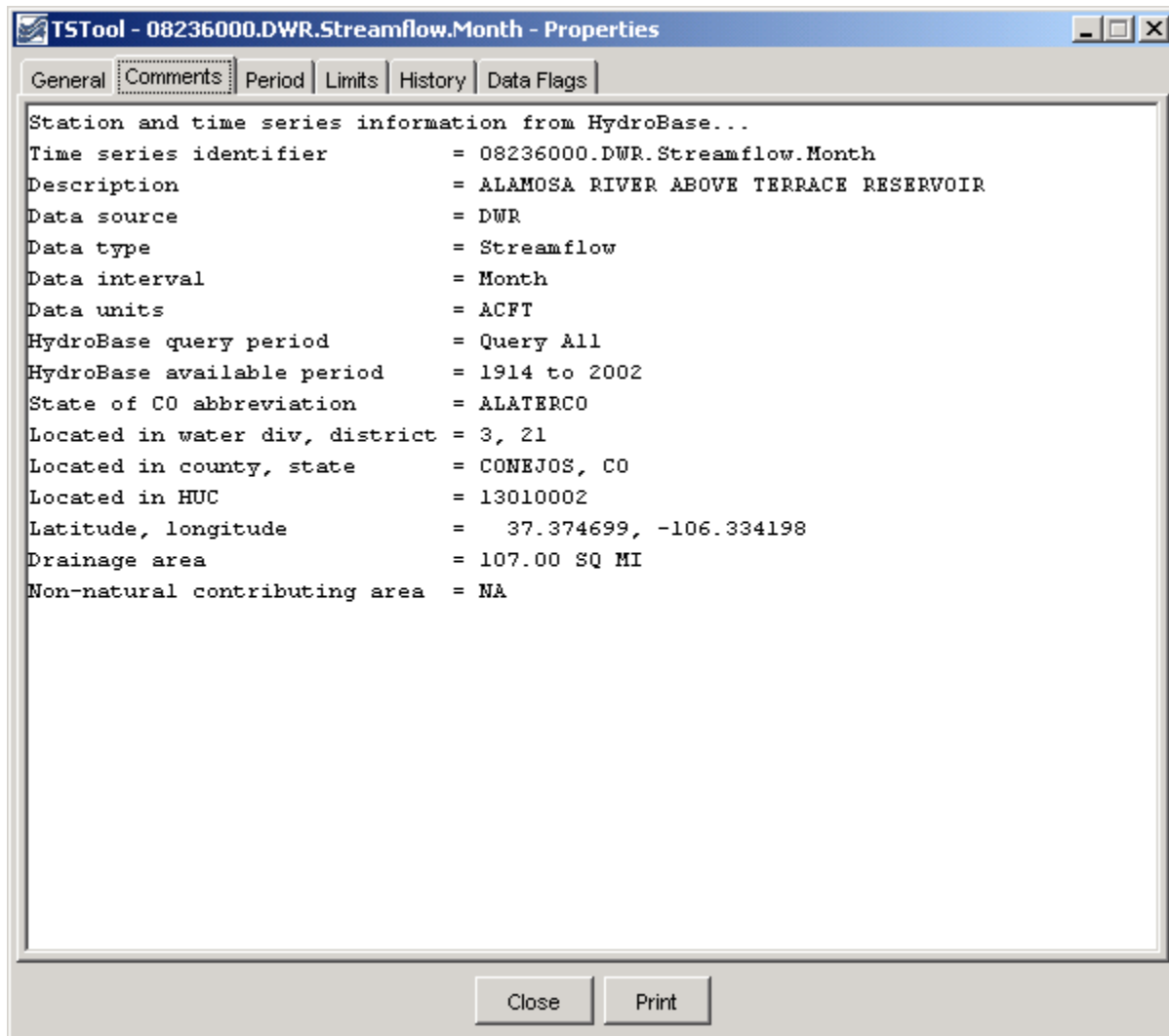
TSView_TSProps_General

Time Series Properties - General

General time series properties are as follows:

Identifier	The five-part time series identifier without the input type and name. This identifier is often used internally in applications to manage time series. See the Time Series Terminology section for a complete explanation of time series identifiers.
Identifier (with input)	The full identifier, including the input type and name (if available). The input type and name indicate the format and storage of the data.
Alias	A time series may be assigned an alias to facilitate processing (e.g., the alias is used by the TSTool application in time series commands).
Sequence Number	If the time series is part of a series of traces, the sequence number is used to identify the trace. Often it is the year for the start of the trace.
Description	The description is a mid-length phrase (i.e., longer than the location but shorter than comments) describing the time series (e.g., XYZ RIVER AT ABC).
Units (Current)	The units that are currently used for data. The units may have been converted from the original.
Units (Original)	The units in the original data source.

Time Series Properties – Comments



TSView_TSProps_Comments

Time Series Properties - Comments

Comments for time series can be created a number of ways and may be formatted specifically for an application. Common ways of creating comments are:

- Read comments from the original data source - this is ideal; however, electronic comments are often not available (e.g., the USGS previously published comments for data stations in hard copy water reports; however, comments may no longer be available electronically),
- Format comments from existing data pieces (e.g., the figure illustrates a standard set of comments for State of Colorado data, using the HydroBase input type).

Time Series Properties – Period

The screenshot shows a Windows-style dialog box titled "TSTool - 08236000.DWR.Streamflow.Month - Properties". It has five tabs: "General", "Comments", "Period" (which is selected), "Limits", and "Data Flags". The "Period" tab contains three text input fields. The first is labeled "Current (reflects manipulation):" and contains the text "1914-05 to 2002-09". The second is labeled "Original (from input):" and also contains "1914-05 to 2002-09". The third is labeled "Total Points:" and contains the number "1061". At the bottom of the dialog are two buttons: "Close" and "Print". In the bottom right corner of the window, the text "TSView_TSProps_Period" is visible.

TSView_TSProps_Period

Time Series Properties - Period

Properties related to the period are as follows:

***Current
(reflects
manipulation)***

The current period is used to allocate computer memory for the time series data. This period may be set by an application (e.g., when creating model input files a specific period may be used). The precision of the date/time objects should generally be consistent with the time series data interval.

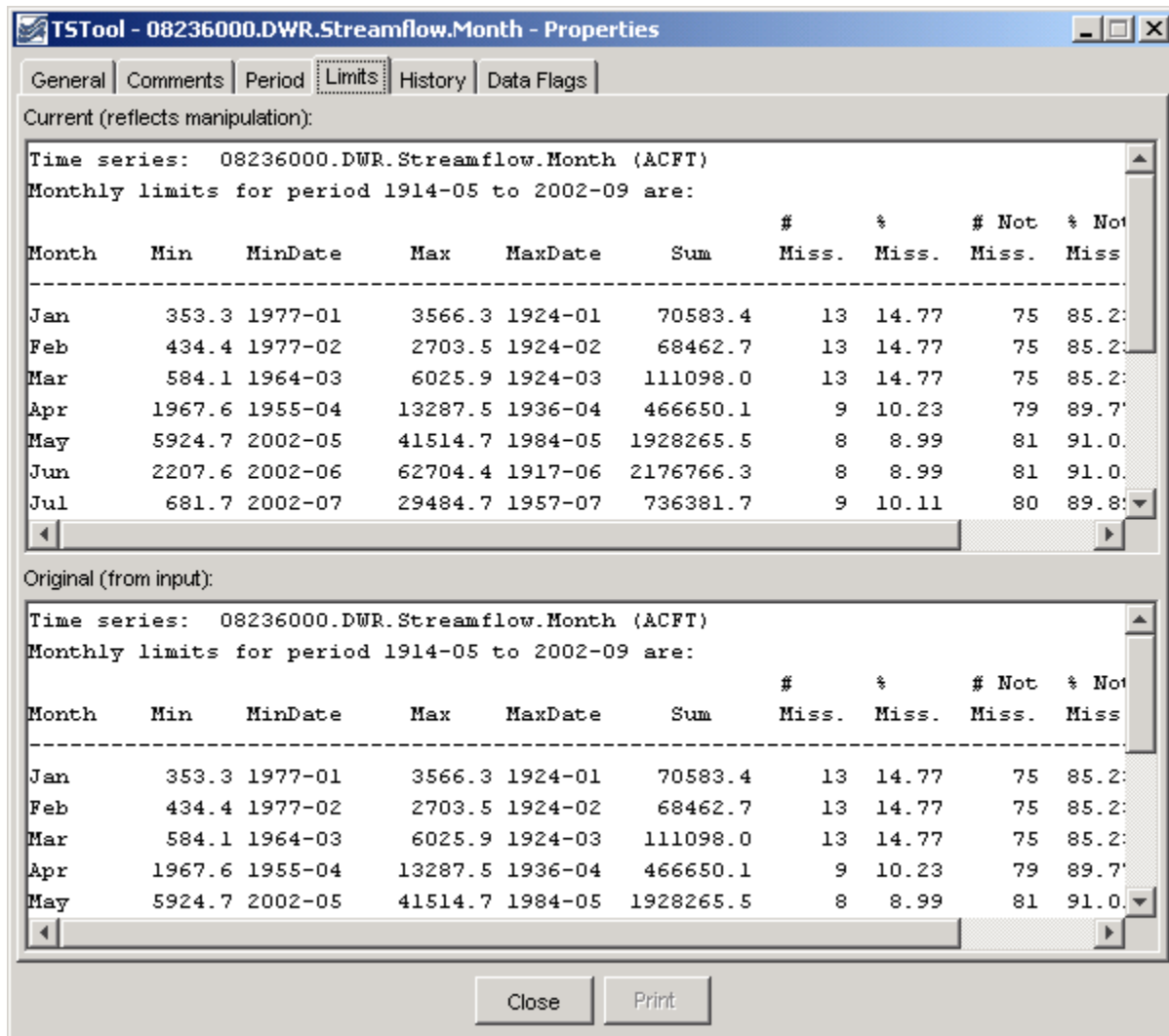
***Original (from
input)***

The original period can be used to indicate the full period available from a database. Setting the original period can sometimes be complicated by how missing data are handled (e.g., a database or file may indicate a certain period but a much shorter period is actually available).

Total Points

Total number of points in a time series. If a regular time series, this can be computed from the period. If an irregular time series, the number of points is determined from a count of all data values. The data points may include missing data – see the data limits for additional information.

Time Series Properties – Limits

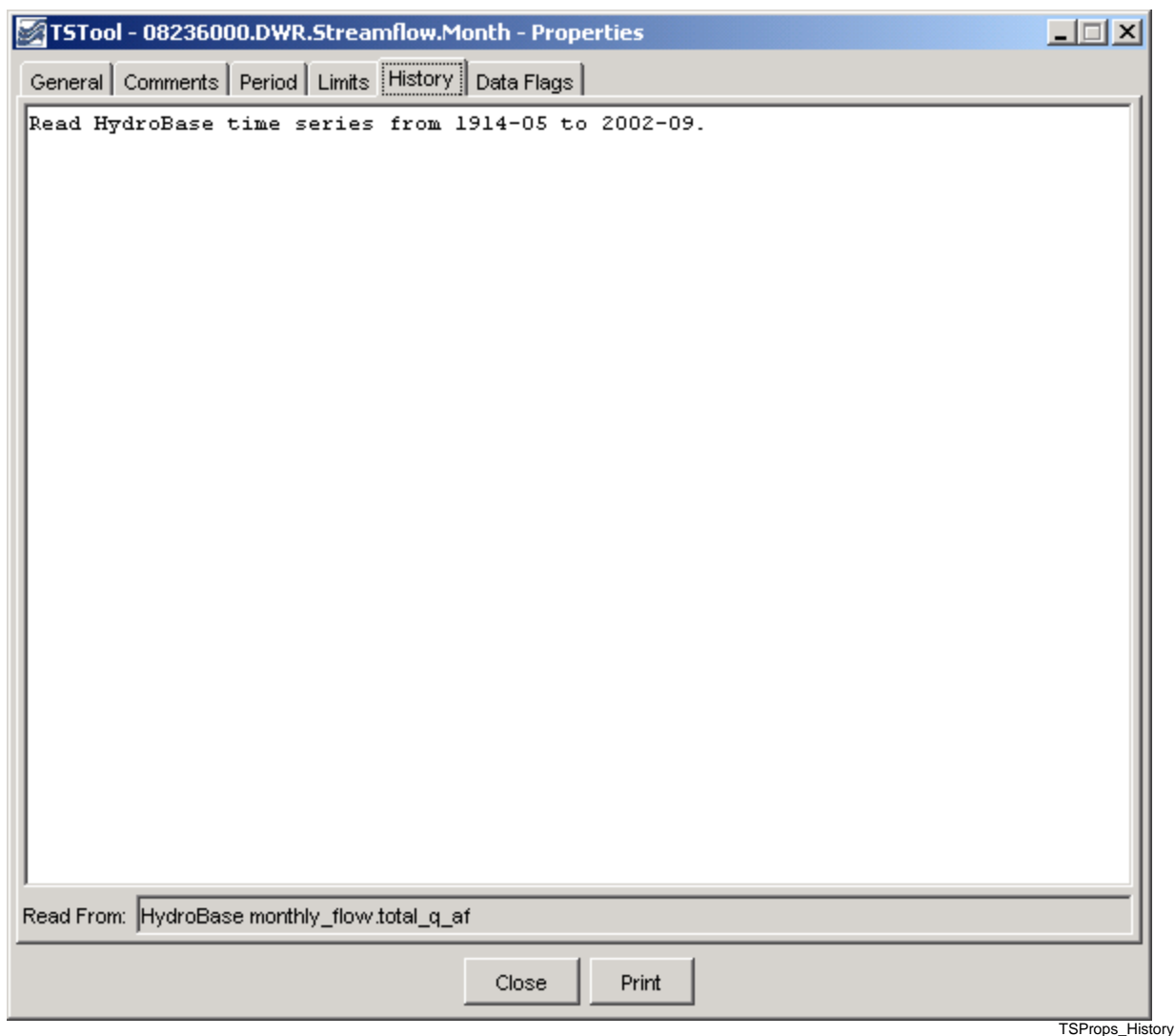


Time Series Properties - Limits

Time series limits are determined for both the current data (top in figure) and the original data (bottom in figure). This is useful because the original data may contain missing data, which are later filled. The data limits are displayed consistent with the data interval. In the example shown, limits are computed for each month. For other time series having other intervals, only overall data limits may be computed.

Theoretically, it is possible that a daily time series could have day limits (e.g., max/min values for each day of the year), month limits (e.g., computed as an average of the daily values by month), and year limits (e.g., computed as an average of all daily values in a year). However, automatically including this level of detail decreases performance and it is difficult to automatically make the right decisions (e.g., about whether to average or total values). Consequently, the limits are currently computed in a basic fashion on the raw data (no interval changes).

Time Series Properties – History

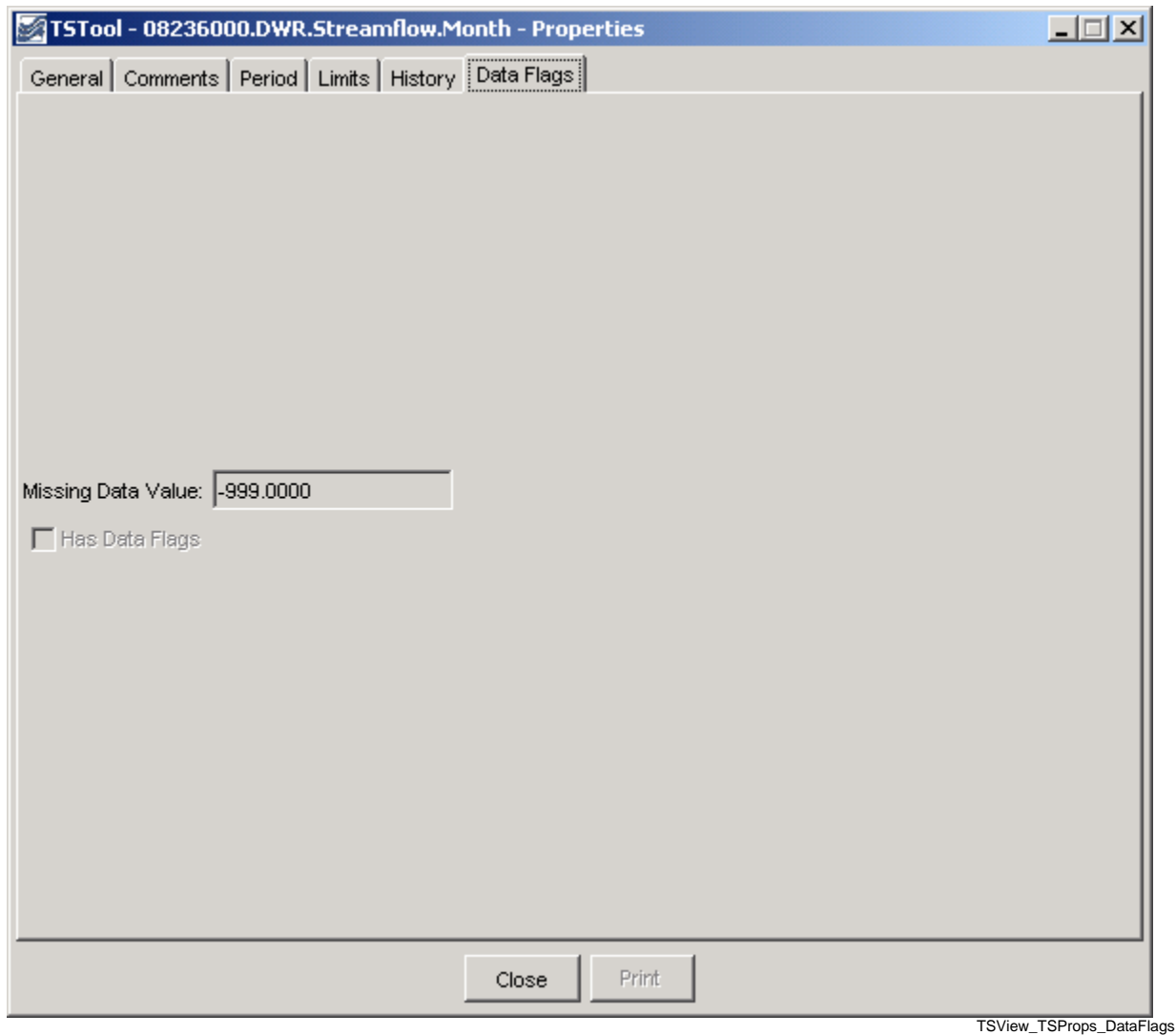


Time Series Properties - History

The time series history (sometimes called the *genesis history*) is a list of comments indicating how the time series has been processed. The completeness of this history is totally dependent on the time series input/output and manipulation software. Although efforts have been made to add appropriate comments as time series are processed, enhancements to the history comments are always being considered.

At the bottom of the history list (see **Read From**) is the input name that was actually used to read the data. This input name may or may not be exactly the same as the input name in the time series identifier. For example, if reading from a HydroBase database, the time series identifier may specify an input type of HydroBase and no input name (because the software knows from the other parts of the time series identifier which database tables to read). However, it is also useful to know the actual table that is read in order to help users and developers understand the data flow. If reading from a file input type, the **Read From** information will show the full path to the file; however, the input name in the time series identifier may only include a relative path.

Time Series Properties – Data Flags



Time Series Properties - Data Flags

Time series data flags contain information that describe the quality of a data point. The missing data value indicates a special number that is used to indicate that a data value is missing at a point. Currently only floating point values are recognized; however the NaN (not a number) value is generally supported for input types that use the convention. All time series are typically assigned a missing data value.

The **Has Data Flags** checkbox indicates whether the time series has data flags. Full support for data flags is being phased in, based on whether an input type supports data flags. The USGS NWIS file format is an example of an input type that supports data flags (e.g., e is used to indicate estimated data).

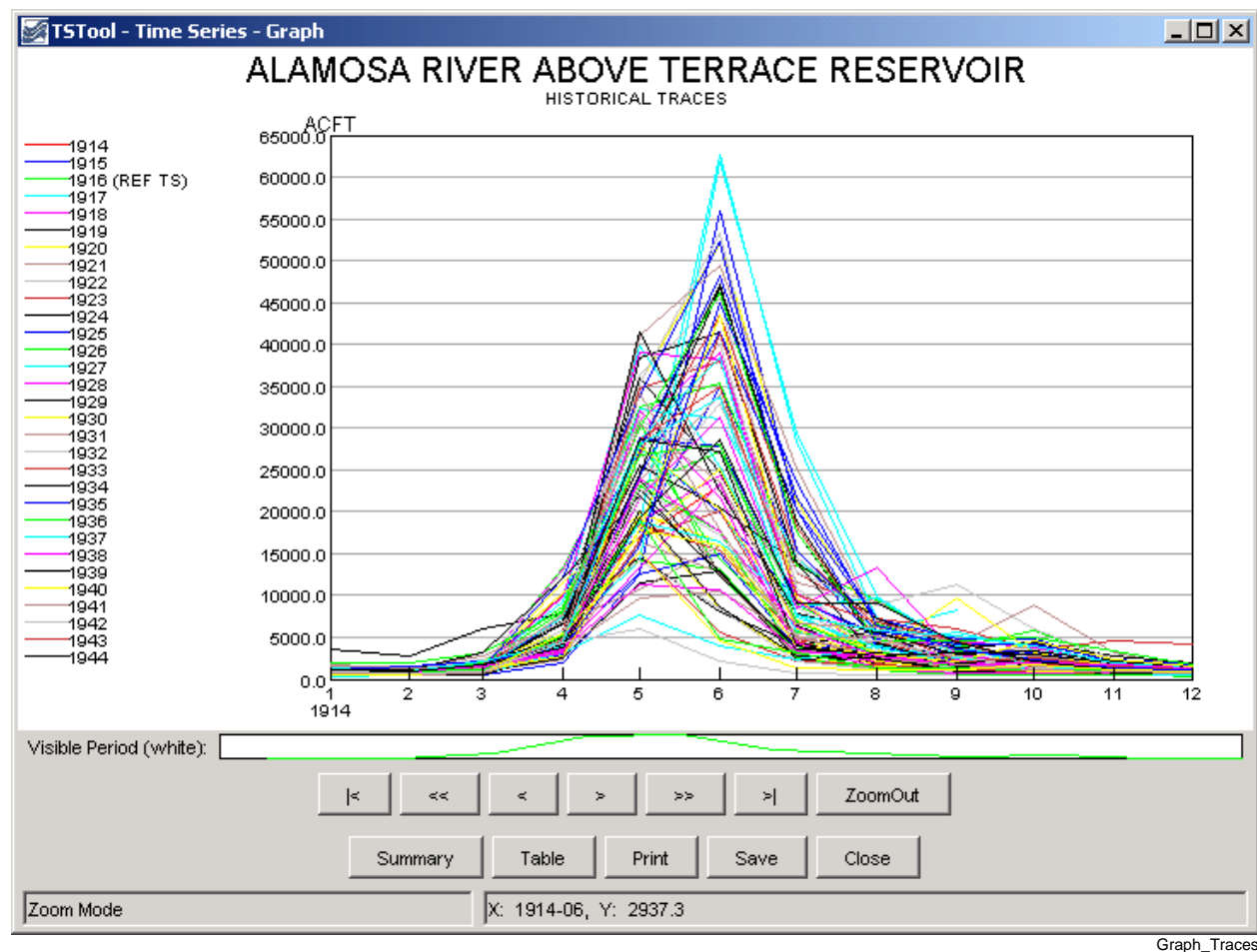
One of the issues with fully supporting data flags is that different input types (and even different data within an input type) treat data flags inconsistently. Therefore, it is easier to add data flags to time series visualization tools (e.g., label points on a graph with the flag) than to integrate data flags in data filling and analysis features. Features related to data flags will continue to be enhanced.

Time Series Traces

In general, the term *time series traces* refers to a group of time series, often shown in overlapping fashion. Common uses of time series traces are:

- Separate a full time series into annual traces and plot them on top of each other, shifted so that they all start at the same date/time,
- Run a model or analytical tool multiple times, with input being a series of input traces, and generating a series of output traces, in order to produce probabilistic simulations.

The power of using traces is that a large amount of data can be used to visualize and study statistical qualities of the data, as shown in the following figure.



Example Graph for Time Series Traces

The TSView package supports time series traces at various levels. Time series properties include a sequence number that can be used to identify a time series as being in a group of traces. However, for data management and viewing, time series identifiers often do not indicate whether a time series is in a group of traces (the sequence number is managed internally). Full support of time series traces is being phased in.

Currently, applications like TSTool include features to create time series traces and TSView tools can be used to view the time series as if they were separate time series. Additional visualization features are being enabled as time allows.

The following sections describe the different time series views that are available in TSView. Although most illustrations use simple time series, most features are also available for use with traces.

Time Series Views

The main components of the TSView package are configured to provide multiple views for time series data. The three main views that are available are:

1. **Graph** - line, bar, or other graph
2. **Summary** - text report suitable for the data type and interval
3. **Table** - spreadsheet-like table with scrolling, suitable for export to other tools

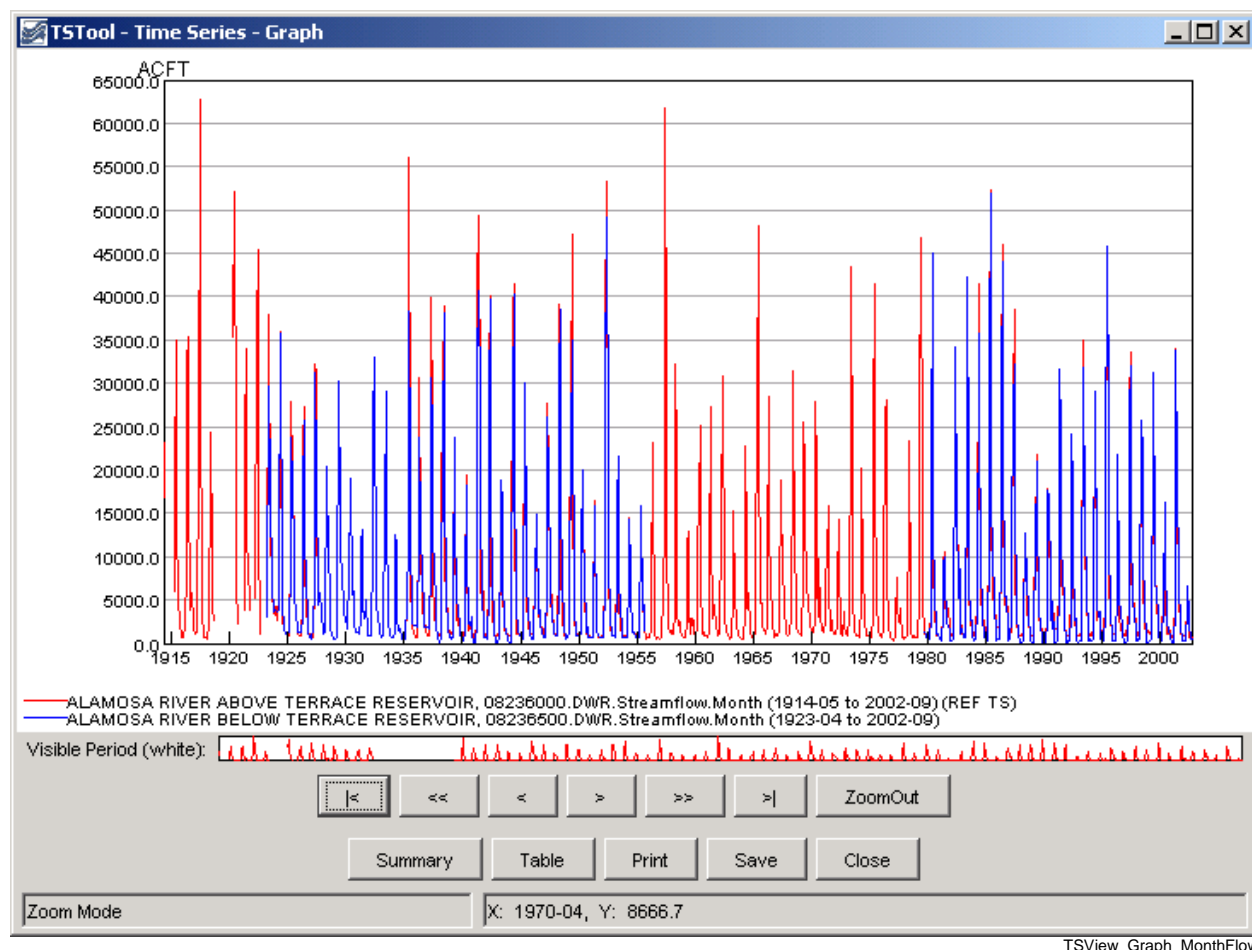
The initial view for a time series list is typically determined from the actions of the software user. For example, a **Graph** button may be displayed on a screen, which when pressed will result in a graph being displayed. The time series that are displayed in the view can typically contain one or more time series (some graph types may have a restriction on the number of graph types). To increase performance and capacity, the TSView package as much as possible uses a single copy of the time series data for visualization. For example, to generate graphs, the data for the time series objects are used directly rather than being copied into a graphing tool's data space. This also allows TSView to more easily display different data intervals on the same view because the data do not need to be forced into a consistent grid data structure.

The following sections describe the three time series views. The graph view type requires more extensive explanation due to the variety of graph properties.

Time Series Graph View

The graph view for time series supports a variety of graph types. The features of the various graph types will be discussed in detail in the following sections, starting with basic graph types, followed by more specific types.

Typically, the graph type is selected in the application (e.g., menus are available in TSTool for selecting the graph type for a list of time series). In many applications, the graph type often defaults to a line graph. The following figure illustrates a line graph for two monthly streamflow time series.



Example Line Graph for Monthly Streamflow

The graph view is divided into the following main areas:

Graph Canvas

The graph canvas is the area where the graph and legend are drawn. This area is used to interact with the graph (e.g., zoom). More than one graph can be drawn in the canvas (see the **Time Series Product Reference** section for additional details). If zooming is supported for the graph, a box can be drawn with the mouse to zoom in to a shorter period. Right-click over a graph of interest to show the popup menu for graph properties and analysis details (e.g., regression results). The canvas area is essentially a preview of a printed graph.

Reference Graph

The reference graph below the main graph canvas shows the current view extent (the white area in reference graph in the figure above). The reference graph is only shown for graph types that support zooming. If shown, it can be used for zooming, similar to the main graph. The time series with the longest period of record is drawn in the reference graph to illustrate variations in the data over time (this time series is noted in the main graph legend with REF TS – this label is not shown in printed output).

Scroll/Zoom Buttons

Under the graph areas is a layer of buttons used for zooming. The **Zoom Out** button will zoom to the full extent of the data.

The other buttons facilitate scrolling through data as described below. For all scrolling operations, the visible graph extent (or *page*) is maintained during the scroll. Scrolling can use the buttons or keys described below. To use the keyboard, first click in the main graph canvas to shift focus to that area.

/<	Home	Scroll the visible window to the start of the period.
<<	Page Down	Scroll the visible window one full page to the left (earlier in time).
<	Left Arrow	Scroll the visible window 1/2 page to the left.
>	Right Arrow	Scroll the visible window 1/2 page to the right (later in time).
>>	Page Up	Scroll the visible window a full page to the right.
>/	End	Scroll the visible window to the end of the period.

Main Buttons

The bottom row of buttons provides features for displaying other views, printing, and exporting:

Summary	Display the summary view for the time series (see the Time Series Summary View section).
Table	Display the table view for the time series (see the Time Series Table View section).
Print	Print the graph. Because the physical extents of the printed page are different from the visible window, the printed graph may not exactly match the viewed version (e.g., more or less axis labels may be used).

Save Save the graph as a Portable Network Graphic (PNG) or JPEG graphic, a DateValue file (a useful time series format), or a *Time Series Product* file (see the **Time Series Product Reference** section) by selecting from the choices. Depending on the main application, saving to a database as a time series product may also be enabled.

Close Close the graph window. If related summary or table windows are still visible, the graph view can be quickly re-displayed by pressing the **Graph** button on the other view windows. If the graph properties have been changed but have not been saved, a warning will be displayed.

Status Message Area The lower-left status message area is used to provide general user instructions and feedback.

Mouse Tracker Area The lower-right status message area is used to indicate the position of the mouse on a graph, in data units. The coordinates are typically shown using an appropriate precision as determined from the time series date/time precision and data units.

Within each graph canvas it is possible to draw more than one graph, each with its own titles, legend, etc. The **Time Series Product Properties** section (below) provides an example and discusses how to edit graph properties. The **Time Series Product Reference** section describes in detail the format of *Time Series Product* files. These files, when saved from the graph view, can be used to recreate a graph interactively or in batch mode, at a later time.

Because TSView is a general tool, a number of rules are in place when viewing time series in graphs (see the **Time Series Product Properties** section for information on changing specific graph properties to override the defaults):

1. Time series plotted on the same graph should generally have the same units or have units that can be readily converted. If the units are not consistent, you will be warned and the units will be displayed in the legend rather on the axis. (A future enhancement may allow multiple axes, each with different units.)
2. Time series can have different data intervals (e.g., daily data can be plotted with monthly data). However, other output options, such as reports, may not allow the same flexibility. It is important to understand the data type characteristics. For example, some data are instantaneous (e.g., real-time streamflow) whereas other data are accumulated (e.g., precipitation) or mean (e.g., mean temperature). Therefore, the representation of the data may need to be selected with care to ensure consistency. For example, some data intervals and types may be better represented as bars and others as lines or points.
3. Data values are plotted at exactly the point that they are recorded. The plot positions are determined using the year as the whole number and months, days, etc. to determine the fractional part of the plot position. The end-user does not typically see these computed positions because labeling uses data units, including dates. The plot positions are determined from the dates associated with data and no

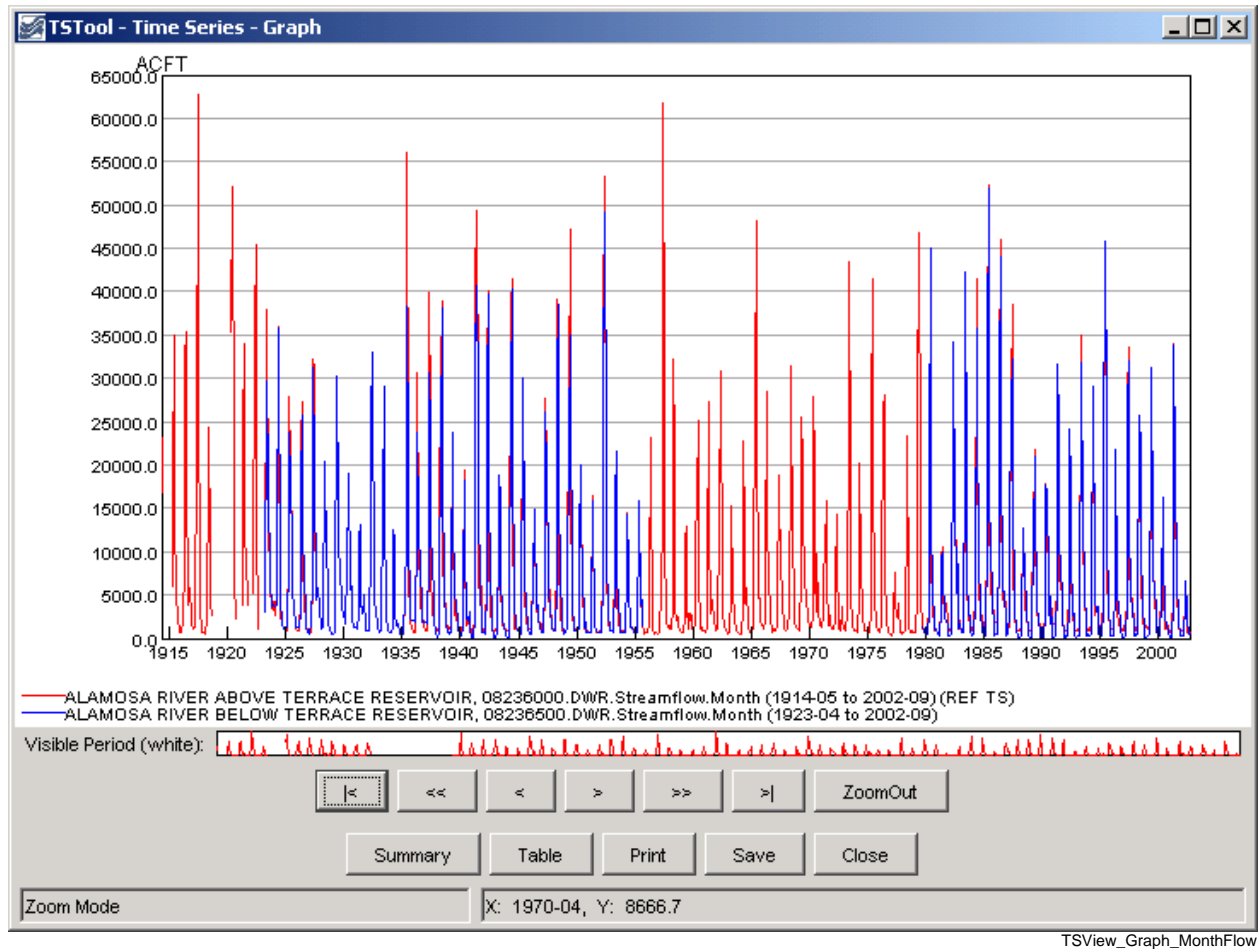
adjustments are made to plot in the middle an interval. For example, monthly data are plotted at the first day of the month (not day 15). Properties to override this convention are being evaluated. Bar graphs allow you to select whether the bars are drawn to the left or right of the date, or centered on the date. This allows flexibility to show a period over which a value was recorded, if appropriate.

4. The mouse coordinates that are displayed by the mouse tracker are computed by interpolating screen pixels back to data coordinates (which involves a conversion of the plot position to date/time notation). Consequently, the values shown may be rounded off (depending on the zoom extent and data precision). The mouse coordinates are displayed based on the precision of the time series data. When moved, the mouse will display the same date until the date changes within the given precision. For example, for monthly data, moving the mouse left to right, the mouse coordinate will display as 1999-01 as soon as the date changes from 1998-12 to 1999-01. The label will remain at 1999-01 until 1999-02 is encountered. Because data values are drawn at points, you should therefore always position the mouse slightly to the right of the point to see the date corresponding to the value. This is very important for bar graphs because the bar may extend over several dates. If specific values need to be determined, use the summary or table views.
5. Labels for axes are determined automatically in most cases based on the font requirements, available display space, and data range. Major and minor tic marks are drawn to help determine the data coordinates. Labels are redrawn as the visible period is changed.
6. Graphs that can be zoomed do not allow the vertical axis to be re-scaled on the fly. This capability is being evaluated.
7. Currently, graph types cannot be mixed for time series on a graph. In other words, a graph cannot contain a bar graph for one time series and a line graph for another time series. This ability may be added in the future. A work-around is to use multiple graphs on a page (see the **Time Series Product Properties** section for an example).
8. The precision used to format graph labels is determined from data unit information provided by the application. This generally produces acceptable graphs. However, in some cases, the range of values being plotted results in inappropriate labels where label values are truncated and/or repeated.
9. Graph types can be changed after the initial display, with limitations. Graphs can be switched between simple types like line and bar graphs; however, simple graphs cannot be changed to more complex types.

The following sections describe various graph types supported by the TSView package. Graph properties are mentioned in some sections. The discussion of how to change graph properties is included in the **Time Series Product Properties** section after the graph type descriptions.

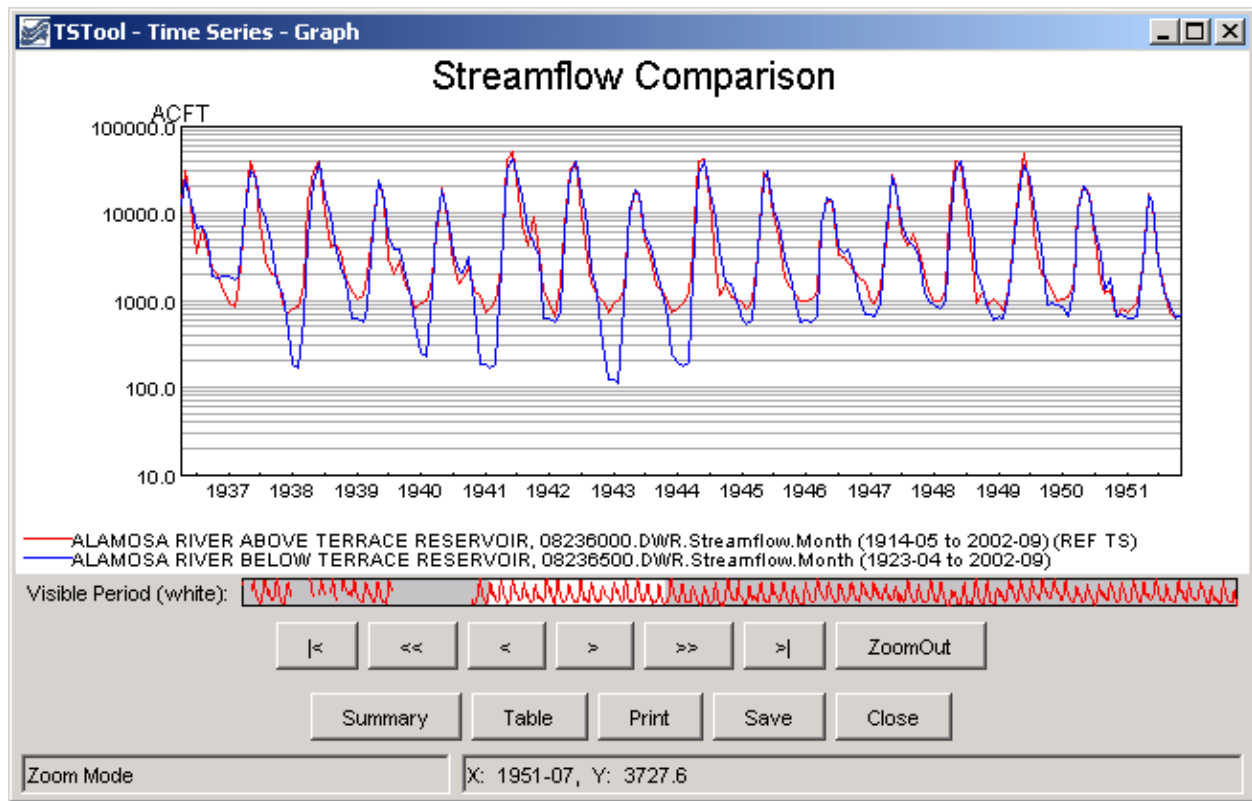
Line Graph

Line graph features have been illustrated in previous discussion. The line graph type is also used to generate graphs with only points by setting the line style to None (for example, software that displays daily data where gaps are expected may default to using symbols and no line).



Line Graph - Log Y Axis

Log-axis line graphs are similar to simple line graphs. The following figure illustrates a typical graph.



TSView_Graph_LogMonthFlow

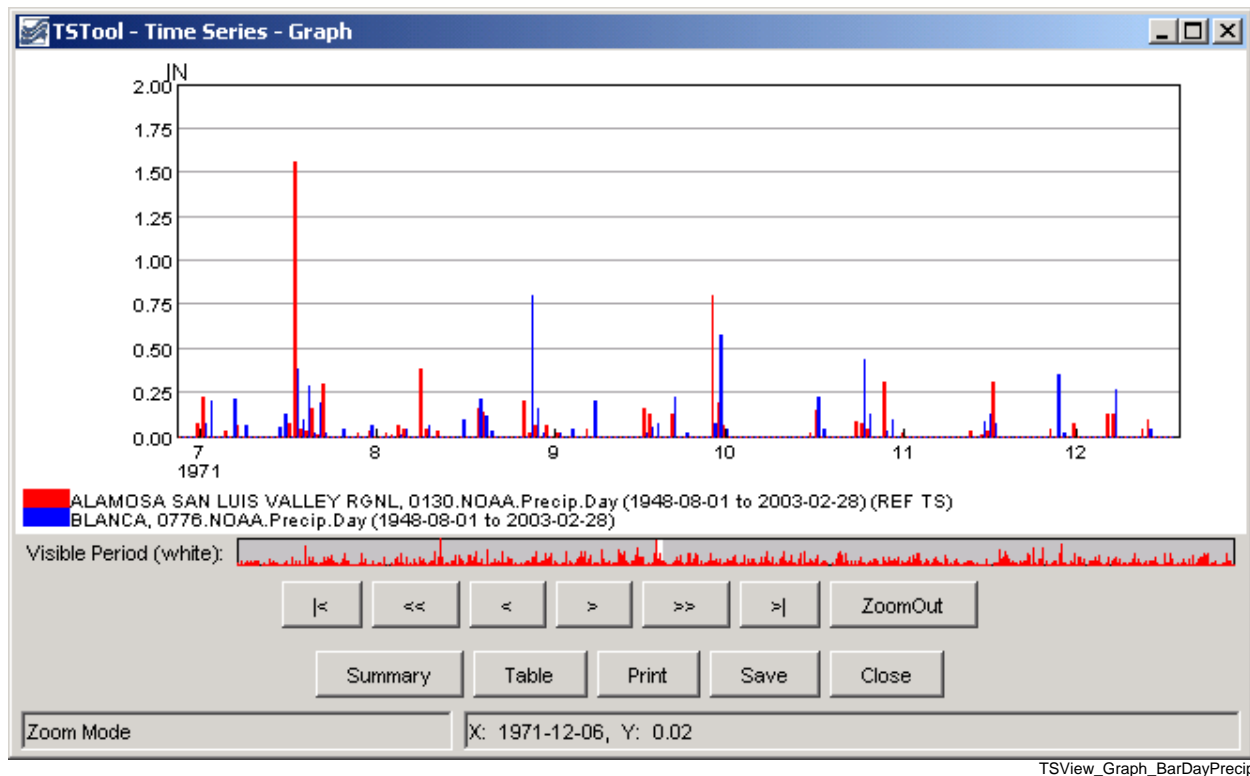
Example Log Y Axis Graph showing Monthly Streamflow

Characteristics of the log plot are:

- If the minimum data value is ≤ 0.0 , then $.001$ is used for the minimum plotting value.

Bar Graph

The bar graph type produces a graph with parallel vertical bars, as shown in the following example:



Example Bar Graph showing Daily Precipitation

The above example illustrates that at the given zoom extent (which is a small part of the full period - see the white area in the reference graph), labels are drawn for months. Zooming in more would display the day in the labels. The mouse tracker in all cases shows days since that is the precision of the data. Characteristics of the bar graph are as follows:

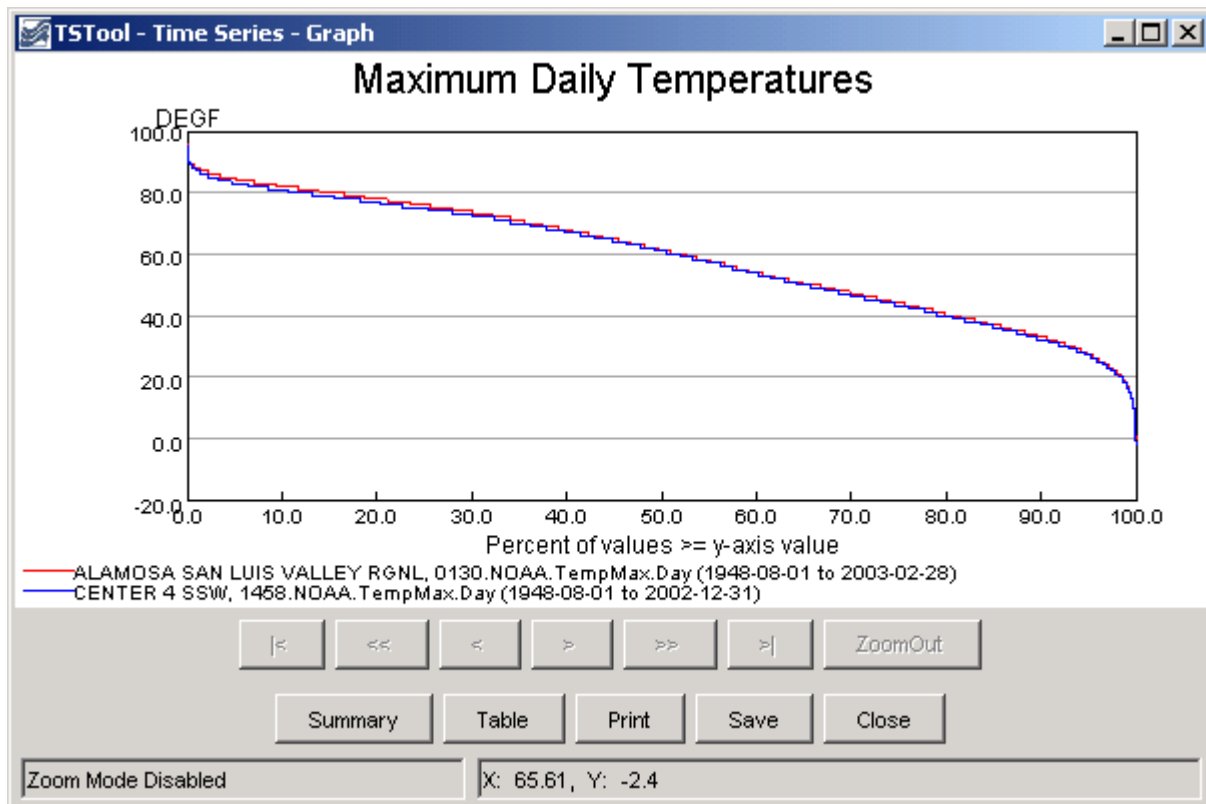
- Bars can be plotted centered on, to the left of, or to the right of the dates. If multiple time series are plotted, the overall total width of the bars will correspond to one data interval. If drawn to the left of the date, the bars for all graphed time series are drawn to the left of the date. If drawn to the right of the date, the bars for all graphed time series are drawn to the right of the date. If centered on the date, half the bars are drawn to the left of the date, and half to the right.
- Bar widths are determined based on the number of time series being plotted. Monthly time series use a slightly narrower bar (larger gap between bars) because the number of days in a month varies. To make bars stand out better, a white line may be drawn to separate adjacent bars. If bars are very narrow the line is not drawn. Bars will always be drawn at least one pixel wide, even if this obscures neighboring bars (zoom in to see more detail). Round-off in drawing bars may result in some bars being slightly wider or narrower than other bars.
- Bars always end at the zero value on the Y axis. In other words, bars extend up or down from zero.
- The mouse cursor display dates relative to the axis and does not determine the data value relative to edges of the bars. For example, if bars are plotted centered on dates, 1/2 of the bar will actually be in the previous date, according to the mouse tracker.

Double Mass Curve

Double mass curves are currently disabled. An alternative is to use the TSTool application and generate cumulative time series, which can be viewed in a line graph.

Duration Graph

A duration graph indicates the range of values in a time series and how often they occur, as shown in the following example:



TSView_Graph_DurationMaxDayTemp

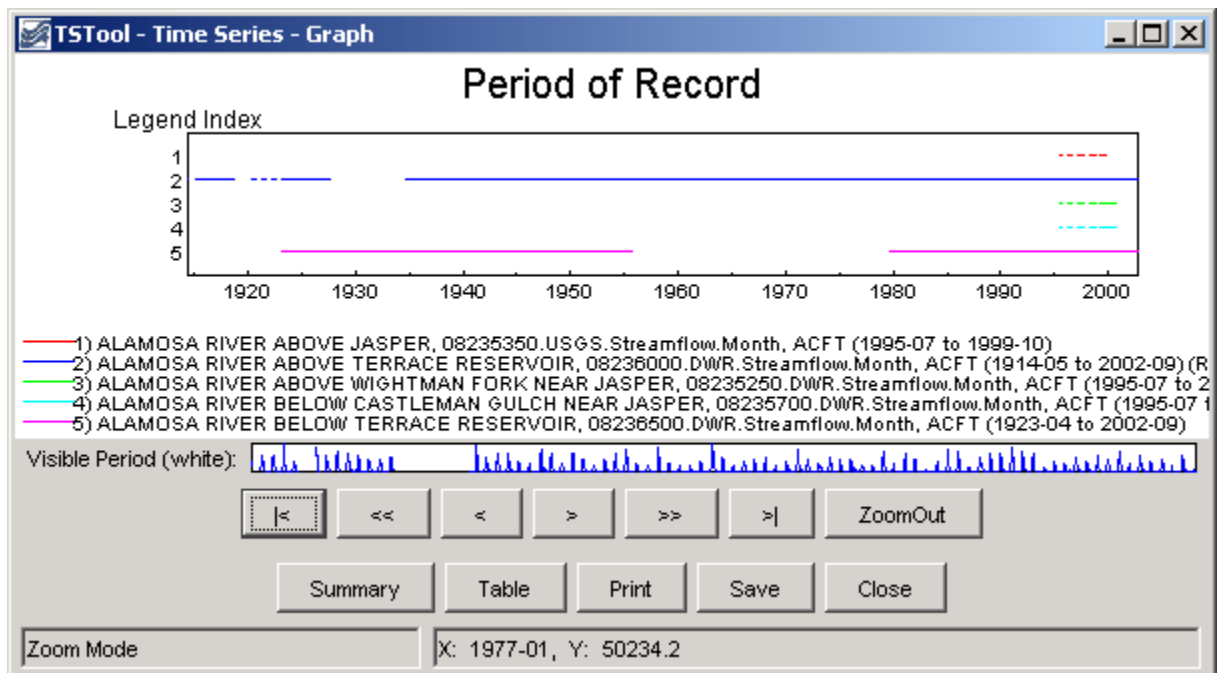
Example Duration Graph showing Maximum Monthly Temperatures

The algorithm for calculating and graphing a duration curve was taken from the book **Handbook of Applied Hydrology** (edited by Ven Te Chow): “When the values of a hydrologic event are arranged in the order of their descending magnitude, the percent of time for each magnitude to be equaled or exceeded can be computed. A plotting of the magnitudes as ordinates against the corresponding percents of time as abscissas results in a so-called duration curve. If the magnitude to be plotted is the discharge of a stream, the duration curve is known as a flow-duration curve.” Features of duration graphs are as follows:

- The zoom feature is disabled for this graph type.
- Although duration curves have traditionally been applied to streamflow or reservoir data, duration graphs can be created for any time series data.
- Noticeable breaks in the curve are caused by a limited number of data points and/or values that are measured as rounded values.

Period of Record Graph

The period of record graph is used to display the availability of data over a period, as shown in the following figure:



TSView_Graph_PORMonthFlow

Example Period of Record Graph showing Monthly Streamflow

Characteristics of the period of record graph type are:

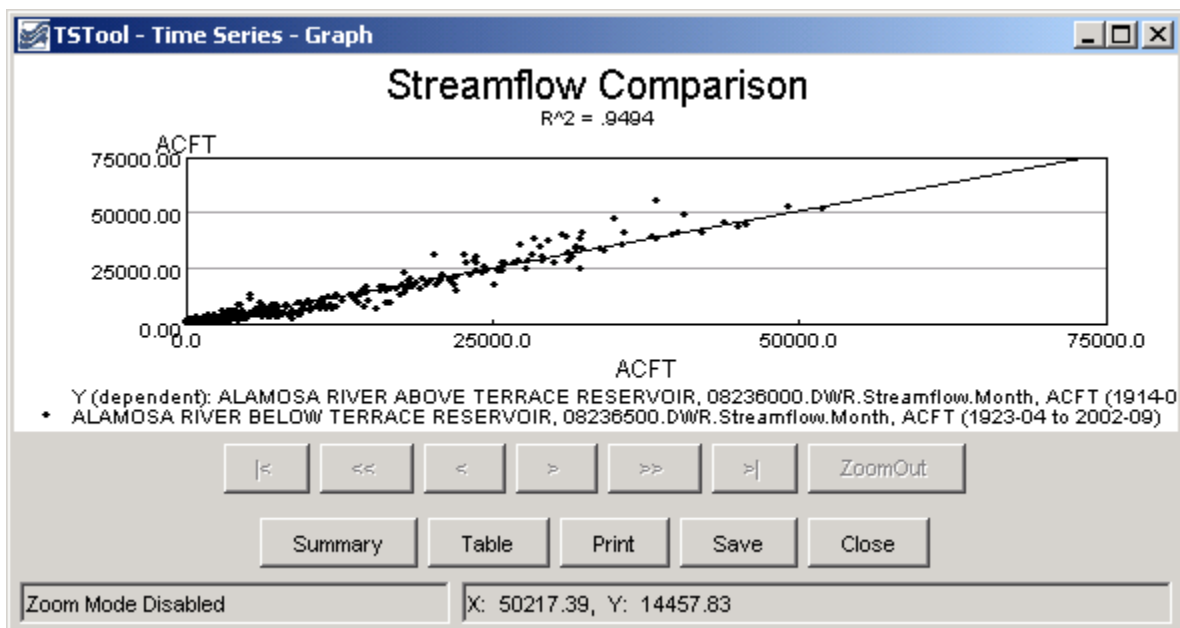
- Horizontal lines are drawn for each time series, with breaks in the line indicating missing data.
- Zooming is fully enabled, however, it may be difficult to see small breaks in the lines – it may be necessary to display symbols at the data points. The data limits properties of each time series can also be used to check for missing data. The TSTool application provides reporting features to summarize data coverage.
- Because data values are not plotted, the y-axis is labeled with a legend index number. This also allows the graph window to be compressed vertically, if desired.

XY-Scatter Graph

The XY-Scatter graph type can be used to compare data having the same data interval (units can be different). This graph type is often used for the following comparisons:

1. The dependent time series (Y) requires filling and multiple independent time series (X) are analyzed to find the best time series to use as the independent time series. One or more independent time series can be plotted on the same graph.
2. The dependent time series (Y) contains observed data and one or more independent simulated time series (X) are analyzed to determine which simulation is closed to actual observations.
3. The independent (X) and dependent (Y) time series are compared to determine whether any time of relationship exists between data points. In this case, a single dependent time series may be compared with multiple independent time series on the same graph.

Currently the XY-Scatter graph can have only a single dependent time series but can have one or more independent time series. The following figure shows a typical graph.



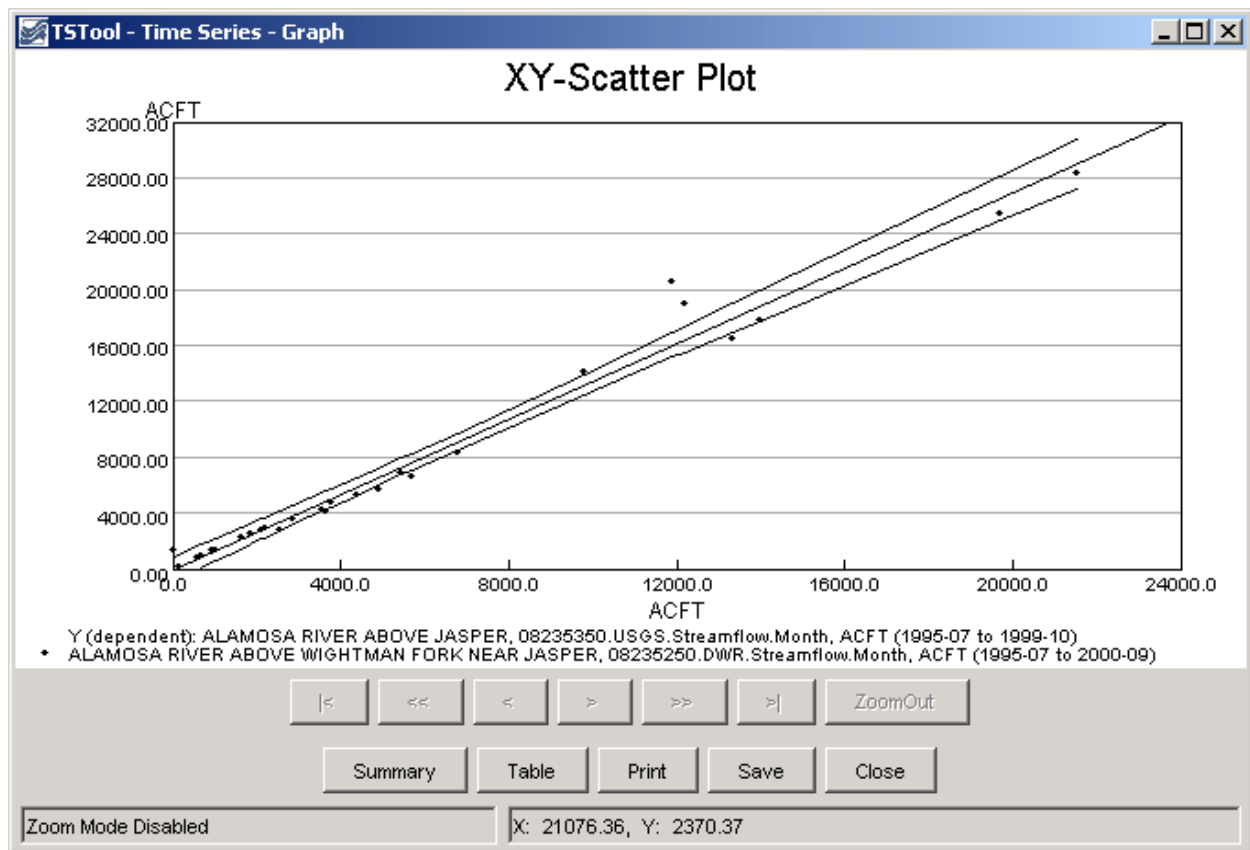
TSView_Graph_XYMonthFlow

Example XY Scatter Graph showing Monthly Streamflow

Characteristics of the XY Scatter graph are:

- Labels and legend are automatically generated. See the **Time Series Product Properties** section below for information about changing the appearance of the graph.
- Simple linear regression is initially performed to determine a line of best fit. See the **Analysis** tab in the **Time Series Product Properties** section below for information about curve fit methods.
- A 45-degree line is currently **not displayed** because time series of different types and units may be compared. Graph properties do allow the line of best fit to be forced to zero. The limits on the axes are not automatically set to equal values for the same reason; however, a property to force the values to be the same will be added.
- Zooming is disabled.
- Two or more time series must be specified and must have the same interval.

- Confidence intervals can be turned on, as shown in the following figure:



TSView_Graph_XYConfidence

The confidence intervals provide a useful way for assessing the quality of a point estimate. When a regression line is of interest, the confidence interval on the line as a whole permits one to make confidence statements about a number of values of the predictor variables simultaneously. Confidence limits for the line are a function of the level of confidence (e.g., gamma = 95% or 99%), and the F-test statistic (2, n-2 degrees of freedom, and level of significance = 1-gamma). The equations used to plot the confidence intervals are shown below (note that because the curves depend on the data points, the shape and smoothness of the curves will depend on the number of points; the points are sorted to generate a continuous line).

$$\hat{y}_{CI_i} = [\bar{y} + B(x_i - \bar{x})] \pm \sqrt{2F} \left[\frac{1}{n-2} \sum_{j=1}^n (\hat{y}_j - y_j)^2 \right]^{1/2} \left[\frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_{j=1}^n (x_j - \bar{x})^2} \right]^{1/2}$$

where:

\hat{y}_{CI_i} = confidence interval y value at x_i

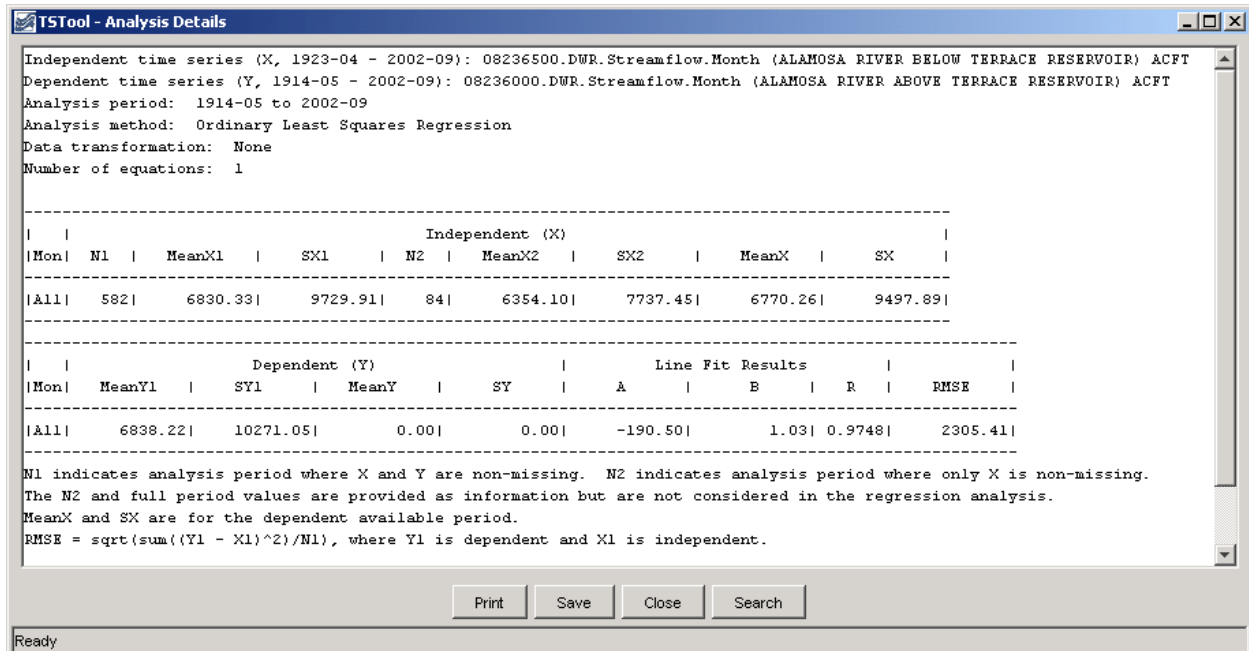
\bar{y} = mean of y

B = slope of regression line equation $y = A + Bx_i$

x_i = x value where \hat{y}_{CI_i} is being computed

- \bar{x} = mean of x
 F = F distribution at $(2, n-2)$ degrees of freedom and gamma significance
 n = number of points with x and y values
 \hat{y}_i = y predicted by the equation $y = A + Bx_i$
 y_i = y value of data point corresponding to x_i

- The best-fit line can be turned off.
- Right-clicking on the graph displays the **Analysis Details** menu, that, if selected, displays curve fit information about the time series, as illustrated in the following figure:



Example Analysis Details

TSView_Graph_XYRegressionDetails

The RMS error (or *RMSE*) is calculated in the following way:

$$SSE = \sum (X_i - Y_i)^2 = \text{Sum of Square Errors}$$

$$MSE = SSE/N = \text{Mean of Sum of Square Errors}$$

$$RMSE = \sqrt{MSE} = \text{Square Root of the MSE}$$

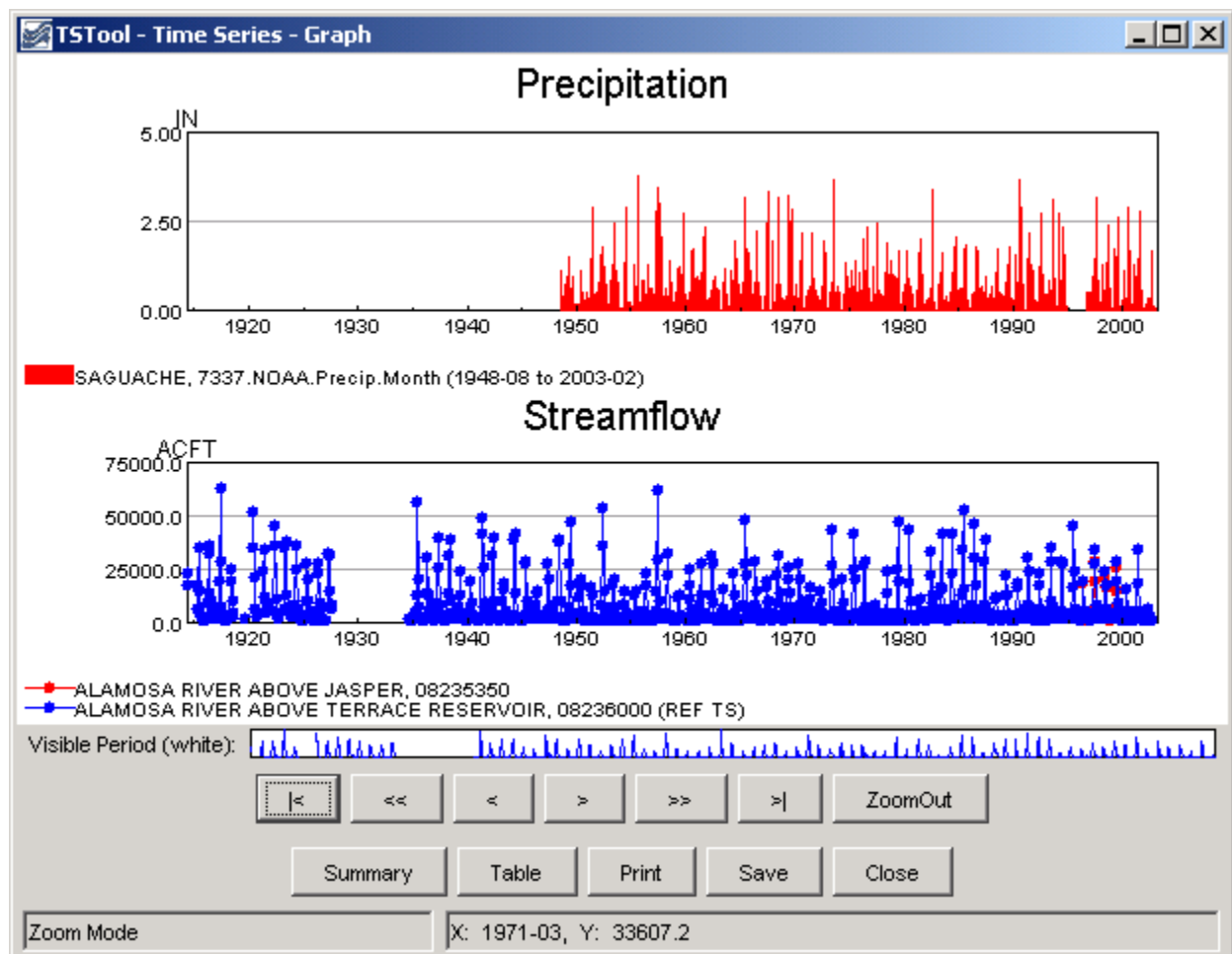
The *RMSE* can have different meanings, depending on how the data are being analyzed:

- If a measured (X) and a simulated (Y) time series are being compared to determine, for example, to determine how well a model is simulating actual observations, then the *RMSE* indicates the error of a simulation when compared to the actual (comparing the values).
- If two time series are evaluated to determine if the relationship between the time series can be used to estimate missing values in one of the time series, then the difference between estimated values (Y_{est}) and the line of best fit (e.g., $A + BX$) is used to compute the *RMSE*. For a perfect fit, the *RMSE* would be zero. Values of *RMSE* can be used to evaluate the estimator for data filling.

To provide as much information as possible for multiple uses, the **XY-Scatter Graph Analysis Details** provides both *RMSE* values. The default is to display a line of best fit, which is usually desirable information. The graph properties allow the analysis to be done for data filling, if desired.

Time Series Product Properties

A time series product is one or more time series graphs, tables, or reports on a “page”, although currently TSView focuses on graph products. Time series product properties can be displayed by right clicking on a graph of interest and selecting the **Properties** menu item from the popup menu. Interactively changing properties allows graphs to be configured as desired. The following figure illustrates a time series product that has two graphs (see the **Time Series Product Reference** section for information about how to define time series product files, which can be used to save a product).



Example Graph Product showing Precipitation and Streamflow

In many cases, a graph product will consist of only a single graph (which may show one or more time series). However, it is also useful to display multi-graph products, especially when related data types are used. The TSView interface includes features to construct multi-graph products interactively, and the product files described in the **Time Series Product Reference** section can be created and processed. The TSView application, for example, can interactively create or read a product file and display a graph similar to the one shown above. Important considerations for multi-graph products are:

- The product page has its own set of properties (e.g., titles and size).

- Each graph area has its own properties (e.g., titles, labels, graph type, legend). These properties comprise most of the properties for a product.
- Each time series has its own properties (e.g., symbol, color).
- If zooming is enabled, then zooming in one graph causes the same zoom to occur in related graphs. Each graph (and the reference graph) is assigned a *zoom group* number. This is used to indicate which graphs should zoom together. Currently, all graphs are in the same zoom group.

Right clicking on a graph and pressing the **Properties** item in the popup menu will display the properties for the graph. The following figures illustrate the properties tabbed panel:

TSTool - Time Series Product Properties

Product Properties:

General Titles Layout

Product ID: Product1

Product Name:

☒ Product Enabled

Graph Properties:

1 - Precipitation

General Graph Type Titles X Axis Y Axis Label Legend Zoom Analysis Annotations

☒ Graph Enabled

Y Percent Size:

Time Series Properties:

1 - 7337.NOAA.Precip.Month~HydroBase

General Graph Type Axes Symbol Label Legend Analysis

☒ Time Series Enabled

Apply Close

TSView_TSProduct_Props_All

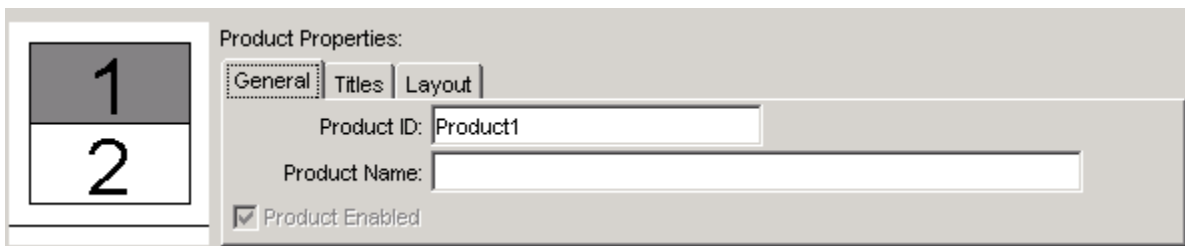
Tabbed Panel to Edit Time Series Product Properties

The time series product properties display as three layers of tabbed panels. Characteristics of the properties window are:

- The window is divided into a layout area (top-left) and tabs for different groups of properties. The layout window shows the overall layout of graphs on a page and allows manipulation of the time series product by dropping time series onto the layout.
- The top layer of tabs (**Product Properties**) is associated with product properties (the page).
- The middle layer of tabs (**Graph Properties**) is associated with subproduct properties (graphs on the page). The graph of interest is selected using the drop-down choice that shows the graph number and graph main title. When initially displayed, the selected graph is the one that was clicked on to display the **Properties** menu.
- The bottom layer of tabs (**Time Series Properties**) is associated with data (time series) properties. A time series within a graph is selected using the drop-down choice that shows the time series number within the graph, and the time series identifier. When initially displayed, the first time series for the selected graph is selected.
- The **Apply** button will apply the current properties and update the graph(s). **Warning - when changing between graphs and time series (where multiple graphs and/or time series exist for a product), properties that are changed are applied automatically. This behavior is being evaluated.**
- The **Close** button will apply the current properties, update the graph(s), and close the properties window.
- Only properties read from an original time series product file or that are set by the user will be saved if a time series product is saved. Internal defaults are not saved. This minimizes the size and complexity of product definition files.

The remaining discussion in this section illustrates each of the tabbed panels. The text-based properties that are displayed in the panels are described in the **Time Series Product Reference** section.

Product Properties - General



Product Properties:

General Titles Layout

Product ID: Product1

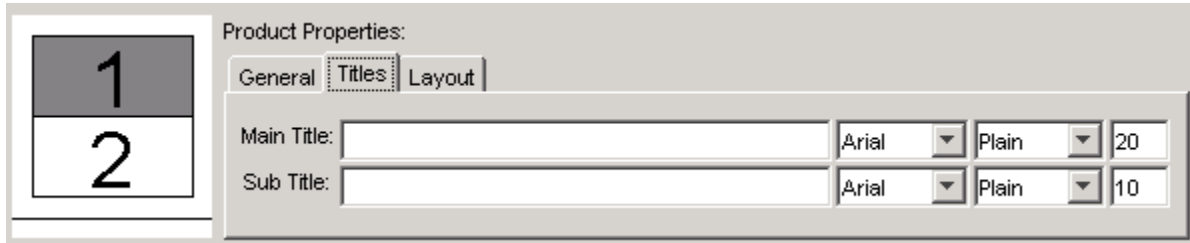
Product Name:

☒ Product Enabled

TSView_TSProduct_Props_Product_General

Example Product General Properties

The above figure illustrates the product **General** properties. The **Product Enabled** checkbox indicates whether the product is enabled (currently view-only). The **Product ID** is used when saving the product definition to a database. The **Product Name** is also used to when displaying lists of products.

Product Properties - Titles


Product Properties:

General **Titles** Layout

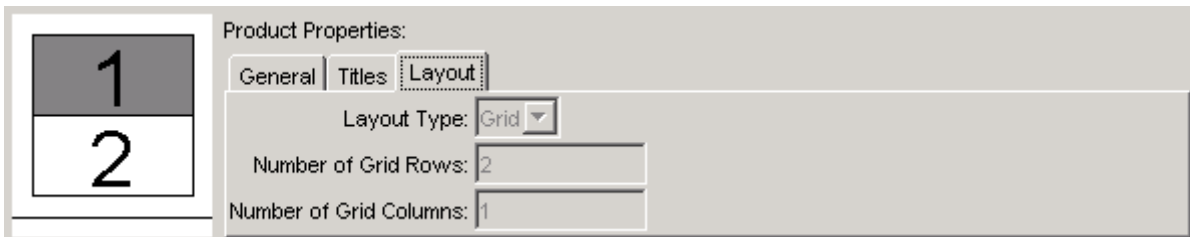
Main Title: Arial Plain 20

Sub Title: Arial Plain 10

TSView_TSProduct_Props_Product_Titles

Example Product Title Properties

Product **Titles** properties include title and subtitle. If blank, no title will be shown. Because graphs (subproducts) also have a title and subtitle, the product titles are often only used when multiple graphs are included on a page.

Product Properties - Layout


Product Properties:

General **Titles** Layout

Layout Type: Grid

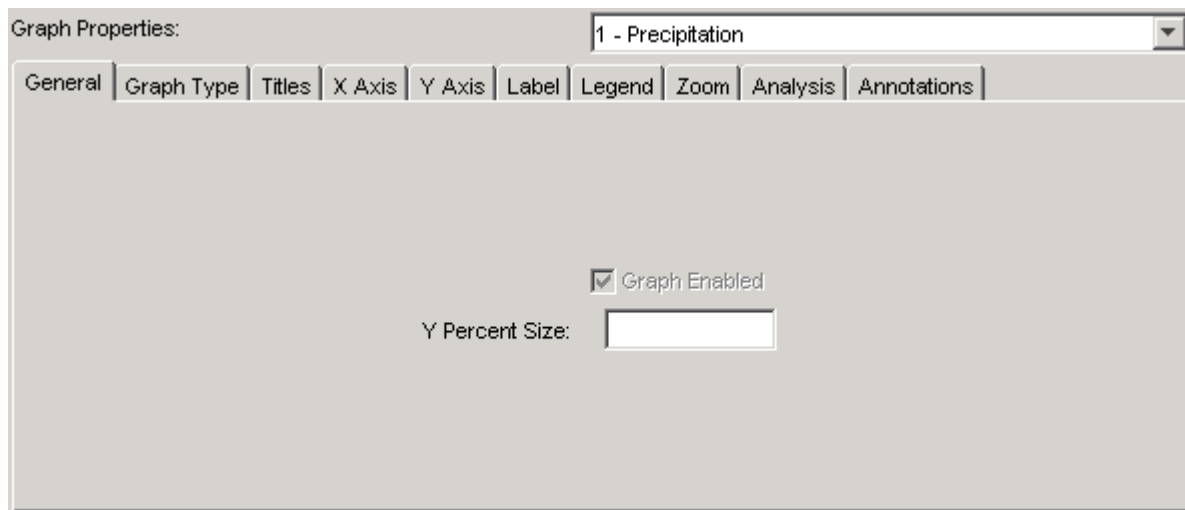
Number of Grid Rows: 2

Number of Grid Columns: 1

TSView_TSProduct_Props_Product_Layout

Example Product Layout Properties

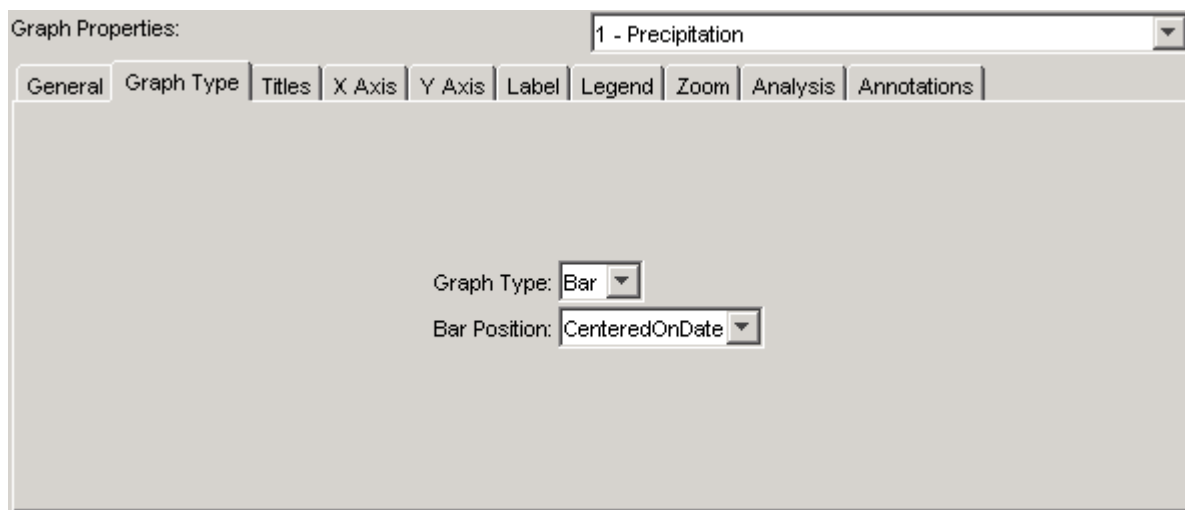
Product **Layout** properties describe how graphs are laid out on the page. Currently, graphs can only be organized in a vertical stack, although the design will support multiple columns. The layout properties are updated automatically as graphs are added to or deleted from the layout window at the left. The relative size of each graph on the page is controlled by using the `LayoutYPercent` general property for each graph on the page (see below).

Graph Properties - General

TSView_TSProduct_Props_Graph_General

Example Graph General Properties

The above figure illustrates graph **General** properties. The **Graph Enabled** checkbox indicates whether the graph is enabled (currently view-only). The vertical size of the graph on the page (percent) can also be specified (the default is to size all the graphs on the page equally).

Graph Properties - Graph Type

TSView_TSProduct_Props_Graph_GraphType

Example Graph Graph Type Properties

Graph Type properties control the overall display of the data. The graph type can be changed after the initial display only when switching between simple graph types (e.g., line and bar graphs). Some graph types may have specific properties (e.g., bar width for bar graphs). If necessary, to change the graph type, you can usually select the type from a main application, and generate a new graph.

Graph Properties - Titles

Graph Properties: 1 - Precipitation

General Graph Type Titles X Axis Y Axis Label Legend Zoom Analysis Annotations

Main Title: Precipitation Arial Plain 20

Sub Title: Arial Plain 10

TSView_TSProduct_Props_Graph_Titles

Example Graph Title Properties

Graph **Titles** properties include title and subtitle. If blank, no title will be shown. Font properties can also be specified. After applying the a change to the main title, the title will be added in the list of graphs.

Graph Properties - X Axis

Graph Properties: 1 - Precipitation

General Graph Type Titles X Axis Y Axis Label Legend Zoom Analysis Annotations

Bottom Title: Arial Plain 12

Label Font: Arial Plain 10

Major Grid Color: None None Custom

TSView_TSProduct_Props_Graph_XAxis

Example Graph X Axis Properties

Graph **X Axis** properties include title, label, and grid properties. The **Major Grid Color** can be specified by selecting from the available choices, which then fill in the text field with the given color selection.

Graph Properties - Y Axis

Graph Properties: 1 - Precipitation

General Graph Type Titles X Axis Y Axis Label Legend Zoom Analysis Annotations

Left Title: IN Arial Plain 12

Label: Precision: 2 Font: Arial Plain 10

Axis Type: Linear Min Value: Auto

☐ Ignore Units Units: IN Max Value: Auto

Major Grid Color: lightgray LightGray Custom

TSView_TSProduct_Props_Graph_YAxis

Example Graph Y Axis Properties

Graph **Y Axis** properties include the following:

- **Left Title** - this may be set to the data units but can be specified (the Y axis title is currently always placed at the top of the Y axis).
- **Label** - the font for labels and precision of numerical labels can be specified.
- **Axis Type** - currently this is view-only.
- **Min Value, Max Value** - currently this is view-only but can be set in time series product definition files (see the **Time Series Product Reference** section).
- **Units, Ignore Units** - currently these are view-only. If time series with incompatible units are graphed, **Ignore Units** will be checked and the units may be shown in the legend.

Graph Properties - Label

Graph Properties: 1 - Precipitation

General Graph Type Titles X Axis Y Axis **Label** Legend Zoom Analysis Annotations

The Label Format, if specified, will override the Time Series label properties.

Position: Right

Format (see choices): %a - Weekday, abbr Arial Plain 10

TSView_TSProduct_Props_Graph_Label

Example Label Properties

Data points are not labeled by default because there are usually too many data labels to be legible. However, for plots with limited data, or after zooming in, labels can be useful to identify points without referring to tabular data. The label format can be defined using the choices next to the text field or by entering literal text. For an XY Scatter plot, repeat the %v format (e.g., %v, %v) to show the independent (X) and dependent (Y) data values. See the **DataLabel** properties in the **Time Series Product Reference** section for label options.

Graph Properties - Legend

Graph Properties: 1 - Precipitation

General Graph Type Titles X Axis Y Axis Label **Legend** Zoom Analysis Annotations

If the Format is Auto, the Time Series Legend Format or defaults will be used.

Position: Bottom

Format (see choices): Auto Auto Arial Plain 10

TSView_TSProduct_Props_Graph_Legend

Example Graph Legend Properties

Graph **Legend** properties include format and font properties. If the **Legend Format** is **Auto**, a default legend format will be constructed from the time series description, identifier, and period of record. See the **LegendFormat** property in the **Time Series Product Reference** section for legend formatting options.

Graph Properties - Zoom

Graph Properties: 1 - Precipitation

General | Graph Type | Titles | X Axis | Y Axis | Label | Legend | **Zoom** | Analysis | Annotations

☒ Zoom Enabled

Zoom Group: 1

TSView_TSProduct_Props_Graph_Zoom

Example Graph Zoom Properties

Graph **Zoom** properties are currently view-only. Zoom will be enabled for graph types that support it (e.g., duration graphs do not). The zoom group indicates how graphs should respond when other related graphs on a page are zoomed and currently defaults to 1 for all graphs.

Graph Properties - Analysis

Graph Properties: 1 - XY-Scatter Plot

General | Graph Type | Titles | X Axis | Y Axis | Label | Legend | Zoom | **Analysis** | Annotations

Select the parameters for the XY-Scatter Graph curve fit analysis (applies to all time series).

Curve Fit Method: OLSRegression Transformation: None

Number of Equations: OneEquation Month(s) of Interest:

Dependent Analysis Period: to

Independent Analysis Period: to

☐ Analyze (RSME) for Filling Intercept:

Fill Period: to

TSView_TSProduct_Props_Graph_Analysis

Example Graph Analysis Properties

Graph **Analysis** properties are available if the graph requires some type of analysis to produce the result (e.g., curve fitting). See also the analysis tab for individual time series. For help with input, place the mouse cursor over a field and a tool tip will be shown.

Graph Properties - Annotations

Graph Properties: 1 - Precipitation

General | Graph Type | Titles | X Axis | Y Axis | Label | Legend | Zoom | Analysis | Annotations

Annotation Properties: Annotation 1 [Add Annotation] [Delete Annotation] [Up] [Down]

Annotation ID: Annotation 1

Shape Type: Text

Order: OnTopOfData

X Axis System: Percent

Y Axis System: Data

Text

Text: Flood Alarm

X: 50 Y: 5.5

Text Position: Right

Font Name: Arial

Font Style: Plain

Font Size: 10

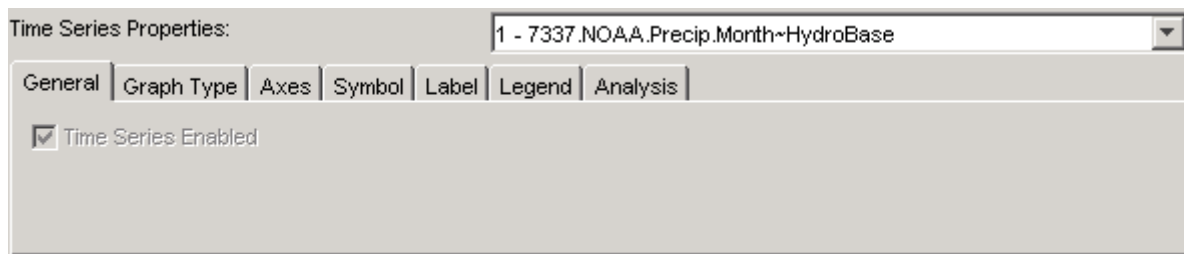
Color: Red

TSView_TSProduct_Props_Graph_Analysis

Example Graph Analysis Properties

Graph **Annotations** properties are used to add annotation objects to a graph. Annotations are text, line, or other simple shapes and are stored as simple text properties in time series products (see the **Time Series Product Reference** section below for more information). Annotations are placed on a graph using data units or a percent of the graph dimension. This allows annotations to move if a graph uses real-time data.

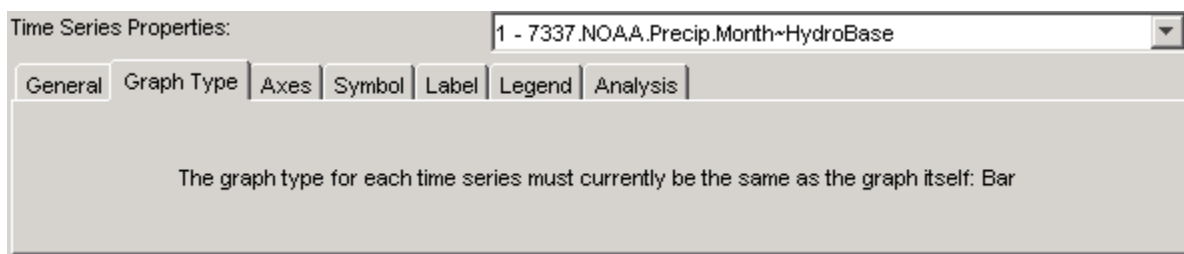
To add an annotation, press the **Add Annotation** button. Then select the **Shape Type** and specify annotation properties, as appropriate. The example shown in the above figure places the string “Flood Alarm” at the horizontal (X) center of the graph at a Y-coordinate of 5.5. A horizontal annotation line could also be drawn using 0 to 100 percent on the X-axis at the same Y-coordinate.

Time Series Properties - General

TSView_TSProduct_Props_TS_General

Example Time Series General Properties

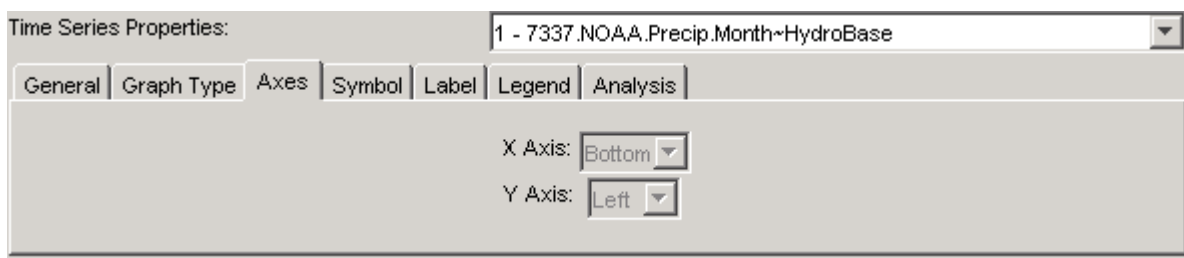
Time series **General** properties are currently view-only and indicate whether the time series is enabled for the graph.

Time Series Properties - Graph Type

TSView_TSProduct_Props_TS_GraphType

Example Time Series Graph Type Properties

Time series **Graph Type** properties are currently disabled. Currently all time series in a graph must have the same graph type.

Time Series Properties - Axes

TSView_TSProduct_Props_TS_Axes

Example Time Series Axes Properties

Time series **Axes** properties are currently view-only and show the graph axes to which a time series is associated.

Time Series Properties - Symbol

Time Series Properties: 1 - 7337.NOAA.Precip.Month~HydroBase

General Graph Type Axes **Symbol** Label Legend Analysis

Line Style: Solid Line Width: 1

Color: red Red Custom

Symbol Style: None Symbol Size: 0

TSView_TSProduct_Props_TS_Symbol

Example Time Series Symbol Properties

Time series **Symbol** properties define the graphical appearance of time series data. Properties are enabled/disabled based on the graph type (e.g., the **Symbol Style** will be disabled if the graph type is Bar). The symbol properties are consistent with the GeoView tools used for maps.

Time Series Properties - Label

Time Series Properties: 1 - 7337.NOAA.Precip.Month~HydroBase

General Graph Type Axes Symbol **Label** Legend Analysis

If the Label Format is Auto, defaults or the Graph Label Format will be used.

Position: Right

Format (see choices): %a - Weekday, abbr

TSView_TSProduct_Props_TS_Label

Example Time Series Label Properties

Time series **Label** properties allow the data label to be changed. Data points are not labeled by default because there are usually too many data labels to be legible. However, for plots with limited data, or after zooming in, labels can be useful to identify points without referring to tabular data. The label format can be defined using the choices next to the text field or by entering literal text. For an XY Scatter plot, repeat the %v format to show the independent (X) and dependent (Y) data values. See the **DataLabel** properties in the **Time Series Product Reference** section for label options.

Time Series Properties - Legend

Time Series Properties: 1 - 7337.NOAA.Precip.Month~HydroBase

General Graph Type Axes Symbol Label **Legend** Analysis

If the Format is Auto, defaults or the Graph Legend Format will be used.

Format (see choices): Auto

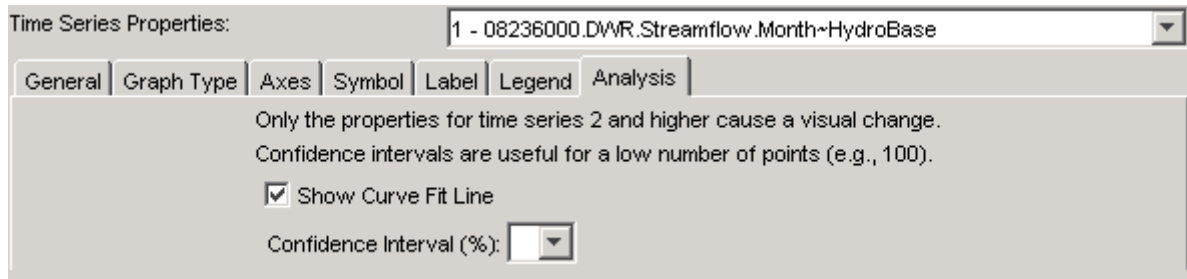
TSView_TSProduct_Props_TS_Legend

Example Time Series Legend Properties

Time series **Legend** properties allow the legend format to be changed. This is useful if the time series is to have different legend labeling than the other time series in the graph. If the **Legend Format** is Auto, a default legend format will be constructed from the time series description, identifier, and period of record.

See the LegendFormat property in the **Time Series Product Reference** section for legend formatting options.

Time Series Properties - Analysis



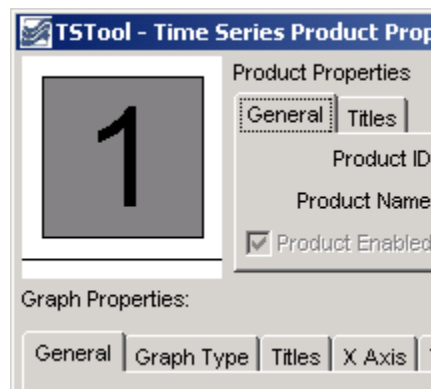
TSView_TSProduct_Props_TS_Analysis

Example Graph Analysis Properties

Time Series **Analysis** properties are available if the graph requires some type of analysis to produce the result (e.g., curve fitting).

Changing a Graph Page Layout

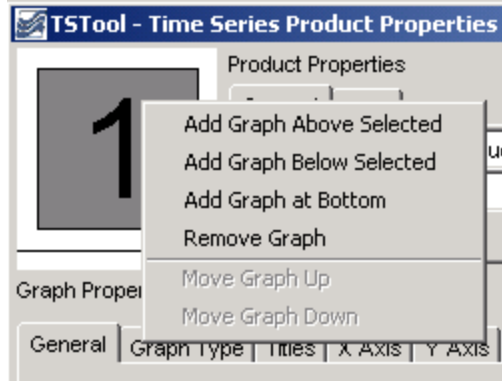
The default page layout for graphs is to display all time series in one graph. In this configuration, the layout area at the top-left corner of the time series product window will display as shown below:



TSView_Layout_1Graph

Layout Window Showing One Graph

The layout area can be used to split the single graph into multiple graphs on the page. For example, two graphs may be needed because of different units, time step, or graph type. Left clicking on a graph in the layout area will select the graph – the selected graph is shown in gray. Right clicking on the layout area displays a menu with available options:



TSView_Layout_Menu

Layout Window Menu

The actions taken by the menus are described below:

Add Graph Above Selected	Add a new graph above the selected graph, renumbering the graphs as needed.
Add Graph	Add a new graph below the selected graph, renumbering the graphs as needed.
Add Graph at Bottom	Add a new graph below all existing graphs, giving the new graph the next number in the sequence.
Remove Graph	Remove the selected graph, renumbering the graphs as needed.
Move Graph Up	Move the graph up one in the sequence, renumbering the graphs as needed. The menu is enabled only when multiple graphs are available.
Move Graph Down	Move the graph down one in the sequence, renumbering the graphs as needed. The menu is enabled only when multiple graphs are available.

When a new graph is added, it will not have any specific properties, time series data, or annotations, other than the default properties that are assigned (e.g., the default graph type is `Line`), and when drawn it will appear as a blank area. To see the graph, it will be necessary to set the graph's properties and provide it with data (and optionally, annotations). Properties and annotations are defined using the properties tabs as documented in previous sections – use the **Apply** button to apply and view the changes. To set graph properties, the graph to be modified should be selected from the choices at the top of the **Graph Properties** tab panel (or by selecting the graph in the layout window).

To add time series data to the new graph (or an existing graph), two approaches can be taken:

1. Find the time series to be moved using the list in the time series properties panel. It may be necessary to select a graph to find the time series – selecting a graph will not impact the ability to move the time series to a different graph. In the list of time series, hold the left mouse button down over a time series choice and drag the time series to a graph on the layout area. During this process, the cursor will change to a new shape, as shown below:



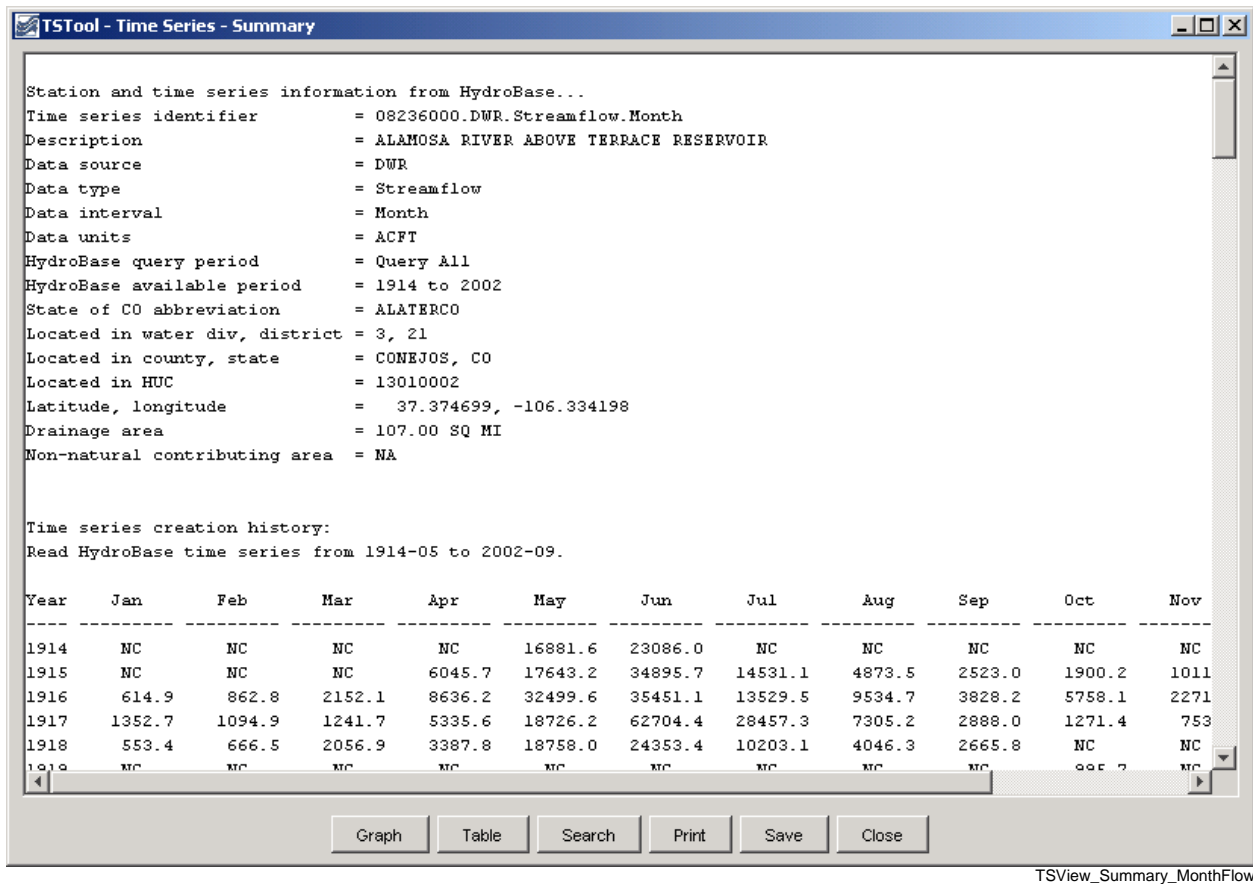
Release the mouse over the graph in the layout area that is to receive the time series. The time series will then be removed from the original graph and will be inserted into the new graph as the last time series in the list.

2. Some software programs will allow dragging a time series from a display to the time series product properties window. Similar to above, drag the time series onto the receiving graph in the layout area. Refer to documentation for the specific software program for additional information about whether this feature is available.

After adding a new graph and moving time series, it may be necessary to change the graph type for a graph. For example, the top graph may show precipitation and the bottom graph may show streamflow resulting from the precipitation. Precipitation is normally shown as bars and streamflow as a line. The graph will initially be shown using the graph type that was originally selected. Change the graph type in the new configuration, as appropriate, by selecting the graph to be changed and then use the **Graph Type** tab.

Time Series Summary View

The time series summary view can be selected from the graph or table view using the **Summary** button. Additionally, applications that use the TSView package may allow displaying a summary from a menu or button option. A time series summary view can usually be produced quickly, whereas the table view uses more resources. The following figure illustrates a typical summary view.



Example Summary View showing Monthly Streamflow

The summary view has the following characteristics:

- The graph view can be displayed using the **Graph** button and the table view can be displayed using the **Table** button.
- Each time series interval (e.g., Month, Day, Hour) has a default summary report format suitable for the interval. This format may be made more specific if time series are read from specific data types (e.g., if daily diversion time series are read from the HydroBase input type, the summary report will use the State of Colorado diversion coding report format).
- The contents of the view can be printed.
- The summary can be saved as a text file or DateValue time series file.
- Limited search capabilities are available to search for a string in the text area.

Time Series Table View

The time series table view can be selected from the graph or summary view using the **Table** button. Additionally, applications that use the TSView package may allow displaying a table from a menu or button option. The table view is useful for viewing date and data values in a spreadsheet-like display. A time series table view for a long period or many time series may require extra time to display, but usually only a few seconds are required. The following figure illustrates a typical table view.

DATE	08236000, Streamflow, ACFT	08236500, Streamflow, ACFT
1914-05	16881.6	
1914-06	23086.0	
1914-07		
1914-08		
1914-09		
1914-10		
1914-11		
1914-12		
1915-01		
1915-02		
1915-03		
1915-04	6045.7	
1915-05	17643.2	
1915-06	34895.7	
1915-07	14531.1	
1915-08	4873.5	
1915-09	2523.0	
1915-10	1900.2	
1915-11	1011.6	
1915-12	614.9	
1916-01	614.9	
1916-02	862.8	
1916-03	2152.1	
1916-04	8636.2	

Currently-selected worksheet: Interval base: Month x 1

TSView_Table_MonthFlow

Example Table View showing Monthly Streamflow

Characteristics of the table view are:

- The summary view can be displayed using the **Summary** button and the graph view can be displayed using the **Graph** button.
- The precision of dates matches the data interval for the time series.
- If time series with different intervals are selected, multiple tables will be displayed in the window.
- The table contents can be saved as a DateValue file, which is a useful delimited format file.

Time Series Product Reference

A *time series product* is a report, table, or graph, although currently TSView focuses on graph products. Examples of time series products and their use are:

- Reports and graphs generated from a database to perform quality checks.
- Reports and graphs generated from model input and output to check a calibration or model results.
- Reports and graphs generated from a database for real-time data products, to monitor current conditions or to create products for a web site.

The TSView package contains features to process time series product files in interactive and batch mode to produce time series products. Currently, only graph products are supported. The time series graph view allows a graph to be saved as a time series product file. This file describes the layout and contents of the product but does not include the time series data itself; therefore, the time series product is relatively small.

Time Series Product File Format

The time series product definition file format consists of comments (lines that start with #), sections (indicated by []), and simple property=value pairs. The following example illustrates the parts of a product file:

```
# Example Time Series Product file
# Comments start with #
# Sections are enclosed in [] and must be included

[Product]

# product properties - surround with double quotes if values contain spaces
xxxxx="xxxxxx  xx"

[SubProduct 1]

# "sub-product", e.g., a graph on a page (page is product and may have
# multiple graphs)

[Data 1.1]

# First data item in the SubProduct (e.g., first time series).
TSID = ...

[Data 1.2]

# Second data item in the SubProduct (e.g., first time series).
TSID = ...

[SubProduct 2]

[Data 2.1]

# Annotations are associated with a SubProduct
[Annotation 2.1]

Annotation properties...
... etc. ...
```

Example Time Series Product File

Most properties, if not specified in the file, will default to reasonable values. The most important property is TSID, which indicates time series identifier to be read for data. The time series identifier follows the conventions described in the **Time Series Terminology** section. Some tools, like TSTool, will match the TSID against time series that have already been read into memory, or, if necessary, read the time series from a file or database if not in memory. The normal convention is to use a *.tsp* extension for time series product file names.

The list of properties that can be used in a time series product definition file is quite extensive and new properties are added as new features are enabled. As shown in the previous section, properties are defined as simple `variable=value` pairs. These properties are used internally by the graph view (and its properties window) regardless of whether the graph originated from a product file or interactively. The following tables list the properties that are currently supported or envisioned to be enabled in the future. The first set of properties are used to define the overall product (the full page).

Top-level Time Series Product Properties

Product Property	Description	Default
Current DateTime	The current date and time to be drawn as a vertical line on all graphs. If the property is not specified, no current date/time line will be drawn. If specified as <code>Auto</code> , the current system time will be used for the date/time. If specified as a valid date/time string (e.g., <code>2002-02-05 15</code>), the string will be parsed to obtain the date/time. This property is often specified internally by the application at run time.	Not drawn.
Current DateTime Color	Color to use to draw the current date and time. Colors can be specified as named colors (e.g., <code>red</code>), hexadecimal RGB values (e.g., <code>0xFF0000</code>), integer triplets (e.g., <code>255, 0, 0</code>) or floating point triplets (e.g., <code>1.0, 0.0, 0.0</code>).	Green
Enabled	Indicates whether the product should be processed. Specify as <code>True</code> or <code>False</code> .	True
LayoutNumber OfColumns	The number of columns in the product.	Currently always 1.
LayoutNumber OfRows	The number of rows in the product.	Currently equal to the number of graphs.
LayoutType	Indicates how the graphs in a product are laid out. Only <code>Grid</code> is supported.	Grid
MainTitle FontName	Name of font to use for main title (Arial, Courier, Helvetica, TimesRoman).	Arial
MainTitle FontSize	Size, in points, for main title.	20
MainTitle FontStyle	Font style (Bold, BoldItalic, Plain, PlainItalic).	Plain
MainTitle String	Main title for the product, centered at the top of the page.	No main title.
OutputFile	Output file when graph product is generated in batch mode. This property is often set at run time by the application. This property is ignored for <code>ProductType=Report</code> and must be specified at a subproduct level.	<code>C:\TEMP\tmp.png</code> on windows, <code>/tmp/tmp.png</code> on UNIX
Owner	An identifier that indicates the owner of the TSProduct, used internally when saving TSProduct definitions to a database that implements permissions.	None – can be blank if permissions are not important.

Top-level Time Series Product Properties (continued)

Product Property	Description	Default
PeriodEnd	Ending date for time series data in the product. The date should be formatted according to common conventions (e.g., YYYY-MM-DD HH:mm), and should ideally be of appropriate precision for the data being queried. This property is often set at run time by the application.	Full period is read.
PeriodStart	Starting date for time series data in the product. The date should be formatted according to common conventions (e.g., YYYY-MM-DD HH:mm), and should ideally be of appropriate precision for the data being queried. This property is often set at run time by the application.	Full period is read.
PreviewOutput	Indicates whether the product should be visually previewed before output. This property is often set at run time by the application and is used to override generation of the OutputFile.	false
ProductType	Time series product type, one of: <ul style="list-style-type: none"> Graph – graph (see graph subproduct properties). Report – report (see report subproduct properties). 	Graph
SubTitleFontName	Name of font to use for subtitle (see MainTitleFontName for font list).	Arial
SubTitleFontSize	Size, in points, for subtitle.	10
SubTitleFontStyle	Font style (see MainTitleFontStyle for style list).	Plain
SubTitleString	Subtitle for the product.	No subtitle.
TotalHeight	Height of the total drawing space, which may include multiple graphs, pixels.	400
TotalWidth	Width of the total drawing space, which may include multiple graphs, pixels.	400

The subproduct properties are associated with the graphs on a page or report files. There can be one or more graphs on a page, each with different properties. It is envisioned that graphs can be grouped into several zoom groups, where zooming in on one graph will cause all graphs to scale similarly. However, at this time, all graphs in a product are placed in a single zoom group. It is also envisioned that graphs could be placed anywhere on the page; however, at this time, multiple graphs on a page can only be stacked vertically, each using the full width of the page.

The following tables describe the subproduct (graph) properties.

Subproduct (Graph) Properties

Subproduct (Graph) Property	Description	Default
BarPosition	For use with bar graphs. This property controls how bars are positioned relative to the date and can have the values CenteredOnDate, LeftOfDate, or RightOfDate.	CenteredOnDate
BottomXAxisLabelFontName	Name of font for bottom x-axis labels (see Product MainLabelFontName).	Arial
BottomXAxisLabelFontSize	Bottom x-axis labels font size, points.	10
BottomXAxisLabelFontStyle	Bottom x-axis labels font style (see Product MainLabelFontStyle).	Plain
BottomXAxisTitleFontName	Name of font for bottom x-axis title (see Product MainTitleFontName).	Helvetica
BottomXAxisTitleFontSize	Bottom x-axis title font size, points.	12
BottomXAxisTitleFontStyle	Bottom x-axis title font style (see Product MainTitleFontStyle).	Plain
BottomXAxisLabelFormat	Format for X-axis labels. Currently this is confined to date/time axes and only MM-DD is recognized.	Determined automatically.
BottomXAxisMajorGridColor	Color to use for the major grid.	Most graph types automatically set to None.
BottomXAxisMinorGridColor	Color to use for the minor grid. This property is not implemented.	None
BottomXAxisTitleString	Bottom X-axis title string.	As appropriate for the graph type (often None if dates).
DataLabelFontName	Name of font for data labels (see Product MainLabelFontName).	Arial
DataLabelFontSize	Data label font size, points.	10
DataLabelFontStyle	Data label font style (see Product MainLabelFontStyle).	Plain

Subproduct (Graph) Properties (continued)

Subproduct (Graph) Property	Description	Default
DataLabelFormat	Format specifiers to use for labeling data points. If blank, no labels will be drawn. If specified, labels are drawn for line graphs and XY scatter plots. The following format specifiers are available (all other text in the format is treated literally). The last three specifiers are related to time series data and all others are related to the date for a point. The %v specifier can be specified twice for XY Scatter plots to display the X and Y values. If specified and the time series data property is not specified, the graph property will be used.	Blank (no data point labels).
	%%	
	%a	
	%A	
	%B	
	%b	
	%d	
	%H	
	%I	
	%J	
	%m	
	%M	
	%p	
	%S	
	%Y	
	%Y	
	%Z	
	%v	
	%U	
	%q	

Subproduct (Graph) Properties (continued)

Subproduct (Graph) Property	Description	Default
DataLabelPosition	Indicates the position of data labels, relative to the data point: UpperRight, Right, LowerRight, Below, LowerLeft, Left, UpperLeft, Above, Center. If specified and the time series data property is not specified, the graph property will be used.	Right
Enabled	Indicates whether the sub-product should be processed. Specify as True or False.	True
GraphHeight	Graph height in pixels. Currently this property is ignored (use Product TotalHeight instead).	Product TotalHeight (minus space for titles, etc.) if one graph, or an even fraction of Product TotalHeight (minus space for titles, etc.) if multiple graphs.
GraphType	Indicates the graph type for all data in a graph product. Available options are: Bar, Duration, Line, PeriodOfRecord, Point, XY-Scatter.	Line
GraphWidth	Graph width in pixels. Currently this property is ignored (use Product TotalWidth instead).	Product TotalWidth (minus space for titles, etc.).
LayoutXPercent	For the product grid layout, the width of the graph as a total width of the product, percent.	100 divided by the number of columns in the layout.
LayoutYPercent	For the product grid layout, the height of the graph as a total width of the product, percent.	100 divided by the number of rows in the layout.
LeftYAxisIgnoreUnits	Indicates whether to ignore units for the left Y-axis. Normally, units are checked to make sure that data can be plotted consistently. If this property is set, then the user will not be prompted at run-time to make a decision. Specify as True or False.	If not specified, the units will be checked at run-time and, if not compatible, the user will be prompted to indicate whether to ignore units in the graphs. The property will not be reset automatically but will be handled internally using the interactively supplied value.

Subproduct (Graph) Properties (continued)

Subproduct (Graph) Property	Description	Default
LeftYAxisLabelFontName	Name of font for left y-axis labels (see Product MainLabelFontName).	Arial
LeftYAxisLabelFontSize	Left y-axis labels font size, points.	10
LeftYAxisLabelFontStyle	Left y-axis labels font style (see Product MainLabelFontStyle).	Plain
LeftYAxisLabelPrecision	If numeric data, the number of digits after the decimal point in labels.	Automatically determined from graph type and/or data units.
LeftYAxisMajorGridColor	Color to use for the major grid.	Most graph types automatically set to lightgray.
LeftYAxisMax	Maximum value for the left Y-Axis.	Auto, automatically determined. If the actual data exceed the value, the property will be ignored.
LeftYAxisMin	Minimum value for the left Y-Axis.	Auto, automatically determined. If the actual data exceed the value, the property will be ignored.
LeftYAxisMinorGridColor	Color to use for the minor grid. This property is not implemented.	None
LeftYAxisTitleFontName	Name of font for left y-axis title (see Product MainTitleFontName).	Arial
LeftYAxisTitleFontSize	Left y-axis title font size, points.	12
LeftYAxisTitleFontStyle	Left y-axis title font style (see Product MainTitleFontStyle).	Plain
LeftYAxisTitleString	Left y-axis title string. Note that due to limitations in Java graphics, the left y-axis title is placed at the top of the left y-axis so that it takes up roughly the same space as the y-axis labels. The top-most label is shifted down to make room for the title.	As appropriate for the graph type (often the data units).
LeftYAxisType	Left y-axis type (Log, or Linear).	Linear
LeftYAxisUnits	Left y-axis units. This property is currently used internally and full support is being phased in. See also LeftYAxisIgnoreUnits.	Units from first valid time series, or as appropriate for the graph type.
LegendFontName	Name of font for legend (see Product MainTitleFontName).	Arial
LegendFontSize	Legend font size, points.	10
LegendFontStyle	Legend font style (see Product MainTitleFontStyle).	Plain

Subproduct (Graph) Properties (continued)

Subproduct (Graph) Property	Description	Default
LegendFormat	The legend format is composed of literal characters and/or time series data format specifiers, as follows.	Auto, which uses Description, Identifier, Units, Period
	Blank No legend will be displayed.	
	%% Literal percent	
	%A Time series alias	
	%D Description (e.g., RED RIVER BELOW MY TOWN)	
	%F Full time series identifier (e.g., XX_FREE.USGS.QME.24HOUR.Trace1)	
	%I Full interval part of the identifier (e.g., 24Hour).	
	%b Base part of the interval (e.g., Hour).	
	%m Multiplier part of the interval (e.g., 24).	
	%L Full location part of the identifier (e.g., XX FREE).	
	%l Main part of the location (e.g., XX).	
	%w Sub-location (e.g., FREE).	
	%S The full source part of the identifier (e.g., USGS).	
	%s Main data source (e.g., USGS).	
	%x Sub-source (reserved for future use).	
	%T Full data type (e.g., QME).	
	%t Main data type.	
	%k Sub-data type.	
	%U Data units (e.g., CFS).	
	%z Sequence number (used with traces).	
	%Z Scenario part of identifier (e.g., Trace1).	
LegendPosition	Position of the legend relative to the graph: Bottom, InsideLowerLeft, InsideLowerRight, InsideUpperLeft, InsideUpperRight, Left, None, Right.	Bottom

Subproduct (Graph) Properties (continued)

Subproduct (Graph) Property	Description	Default
MainTitleFontName	Name of font to use for graph main title (see Product MainTitleFontName).	Arial
MainTitleFontSize	Size, in points, for graph main title.	10
MainTitleFontStyle	Graph main title font style (see Product MainTitleFontStyle).	Plain
MainTitleString	Main title for the graph.	None, or appropriate for graph type.
PeriodEnd	Ending date for time series data in the sub-product. The date should be formatted according to common conventions (e.g., YYYY-MM-DD HH:mm), and should ideally be of appropriate precision for the data being queried. This property is often set at run time.	Full period is read.
PeriodStart	Starting date for time series data in the sub-product. The date should be formatted according to common conventions (e.g., YYYY-MM-DD HH:mm), and should ideally be of appropriate precision for the data being queried. This property is often set at run time.	Full period is read.
RightYAxisLabelFontName	Name of font for right y-axis labels (see Product.MainLabelFontName). This property is not enabled.	Arial
RightYAxisLabelFontSize	Right y-axis labels font size, points. This property is not enabled.	10
RightYAxisLabelFontStyle	Right y-axis labels font style (see Product MainLabelFontStyle). This property is not enabled.	Plain
RightYAxisTitleFontName	Name of font for right y-axis title (see Product MainTitleFontName). This property is not enabled.	Arial
RightYAxisTitleFontSize	Right y-axis title font size, points. This property is not enabled.	12
RightYAxisTitleFontStyle	Right y-axis title font style (see Product MainTitleFontStyle). This property is not enabled.	Plain
RightYAxisTitleString	Right y-axis title string. This property is not enabled.	

Subproduct (Graph) Properties (continued)

Subproduct (Graph) Property	Description	Default
SubTitleFontName	Name of font to use for graph Sub title (see Product MainTitleFontName).	Arial
SubTitleFontSize	Size, in points, for graph sub title.	10
SubTitleFontStyle	Graph sub title font style (see Product MainTitleFontStyle).	Plain
SubTitleString	Sub title for the graph.	No subtitle.
TopXAxisLabelFontName	Name of font for Top x-axis labels (see Product.MainLabelFontName). This property is not enabled.	Arial
TopXAxisLabelFontSize	Top x-axis labels font size, points. This property is not enabled.	10
TopXAxisLabelFontStyle	Top x-axis labels font style (see Product MainLabelFontStyle). This property is not enabled.	Plain
TopXAxisTitleFontName	Name of font for Top x-axis title (see Product MainTitleFontName). This property is not enabled.	Arial
TopXAxisTitleFontSize	Top x-axis title font size, points. This property is not enabled.	12
TopXAxisTitleFontStyle	Top x-axis title font style (see Product MainTitleFontStyle). This property is not enabled.	Plain
TopXAxisTitleString	Top X axis title string. This property is not enabled.	As appropriate for the graph type.
XYScatterAnalyzeForFilling	Indicate whether the analysis should be used to analyze for filling. If true, then the XYScatterIntercept, XYScatterFillPeriodStart, and XYScatterFillPeriodEnd properties may be specified.	False
XYScatterDependentAnalysisPeriodEnd	Specify the ending date/time for the period to analyze the dependent time series data, to determine the best-fit line.	Blank (analyze full period).
XYScatterDependentAnalysisPeriodStart	Specify the starting date/time for the period to analyze the dependent time series data, to determine the best-fit line.	Blank (analyze full period).
XYScatterFillPeriodEnd	When XYScatterAnalyzeForFilling=true, indicates the ending date/time of the period to fill, using standard date/time string.	Blank (fill full period).

Subproduct (Graph) Properties (continued)

Subproduct (Graph) Property	Description	Default
XYScatterFillPeriodStart	When XYScatterAnalyzeForFilling = true, indicates the starting date/time of the period to fill, using standard date/time string.	Blank (fill full period).
XYScatterIndependentAnalysisPeriodEnd	Specify the ending date/time for the period to analyze the independent time series data, to determine the best-fit line.	Blank (analyze full period).
XYScatterIndependentAnalysisPeriodStart	Specify the starting date/time for the period to analyze the independent time series data, to determine the best-fit line.	Blank (analyze full period).
XYScatterIntercept	The value of A in the best-fit equation $A + bX$. If specified, the value of B is adjusted accordingly. This property cannot be used with transformed data and if specified must be 0.	Blank (do not force the intercept).
XYScatterMethod	Curve fit method used when analyzing data for the XY Scatter graph (OLSRegression or MOVE2).	OLSRegression
XYScatterMonth	One or more month numbers used when analyzing data for the XY Scatter graph, separated by commas or spaces (1=Jan).	Blank (analyze all)
XYScatterNumberOfEquations	Number of equations used when analyzing data for the XY Scatter graph (OneEquation or MonthlyEquations).	OneEquation
XYScatterTransformation	Data transformation used when analyzing data for the XY Scatter graph (None or Log). This property is not enabled.	None
ZoomEnabled	Indicates whether the graph can be zoomed (true) or not (false).	Graph types are evaluated and the property is automatically set. XY-Scatter and Duration graphs can't zoom.
ZoomGroup	Indicate a group identifier that is used to associate graphs for zooming purposes. For example, there may be more than one distinct group of graphs, each with its own overall period or data limits. The graph types may also be incompatible for zooming. This is an experimental feature and should currently not be specified in product files.	All graphs are assigned to zoom group 1.

Report Subproduct Properties

The following table describes the subproduct (report) properties. Limited support for report products are currently enabled. Reports are defined as any format other than graphical output, including raw data formats like delimited and DateValue files. The number of properties for reports will continue to be expanded as additional features are enabled. An example of a report product file is as follows (in this case for NWSRFS FS5Files input type time series):

```
[Product]

ProductType = Report
Enabled = true

[SubProduct 1]

OutputFile = C:\Report_6_Hour
ReportType = DateValue

[Data 1.1]
TSID = FZRDR.NWSRFS.SPEL.6HOUR~NWSRFS_FS5Files

[SubProduct 2]

OutputFile = C:\Report_24_Hour
ReportType = DateValue

[Data 2.1]
TSID = FZRDR.NWSRFS.PELV.24HOUR~NWSRFS_FS5Files
```

Subproduct (Report) Properties

Subproduct (Report) Property	Description	Default
OutputFile	Output file when report product is generated in batch mode. If a relative path is given, the file will be written relative to the working directory for the software. This property is often set at run time by the application.	<i>C:\TEMP\tmp_report_N</i> on windows, <i>/tmp/tmp_report_N</i> on UNIX
Enabled	Indicates whether the sub-product should be processed. Specify as true or false.	true

Time Series Properties

Each subproduct (graph) includes time series data, and the presentation of each time series can be configured using data (time series) properties. In some cases, properties are layered, allowing a property to be defined for the subproduct (graph) for use by all time series (e.g., legend text).

The following tables list data (time series) properties.

Data (Time Series) Properties

Data (Time Series) Property	Description	Default
Color	Color to use when drawing the data. Examples are named colors (e.g., red), RGB triplets (e.g., 255, 0, 128), and hexadecimal RGB (e.g., 0xFF0088).	Repeating, using common colors.
DataLabelFormat	Data label format specifiers. See the graph DataLabelFormat property. If the graph property is specified and the time series property is not, the graph property will be used.	Blank (no labels).
DataLabelPosition	Data label position. See the graph DataLabelPosition property. If the graph property is specified and the time series property is not, the graph property will be used.	Right
Enabled	Indicates whether the data should be processed. Specify as true or false.	true
GraphType	Indicates the graph type for the data in a graph product. Available options are: Bar, Duration, Line, PeriodOfRecord, Point, XY-Scatter. Currently the sub-product property is used for all data. It is envisioned that this property will be enabled in the future to allow different data representations to be plotted together (e.g., monthly as bars, daily as line).	Property not enabled.
LegendFormat	The legend for the data can be specified and will override the SubProduct LegendFormat property (see that property for details).	Auto
LineStyle	Line style. Currently only None (e.g., for symbols only) and Solid are allowed.	Solid
LineWidth	Line width, pixels. Currently a line width of 1 pixel is always used.	1

Data (Time Series) Properties (continued)

Data (Time Series) Property	Description	Default
PeriodEnd	Ending date for time series data in the data item. The date should be formatted according to common conventions (e.g., YYYY-MM-DD HH:mm), and should ideally be of appropriate precision for the data being queried. This property is often set at run time.	Full period is read.
PeriodStart	Starting date for time series data in the data item. The date should be formatted according to common conventions (e.g., YYYY-MM-DD HH:mm), and should ideally be of appropriate precision for the data being queried. This property is often set at run time.	Full period is read.
RegressionLineEnabled	Indicates whether the regression line should be shown (currently only used with the XY-Scatter graph type). The line is drawn in black (there is currently not a property to set the line color).	true
SymbolSize	Symbol size in pixels.	0 (no symbol)
SymbolStyle	Symbol style. Recognized styles are: <ul style="list-style-type: none"> • None • Arrow-Down, Arrow-Left, Arrow-Right, Arrow-Up • Asterisk • Circle-Hollow, Circle-Filled • Diamond-Hollow, Diamond-Filled • Plus, Plus-Square • Square-Hollow, Square-Filled • Triangle-Down-Hollow, Triangle-Down-Filled, Triangle-Left-Hollow, Triangle-Left-Filled, Triangle-Right-Hollow, Triangle-Right-Filled, Triangle-Up-Hollow, Triangle-Up-Filled • X, X-Cap, X-Diamond, X-Edge, X-Square 	None

Data (Time Series) Properties (continued)

Data (Time Series) Property	Description	Default
TSID	Time series identifier.	Must specify.
XAxis	X-axis to use (Bottom or Top). This currently always defaults to bottom.	Bottom
XYScatterConfidenceInterval	This property is only used with XY scatter plots. If not blank, the value indicates that confidence level lines should be drawn on the XY Scatter plot for the given confidence interval, percent. Currently only 99 and 95 percent confidence intervals are supported. The lines will only be drawn if the curve fit line is drawn (see <code>RegressionLineEnabled</code>).	Blank (do not draw).
YAxis	Y-axis to use (Left or Right). This currently always defaults to left.	Left

Annotation Properties

Annotations are associated with subproducts (graphs) and are implemented as simple shapes that are drawn on normal graphs. It is envisioned that all shapes supported by the drawing package will eventually be supported but currently only text labels and lines can be specified as annotations.

To allow flexibility, annotations can be placed using two coordinate systems. For example, if a product is generated using real-time data, the date/time axis will have a different range over time. Therefore, placing an annotation using a fixed coordinate would cause the annotation to scroll off the graph as time passes. To resolve this issue and still allow absolute positioning of annotations, as appropriate, the following coordinate systems are supported, as specified by the `XAxisSystem` and `YAxisSystem` properties:

Data When using the data coordinate system, it is expected that the coordinates used to define the annotation will agree with the data units being drawn. For example, for a normal time series graph, the x-axis coordinate would be specified as a date/time to the necessary precision and the y-axis coordinate would be specified using data values.

It is envisioned that a notation +NNN and -NNN will be implemented in the future to allow offsets from the edges of the graph, in data units.

Percent When using the percent coordinate system, it is expected that the coordinates used to define the annotation are specified as a percent of the graph width or height, with 0 being the lower/left and 100 being the upper/right.

Each axis can have a different coordinate system (e.g., the y-axis value can be set using data units and the x-axis value can be set using percent).

The following tables list annotation properties.

Annotation Properties (All Shapes)

Annotation Property	Description	Default
AnnotationID	A string that identifies the annotation, to be used in software displays. If there are many annotations, this helps identify them when editing.	Annotation + annotation number (1+) (e.g., Annotation1).
Color	Color to use when drawing the annotation. Examples are named colors (e.g., red), RGB triplets (e.g., 255, 0, 128), and hexadecimal RGB (e.g., 0xFF0088).	Black
Order	The drawing order for the annotation, either BehindData to draw behind time series data or OnTopOfData to draw on top of time series data.	OnTopOfData
ShapeType	The type of shape to be drawn for the annotation. Currently accepted values are Text and Line.	None – must be specified.
XAxisSystem	Indicates the system for X coordinates: <ul style="list-style-type: none">• If Data, the X coordinates that are specified will be in data units.• If Percent, the X coordinates are percent of the graph (0% is left and 100% is right).	Data
YAxisSystem	Indicates the system for Y coordinates: <ul style="list-style-type: none">• If Data, the Y coordinates that are specified will be in data units.• If Percent, the Y coordinates are percent of the graph (0% is bottom and 100% is top).	Data

Annotation Properties (Line Shape)

Annotation Property	Description	Default
LineStyle	Line style. Currently only None and Solid are allowed.	Solid
LineWidth	Line width, pixels. Currently a line width of 1 point (pixel) is always used.	1
Points	X and Y coordinates for the line endpoints, as follows: X1,Y1,X2,Y2 or X1,Y2,X2,Y2.	None – must be specified.

Annotation Properties (Text Shape)

Annotation Property	Description	Default
FontSize	Annotation text font size, points.	10
FontStyle	Annotation text font style (see Product MainLabelFontStyle).	Plain
FontName	Annotation font name (see Product MainTitleFontName).	Arial
Point	X and Y coordinates for the text position, as follows: X1,Y1	None – must be specified.
Text	The string to display.	Blank
TextPosition	Indicates the position of text, relative to the point: UpperRight, Right, LowerRight, Below, LowerLeft, Left, UpperLeft, Above, Center.	Right

This page is intentionally blank.

Appendix: GeoView Mapping Tools

Color, 2004-05-27, Original Maintained with TSTool, Acrobat Distiller

Overview
GeoView Terminology
The GeoView Panel
Interacting with the GeoView Map
Setting GeoView Properties
Viewing a Layer's Attributes
Using GeoView with a Software Application
Limitations
GeoView Configuration – the GeoView Project File
Color Specification
Color Tables
Symbol Style – Point Data
Classification
GeoView Project File Examples

Overview

The GeoView package contains integrated software components that can be used with software to enable map-based interfaces. The main purpose of the GeoView package is to provide simple and flexible map displays that can be used in a variety of software applications with little or no reconfiguration.

The GeoView package has been developed by Riverside Technology, inc., using Java technology. GeoView interfaces can be embedded in Java applications (e.g., use GeoView as the main window interface), can be enabled as a separate floating window (e.g., to support an application's features without being embedded in the main window), and can be used in web pages either as embedded map applets or stand-alone map windows. GeoView tools operate similarly on Microsoft Windows and UNIX operating systems.

This appendix describes general GeoView features and can be used as a reference for how to configure and use GeoView components. Specific uses of GeoView in a software program are discussed in the documentation for the specific software. Some of the figures shown in this documentation were generated using GeoView with the TSTool software and consequently title bars include TSTool in the wording.

GeoView Terminology

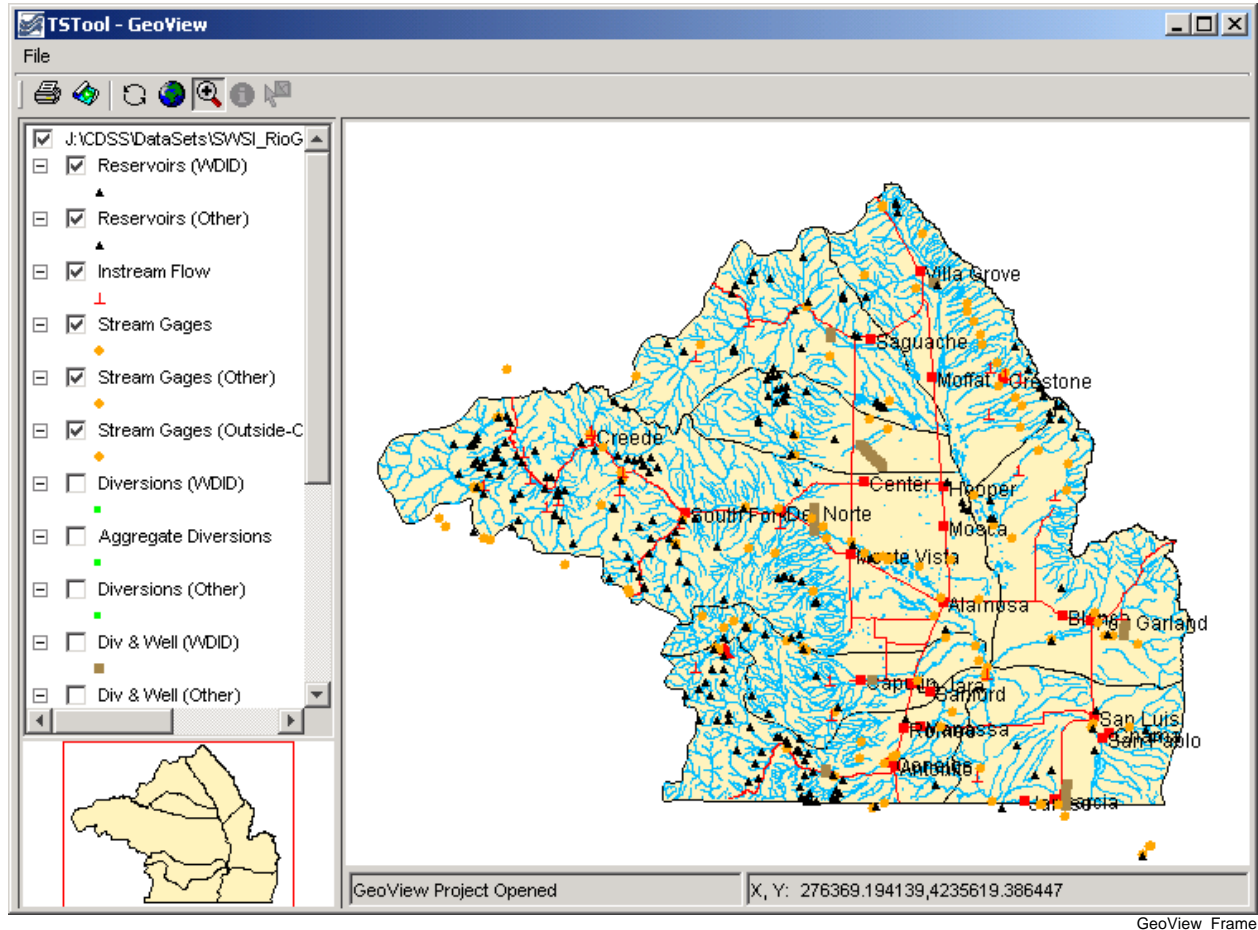
GeoView terminology is similar to other GIS product terminology. Important terms are shown in the following table. These terms are used infrequently in most user interfaces and applications but are visible at times in property dialogs and configuration files.

GeoView Terminology

Term	Description
<i>GeoView</i>	The visible map window where maps are displayed. Currently there can be only one main GeoView. A reference GeoView may be used to show the zoom extents in the main GeoView.
<i>GeoLayer</i>	A data layer, in its "raw" form (e.g., an ESRI Shapefile). The more generic term "layer" is often used.
<i>GeoLayerView</i>	A view of a GeoLayer, with symbol properties for visualization. The more generic term "layer view" is often used. This is equivalent to a "theme" in some software packages.
<i>Feature</i>	A general term describing an item on the map, consisting of shape and attribute data.
<i>Shape</i>	A general term defining the type of feature (e.g., point, polygon). The shape type defines the ways that a feature can be symbolized and used in analysis. Shape types can typically be determined automatically from input data.
<i>Attributes</i>	A general term defining non-shape data that are associated with a feature. Often, attributes are stored in a tabular form, such as a relational database table. Attributes are usually associated with a shape using some type of index number (shape index).
<i>Symbol</i>	The combination of properties used to visualize a layer (e.g., symbol style, color, labeling, classification). The feature shape type controls how the feature can be symbolized.
<i>GeoView Project</i>	A GeoView Project file (.gvp) can be used to define the layers and global viewing properties for a GeoView. The contents of this file are described in more detail in the GeoView Configuration – the GeoView Project File section at the end of this appendix.
<i>Application Layer Type</i>	Because GeoView is a generic tool, it has no implicit understanding of the types of data that are important to an application. The <code>AppLayerType</code> is a property that can be assigned to layers in a GeoView Project file to help an application know that a layer is important to the application. An application layer type of "BaseLayer" indicates that a layer should be used for background information and is not specific to the application. See the Using GeoView with a Software Application section below for more information.

The GeoView Panel

An example GeoView interface is shown in the following figure. This example uses a floating GeoView window. Some programs use a GeoView that is embedded in the main application window, and some rely on secondary map windows (as shown below). If the map is in the main window, the menus at the top of the window will be those specific to the software (whereas below the single GeoView **File** menu is shown).



Example GeoView Interface (from TSTool)

The GeoView Panel is a self-contained component that offers a standard map-based interface that can be used in many applications. In the above figure, the GeoView Panel includes everything shown, except for the top menu bar (with the **File** menu). The general purpose GeoView Frame includes the menu bar and a GeoView Panel. The GeoView Panel contains the following components:

Table of Contents
(left edge)

The table of contents displays a list of layer views, showing the top-most layer at the top of the legend. Layers can be enabled/disabled by toggling the check box. A layer can be selected/deselected by clicking on the layer in the table of contents. Layers that are selected can be acted on (e.g., properties can be viewed). The table of contents also indicates the symbol for the layer.

Main GeoView
(large map)

The main GeoView displays the enabled layers and allows you to interact with the map using the mouse and keyboard (e.g., zoom, select).

Reference GeoView (lower left) The reference GeoView displays layers that have the property `ReferenceLayer` set to `true`. This view shows the current zoom extent relative to the maximum extent of the data and can also be used to initiate a zoom to a region on the main map (the reference map is always in zoom mode).

Standard Controls (top, below menu bar) Standard controls perform actions on the visible map as follows:



Print

Print the visible map. You will be able to pick the printer and orientation.



Save Image

Saves the map as a Portable Network Graphics (PNG), JPEG, or other supported graphic file format.



Refresh

Refresh the map display by redrawing features in enabled layers that are in the visible window. This does **not** re-read the original data. GeoView normally refreshes automatically as needed.



Zoom Out

Zoom to the maximum data extents.



Select the Mode as Zoom, Info, or Select

Select the interaction mode. The **Zoom** mode allows a rectangle to be drawn on the map to zoom to the specified region and is the default mode if no layer is selected. The **Info** mode allows features to be selected (by clicking on or drawing a box around), after which geographic information about the features is displayed. The **Select** mode is similar to **Info**; however, its purpose is to select features for an additional action (e.g., exporting data or performing a query). The **Info** and **Select** modes are only enabled if one or more layers are selected in the table of contents.





See the next section for more information about using these features.

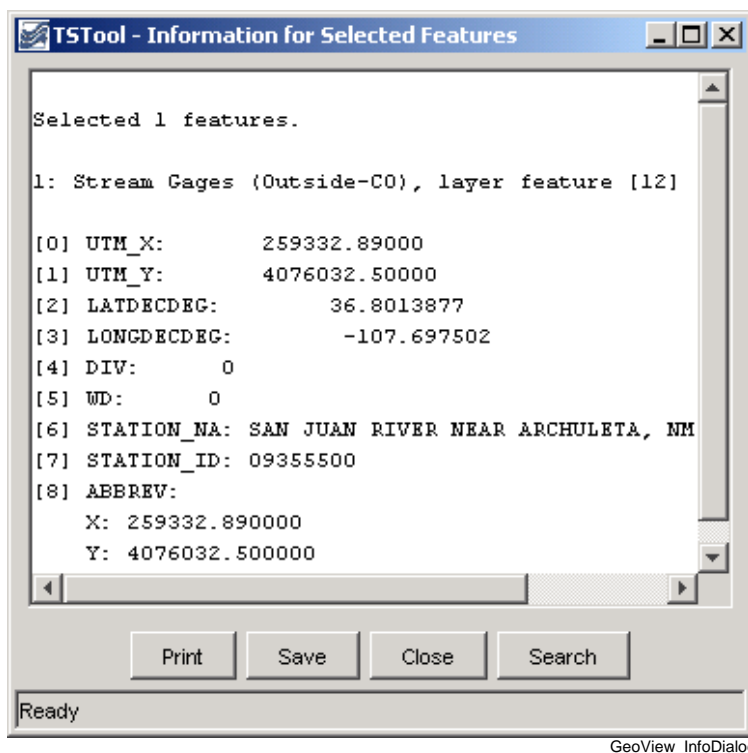
Message Areas (bottom) The message areas are used to display the mouse coordinates and provide other feedback.

The GeoView Panel components work with each other to provide interaction with the maps, as described below.

Interacting with the GeoView Map

The layers shown on the map are initially displayed according to the GeoView Project settings. Once displayed, you can interact with the map in the following ways:


Disable/Enable a layer view	<p>Layers can be enabled/disabled to make the map more readable or useful:</p> <ol style="list-style-type: none"> 1. Use the check boxes in the table of contents to disable and enable layer views, as appropriate. The map will automatically refresh, resulting in a slight delay as the map is redrawn. 2. If necessary, use the Refresh tool () to cause the map to be updated (automatic refresh may be disabled for some applications, due to performance reasons).
Change layer view order	<p>Currently the layer view order can only be changed by editing the GeoView Project file.</p>
Zoom in/out	<p>Zooming is useful make symbols and labels more readable. To zoom in:</p> <ol style="list-style-type: none"> 1. Set the GeoView interaction mode to "Zoom" by selecting the zoom tool () at the top of the window. 2. Use the mouse to draw a box around an area of interest (left mouse button down to start, move the mouse, and then release). The main GeoView map will zoom to the selected region and the reference map will show the zoom extent. 3. Use the Zoom Out tool () to zoom to the full extent or use the reference GeoView to zoom to a different region.
Change symbols for a layer view	<p>To change the symbols and labels for a layer view:</p> <ol style="list-style-type: none"> 1. Select the layer view in the table of contents 2. Right-click and select the Properties menu. See the Setting GeoView Properties section below for information about the properties.
Display geographic information for features	<p>The GeoView interface can display information about geographic features (shape and attribute data) from the original geographic data. To do so:</p> <ol style="list-style-type: none"> 1. Select layer views in the table of contents that are to be searched for information. 2. Set the GeoView interaction mode to Info () 3. Click near the feature or draw a box around multiple features. The layers will be searched and the following dialog will be shown.



The resulting dialog will show information about the selected features, including basic layer information, and information about the specific shapes and attributes. **The display is for geographic data only. Attribute names and values are as they appear in the original data. Additional application-specific data are typically provided by a separate software interface.**

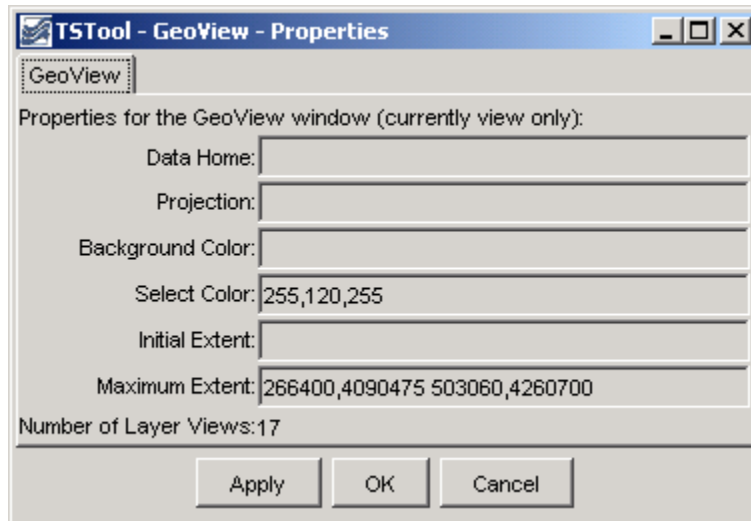
Select features

Features can be selected for a number of reasons. Currently, GeoView has limited select tools, which are mainly used internally when integrated with an application (e.g., an application can select features internally, which are then highlighted on the map). In the future, interfaces to select features from the GeoView interface using query criteria may be added.

Features can be selected () similar to the **Info** mode described above. The selected features are highlighted on the map. In the past, yellow, or cyan have been used to highlight selected features. However, yellow is not clearly visible when earth-tone colors are used for background layers and cyan is not clearly visible when water-tone colors are used for background layers. Therefore, GeoView is phasing in a magenta/pink selection color, which is rarely used for background layers.

Setting GeoView Properties

GeoView properties are initially set in a GeoView Project file or are assigned internally by the software. Most properties control how layers are displayed (colors, labels, etc). To view general GeoView properties, right click on the GeoView map and select the **Properties** menu. Some properties are currently view-only. Refer to the **GeoView Configuration – the GeoView Project File** section below for a complete list of properties that can be defined in a GeoView Project file.



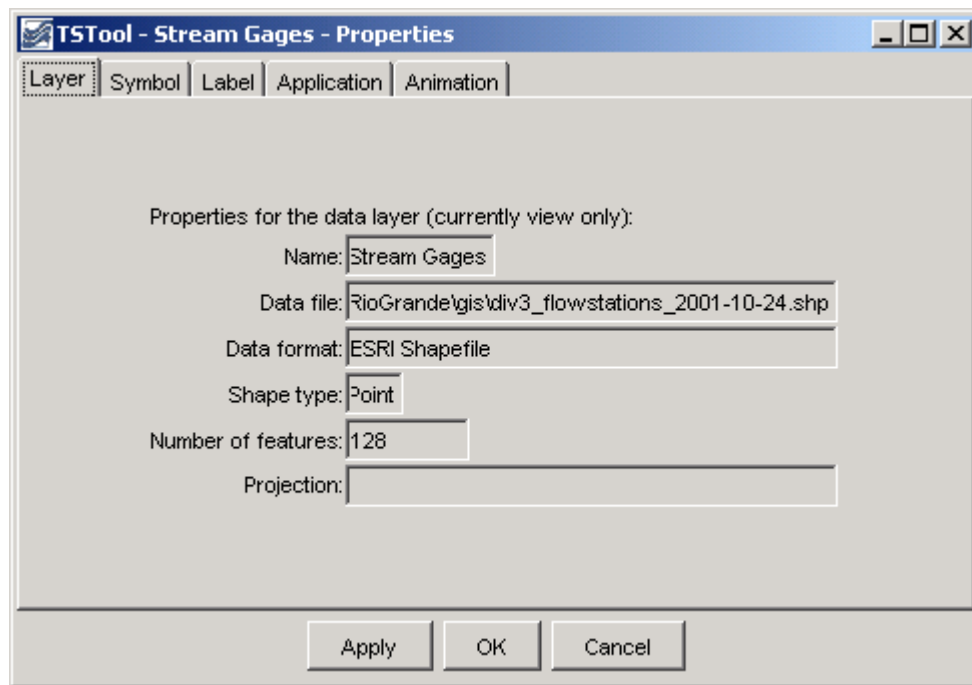
GeoView_Props

Main GeoView Properties

GeoView properties, as shown in the above figure, apply to the main GeoView and are shared between layers. These properties are typically not edited by end users. One important property is the projection property. If all data layers are projected consistently (e.g., for ESRI shapefiles) then a projection does not need to be defined. However, if the layers have different projects, a GeoView projection and projections for each layer can be defined to allow the GeoView to project data consistently for visualization.

If the **OK** or **Apply** buttons are pressed, the GeoView properties will be updated in memory (the GeoView Project file is not updated) and the map will redraw. Pressing **OK** will additionally close the properties dialog. The **Cancel** button causes the dialog to close, without updating the properties.

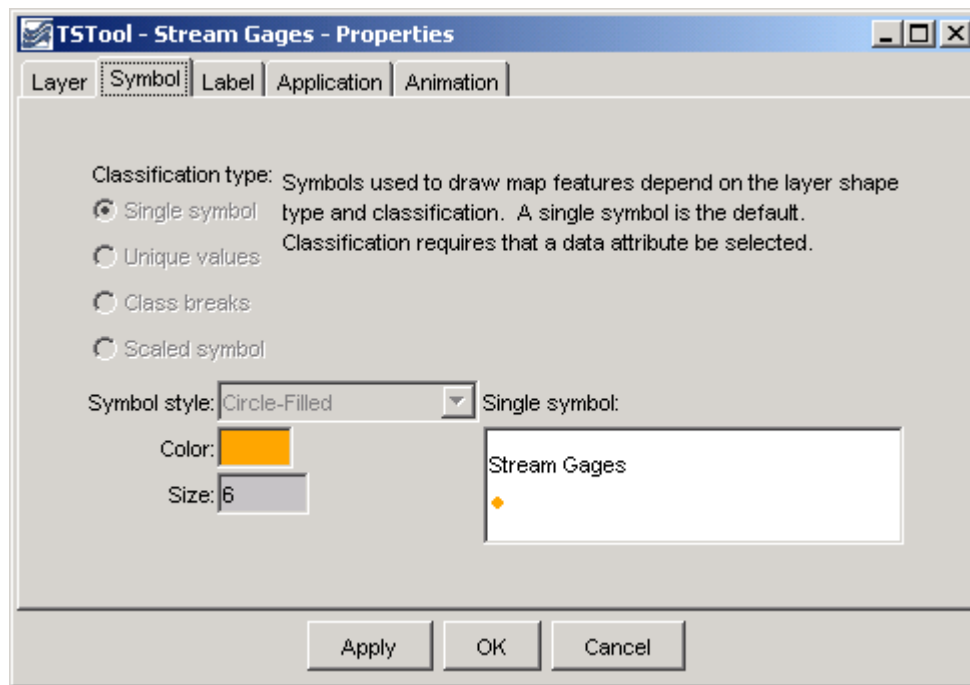
To view or change properties for a layer, select a layer view in the table of contents, right-click and select the **Properties** menu item. The following tabbed dialog will be displayed for the first selected layer view. The tabbed panels are discussed below the each figure.



GeoView_Props_Layer

GeoView Layer Properties

GeoView layer properties, as shown in the above figure, apply to the input source. Currently these properties are used for information purposes and cannot be interactively edited.

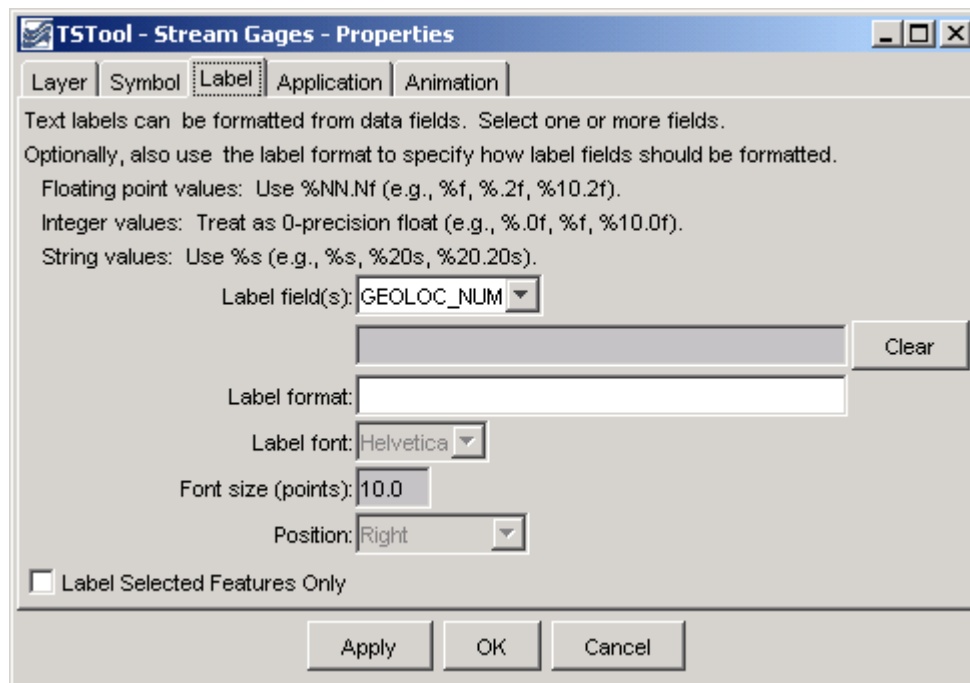


GeoView_Props_Symbol

Layer View Symbol Properties

Symbol properties, as shown in the above figure, indicate how the layer is to be drawn (symbolized) on the map and in the table of contents. A sample of the symbol is shown in the dialog, although it may appear slightly different on the map and table of contents.

Symbol terminology corresponds to standard GIS tools.



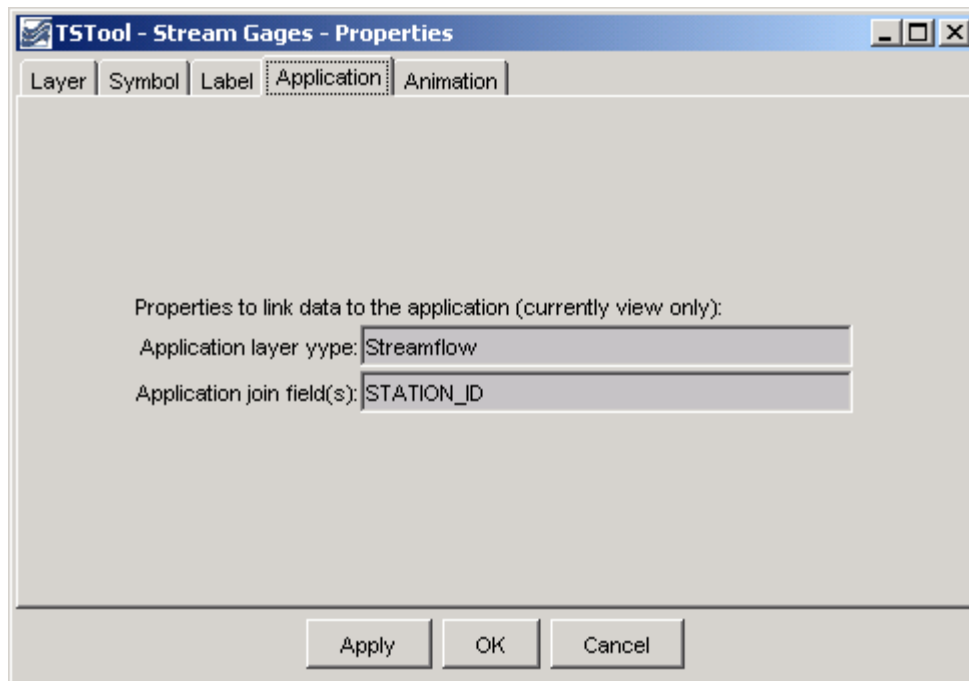
GeoView_Props_Label

Layer View Label Properties

Label properties, as shown in the above figure, can be modified to label features with attribute data or literal strings. Currently, only point features can be labeled. Labels can consist of a combination of attribute values. To label features, select the attribute fields from the available choices, in the order that they should appear in the label.

The label format, if not specified, defaults to the use the full field with of the attribute. For example, if an attribute field is defined as being twenty characters wide, the label may be the full width, including leading and trailing spaces. More often, it is desirable to omit the spaces. To do so, or to format numbers using a more appropriate format than the full. width default, use the **Label Format** information. The dialog box notes illustrate valid formats. For example, if a string field and an integer field are available, the following label format would show the labels with only a comma and one space between the values:

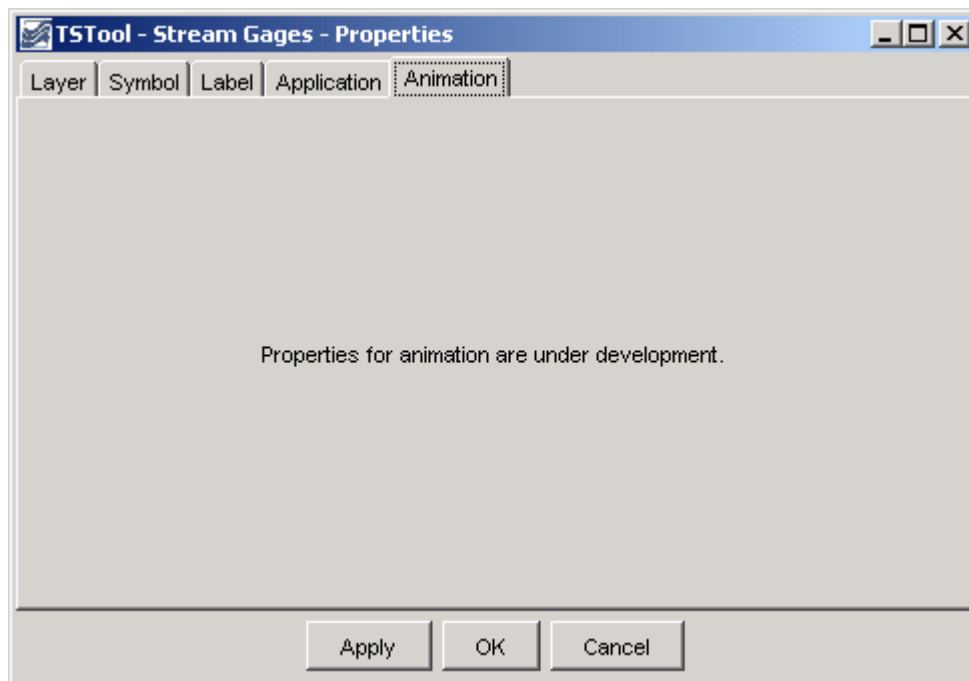
%s, %d



GeoView_Props_Application

Application Layer Type

Layer application properties (above) are used to link a layer's data to an application. This process allows general GeoView features to be used more specifically by specific software programs. The **Using GeoView with a Software Application** section (see below) describes this functionality in more detail.



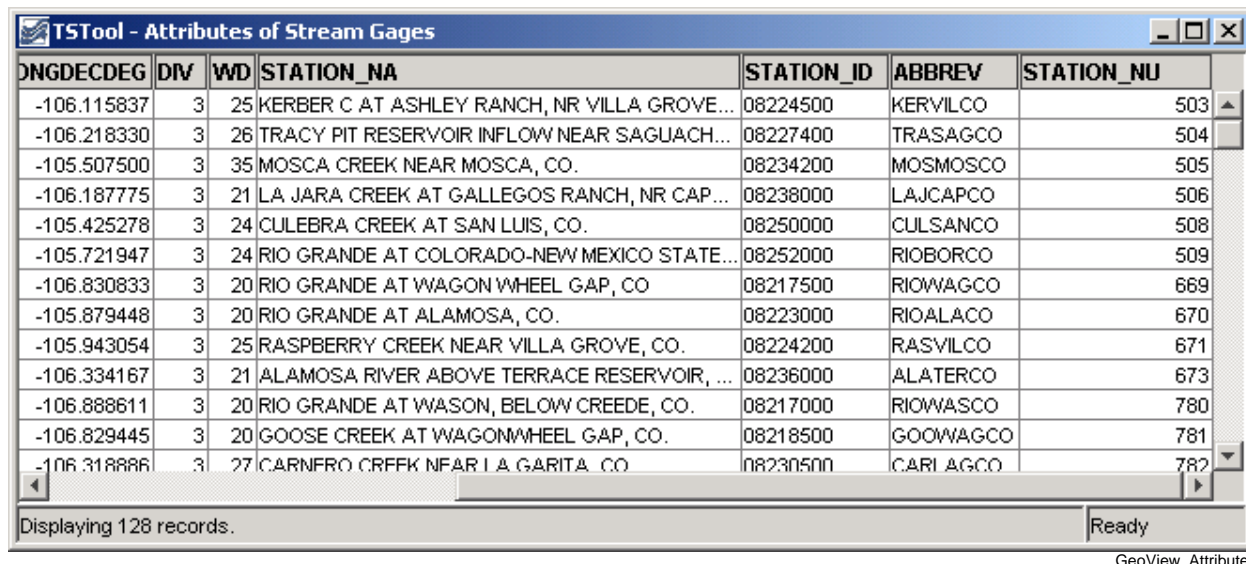
GeoView_Props_Animation

Layer View Animation Properties

Layer view animation properties are currently under development. Animation properties will define, for example, the time series data that are used for symbolization during animation.

Viewing a Layer's Attributes

Each feature in a layer includes geographic shape information (e.g., the coordinates that define a polygon). Each feature also can have attribute data, which are typically represented in a tabular fashion. To view the attributes for a layer, first select the layer in the table of contents, then right-click and press the **View Attribute Table** menu choice. A window similar to the following will be shown:



The screenshot shows a window titled "TSTool - Attributes of Stream Gages". It contains a table with the following columns: LONGDECDEG, DIV, WD, STATION_NA, STATION_ID, ABBREV, and STATION_NU. The table displays 128 records, with the first 12 records visible. The status bar at the bottom indicates "Displaying 128 records." and the window is in "Ready" state.

LONGDECDEG	DIV	WD	STATION_NA	STATION_ID	ABBREV	STATION_NU
-106.115837	3	25	KERBER C AT ASHLEY RANCH, NR VILLA GROVE...	08224500	KERVILCO	503
-106.218330	3	26	TRACY PIT RESERVOIR INFLOW NEAR SAGUACH...	08227400	TRASAGCO	504
-105.507500	3	35	MOSCA CREEK NEAR MOSCA, CO.	08234200	MOSMOSCO	505
-106.187775	3	21	LA JARA CREEK AT GALLEGOS RANCH, NR CAP...	08238000	LAJCAPCO	506
-105.425278	3	24	CULEBRA CREEK AT SAN LUIS, CO.	08250000	CULSANCO	508
-105.721947	3	24	RIO GRANDE AT COLORADO-NEW MEXICO STATE...	08252000	RIOBORCO	509
-106.830833	3	20	RIO GRANDE AT WAGON WHEEL GAP, CO	08217500	RIOWAGCO	669
-105.879448	3	20	RIO GRANDE AT ALAMOSA, CO.	08223000	RIOALACO	670
-105.943054	3	25	RASPBERRY CREEK NEAR VILLA GROVE, CO.	08224200	RASVILCO	671
-106.334167	3	21	ALAMOSA RIVER ABOVE TERRACE RESERVOIR, ...	08236000	ALATERCO	673
-106.888611	3	20	RIO GRANDE AT WASON, BELOW CREEDE, CO.	08217000	RIOWASCO	780
-106.829445	3	20	GOOSE CREEK AT WAGONWHEEL GAP, CO.	08218500	GOOWAGCO	781
-106.318886	3	27	CARNERO CREEK NEAR LA GARITA, CO	08230500	CARLAGCO	782

Attributes Table for a Layer

The attributes are displayed in the order and format determined from the input data. Attribute names in ESRI shapefiles are limited to ten characters. Information in the table can be selected (use **Ctrl-A** to select all) and can be copied to other software.

Using GeoView with a Software Application

Software developers integrate the GeoView components with software applications and typically the software user does not need to know how GeoView works with the application. However, this section describes a few important concepts that will help facilitate setting up data for use by an application.

Basic GeoView implementations involve defining a GeoView Project (see the **GeoView Configuration – the GeoView Project File** section below for details on project file properties) and then interacting with the GeoView interface when the map is displayed. In a basic application, a GeoView can be added to show maps for reference purposes only. For example, the application may be an interface to a database containing location information. If a GeoView project file is defined with only base layers, then the zooming and features will allow a user to zoom into a region of interest, but there will be no interaction between the GeoView and the application.

In a more advanced application, layers in the GeoView Project file are assigned an AppLayerType property, which is recognized by the application. For example, a layer may be assigned an application type of "Streamflow" to indicate a streamflow gage. Additionally, the AppJoinField property can be defined to allow the application to join its data to the geographic data. This assignment in and of itself causes no effect in the GeoView. However, the application can now interact with the GeoView by asking for the "Streamflow" layer. This allows features in the GeoView to be selected from the application (e.g., in a database query screen) and allows the GeoView to provide information about the layer to the

application. For this type of implementation, it is important that the application layer types, feature (shape) type, and the required join fields are documented; consequently, new data layers can be used with the application with only a few configuration changes.

Some applications may automatically update the map interface by zooming to selected areas, selecting features, etc. Standard GeoView features are typically still available, as previously described.

Limitations

The GeoView components have been developed not to serve as full-featured GIS components, but to support many common GIS activities like selection, zooming, and symbolization. The components have been developed to integrate with existing applications and use other tool sets, including time series viewing tools. Basic features have been implemented to address important needs for applications; however, additional features are implemented as requirements change. The GeoView components are not envisioned as a replacement for pure GIS tools like ESRI's ArcGIS products. In many cases, ESRI or other tools can be used to develop the data for use with GeoView.

Currently, properties that are changed interactively cannot be saved to the GeoView Project file.

GeoView software currently does not examine projection files optionally distributed with ESRI shapefiles. Projections must be defined in the project file (or, if omitted, the projection is assumed to be consistent for data layers). Only a few projections are recognized, as needed by specific GeoView software implementations.

GeoView Configuration – the GeoView Project File

A GeoView display is configured mainly by using a GeoView Project (.gvp) file, which is either read at software startup or when selected by the user. The purpose of the file is to persistently store the configuration of a map display so that it can be loaded again without redefining the configuration. The file format is simple text properties and can be read by applications implemented in various technologies running in various environments. An example of the file is shown below.

```
# Main properties global to the GeoView
# The format is:
# [Prefix]
# Prop=value
[GeoView]
GeoDataHome = .

# Properties for each GeoLayerView (data source and
# symbols)...
[GeoLayerView 1]
GeoLayer = xxx.shp

[GeoLayerView 2]
GeoLayer = xxx.shp
```

The GeoLayerViews listed first in the project file are drawn first and are therefore behind other layers on the map. For all properties, the comma is used as an internal delimiter and the semi-colon is used as a second layer of separation, as appropriate. Most properties will default to appropriate values if not specified (see tables below). The most important properties, as shown in the example above, are the GeoDataHome, which defines where data can be found, and GeoLayer, which defines where the data file is for each layer. Recognized layer file formats are listed in the following table and are described further in separate appendices. Support for additional layer types can be added as necessary.

Layer Type	Description
ESRI shapefile	ESRI shapefiles are commonly used with ESRI software such as ArcView, ArcMap, and ArcExplorer. GeoView determines the type by looking for the .shp file extension and checking the internal file format. No projection is assumed but the Projection property for the GeoView and individual layers can be used to indicate the projection.
NWSRFS GeoData files	The National Weather Service River Forecast System (NWSRFS) uses ASCII and binary files defining various geographic layers. This format is detected by checking the file names, which are predefined for NWSRFS. The Projection property is defined as Geographic if ASCII data and HRAP if binary data.
NWS XMRG Radar Files	XMRG files are gridded radar files produced by the National Weather Service. GeoView treats these files as grid files. This format is detected by looking for an “xmrg” string in the filename. The Projection property is defined as HRAP.

The following main GeoView properties can be defined in the project file. Graphical user interfaces to allow interactive editing of all properties are being implemented.

Main GeoView Property	Description	Default Value
Color	Background color for map. See the discussion after the properties tables for a discussion of how to define colors.	White.
FontName	Name of font to use for GeoView components (e.g., buttons). This property currently can only be set internally with software.	System-specific.
FontSize	Size of font, in points, to use for GeoView components (e.g., buttons). This property currently can only be set internally with software.	System-specific.
FontStyle	Font style to use for GeoView components (e.g., buttons). This property currently can only be set internally with software.	System-specific.
GeoDataHome	Directory where the GIS data exist. This directory will be prepended to layer files if they are not absolute paths already.	If not specified or if specified as ".", the directory will be set as the home of the GeoView Project file.
InitialExtent	Initial extent of the map display, in data coordinates. The coordinates should be specified as "XMIN,YMIN XMAX, YMAX", where the first pair is the lower-left corner of the extents and the second pair is the upper right. This property has not been implemented. See the MaximumExtent property.	No default. The initial extent will be the maximum data extent.
MaximumExtent	Maximum extent of the map display when zoomed out, in data coordinates. The coordinates should be specified as "XMIN,YMIN XMAX,YMAX", where the first pair is the lower-left corner of the extents and the second pair is the upper right.	No default. The maximum extent will be the maximum data extent.
Projection	Projection for the GeoView. The projection definition varies depending on the projection (some projections require more parameters). The following projections are currently supported: Geographic - no projection (decimal degrees) HRAP - used by National Weather Service UTM,Zone[,Datum,FalseEasting] [,FalseNorthing][,CentralLongitude] [,OriginLatitude][,Scale] - Universal Transverse Mercator. The Zone is required (e.g., 13 for Colorado). Datum defaults to NAD83. The FalseEasting defaults to 500000. The FalseNorthing defaults to 0. The CentralLongitude is computed from the Zone. The OriginLatitude defaults to zero. The Scale defaults to .9996.	No default. All data are assumed to be the same projection.
ProjectAtRead	Indicates whether layer features are projected at read-time to the GeoView projection. This slows down the application initially but increases performance later during map refreshes.	false (it is usually best to project all data to a common projection rather than relying on GeoView to do projections)
SelectColor	Color to use for selected features. See the discussion after this table for examples of how to specify colors.	Yellow A more unique magenta/pink color with RGB 255,120,255 is being considered.

The following GeoLayerView properties can be defined, corresponding to each data layer/file:

GeoLayer View Property	Description	Default Value
AppJoinField	Specify the field(s) that should be used by an application to join the layer data to application data. If multiple fields are necessary, separate the field names by commas (e.g., "wd, id").	None.
AppLayerType	Indicate a layer type to be handled by an application. For example, a layer may be tagged as "Streamflow". The application can then use this information to treat the layer differently (e.g., to know how to join the data to application data). Valid AppLayerType values must be defined and understood by the application.	None.
Color	Color for features when the SymbolClassification is SingleSymbol. If point data, this is the main color for the symbol. If line data, this is the line color. If polygon data, this is the fill color. See the discussion after this table for examples of how to specify colors.	Random.
ColorTable	Used when the SymbolClassification property is ClassBreaks or UniqueValues and requires more than a single color. The number of colors should be one more than the number of class break values if SymbolClassBreaks is used and equal the number of class values if UniqueValues is used. Color tables can be defined in three ways: <ol style="list-style-type: none"> ColorTableName; NumColors Predefined tables include Gray, BlueToCyan, BlueToMagenta, BlueToRed, CyanToYellow, MagentaToCyan, MagentaToRed, YellowToMagenta, and YellowToRed. These named tables choose primary colors where necessary to provide clean color breaks. Ramp; NumColors; Color1; Color2 Custom; NumColors; Color1; ...; ColorN Only the first option is currently enabled. See the discussion after this table for examples of how to specify colors.	Named color table using Gray.
EventLayerView	Indicates if the layer view is an event layer (ESRI Map Object notation). This property is not currently used in the Java tools.	false
GeoLayer	Name of file for the data layer, typically with the file extension. If an ESRI shapefile, specify the .shp file. If a relative path, the GeoView.GeoDataHome property will be prepended to the file name. This property is used to detect a break in the GeoLayerView numbering, indicating the end of layer views.	No default. Should always be specified.
IgnoreDataOutside	Indicate a range of values that should be considered. Currently this applies only to grid layer types. Specifying a range can be used, for example, to draw only cells with positive data values. The range should be specified as two numbers separated by a comma (e.g., -00001, 100.0).	Not specified. All cells are considered.
LabelField	Specify one or more fields to be used for the label, separated by commas. If a LabelFormat property is specified, use it to format the label; otherwise, format each field according to the field specifications from the attribute data source.	No default. Specify one or more fields for the label.

GeoLayer View	Description	Default Value
LabelFontName	Font to use for labels. This property has not been enabled.	Helvetica
LabelFontSize	Label font height, points. This property has not been enabled.	10
LabelFormat	Specify a C-style format string to format the fields. The format specifications must agree with the data types being formatted. For example, if two floating-point fields are specified with the <code>LabelField</code> property, the corresponding format may be <code>"%10.1f, %5.2f"</code> .	If not specified, the label will be formatted using the field width and precision determined from the data table, with values separated by commas.
LabelPosition	Label position. If point data, the position is relative to the point coordinates. If line or polygon data, the position is relative to the centroid coordinates. The position of the text will be offset to not overwrite a symbol and can be <code>UpperRight</code> , <code>Right</code> , <code>LowerRight</code> , <code>Below</code> , <code>LowerLeft</code> , <code>Left</code> , <code>UpperLeft</code> , or <code>Above</code> .	Right
Name	The layer view name that should be displayed in the legend.	No default. If not specified, the file name will be used in the legend.
OutlineColor	Outline color for point or polygon symbols. See the discussion after this table for examples of how to specify colors.	Default to the same as main color).
Projection	Projection for the layer's data. See the main <code>GeoView</code> <code>Projection</code> property for available values.	No default. It is assumed that all data in a project have a consistent projection.
ProjectAtRead	Indicates whether features are projected at read-time to the <code>GeoView</code> projection. This slows down the application initially but increases performance later during map refreshes. This property can be set once in the <code>GeoView</code> main properties.	<code>false</code> (it is usually best to project all data to a common projection rather than relying on <code>GeoView</code> to do projections)
ReadAttributes	Indicates whether attributes should be read when the data are read. If possible, based on the layer data format, attributes will be read on the fly as needed. Reading the attributes (<code>true</code>) takes more memory but will result in faster performance.	<code>false</code>
ReferenceLayer	Indicates whether the layer should be drawn in the reference <code>GeoView</code> . Indicate as <code>true</code> or <code>false</code> . Typically only the most general boundary information should be used in the reference layer.	<code>false</code>
SelectColor	Specify the color to be used when drawing selected features. This property is useful if the default select color does is not easily viewed.	Use the <code>GeoView</code> . <code>SelectColor</code> property.
SkipLayerView	Can be set to <code>true</code> to skip the layer altogether when reading the project file (useful for commenting out layers during development). If this property is used, the number sequence for the layer views can be kept the same.	<code>false</code> (the layer view will be displayed)
SymbolClassBreaks	Class breaks that correspond to the <code>ClassBreaks</code> <code>SymbolClassification</code> property. The number of values should be one less than the number of values in the <code>ColorTable</code> property for the classification.	No default, although an application may suggest values.
SymbolClassField	Attribute data field that is used when the <code>SymbolClassification</code> property is <code>ClassBreaks</code> or <code>UniqueValues</code> .	No default, although an application may suggest a value based on the available attributes.

GeoLayer View	Description	Default Value
SymbolClassification	Indicates how data are to be classified when displaying the shape symbols. Values can be <code>SingleSymbol</code> (e.g., single point symbol, line style, or polygon fill color), <code>UniqueValues</code> (display a unique symbol style for each value, currently not implemented), or <code>ClassBreaks</code> (display a unique symbol style for groupings of values - requires specification of the <code>SymbolClassField</code> and <code>SymbolClassBreaks</code> properties).	<code>SingleSymbol</code>
SymbolSize	For point data, specify the symbol size in pixels. For line data specify the line width in pixels. Not used for polygon data. This property may need to be expanded to properly handle printed output (might need to use points rather than pixels or allow the units of measure to be set in the property). This property is currently not enabled.	6 pixels for points, 1 pixel for lines.
SymbolStyle	Indicates the symbol style. If point symbols, the style is the symbol identifier (e.g., <code>CircleFilled</code>). If line data, the symbol style is the line style (currently only <code>Solid</code> is supported). If polygon data, the symbol style is the fill pattern (currently only <code>FillSolid</code> is supported). See below for a full discussion of symbol styles.	None for points, <code>Solid</code> for lines, <code>FillSolid</code> for polygons.

Color Specification

Colors are specified for a number of different properties, including the feature color and outline color. In order to allow flexibility in specifying colors, a number of formats are supported:

- Named color. Available colors are: None (transparent), Black, Blue, Cyan, DarkGray, Gray, Green, LightGray, Magenta, Orange, Pink, Red, White, Yellow
- Comma-separated Color Triplets as 0-255 (e.g., 255, 0, 0) or 0.0 -1.0 (e.g., 1.0, 0.0, 0.0).
- Hexadecimal: 0xRRGGBB (e.g., 0xFF0000 for red)

Color Tables

Color tables are simply a list of colors. Typically the symbol maintains a color table if the classification is other than `SingleSymbol`. The symbol will also keep track of unique values or class breaks and use this information to determine a color to display for a shape. A number of predefined color tables are supported but and user-defined tables is supported in the property format.

Symbol Style - Point Data

Symbol styles for point data are the same as for time series viewing tools. The following styles are available:

- None
- Arrow-Down, Arrow-Left
- Asterisk
- Circle-Hollow, Circle-Filled
- Diamond-Hollow, Diamond-Filled
- Plus, Plus-Square
- Square-Hollow, Square-Filled

- Triangle-Down-Hollow, Triangle-Down-Filled, Triangle-Left-Hollow, Triangle-Left-Filled, Triangle-Right-Hollow, Triangle-Right-Filled, Triangle-Up-Hollow, Triangle-Up-Filled
- X-Cap, X-Diamond, X-Edge, X-Square

Classification

Classification is used to symbolize data. The following classifications are supported:

Classification Type	Description
SingleSymbol	This is the default for all layers if not specified. For point data, a single symbol is used, centered on the . For line data, a single line width and color is used. For polygon data, single fill and outline colors are used.
UniqueValues	The data values for the field specified with the SymbolClassField property is sorted and checked for unique values. Each value is then assigned a color in the color table.
ClassBreaks	<p>The number of class breaks should be one less than number of colors in the color table for the symbol. Breaks are defined by using a groupings of features based on the values of the field specified with the SymbolClassField property:</p> <p>< first value >= first value < second value ... > last value</p>

GeoView Project File Examples

This section provides several examples, extracted from GeoView Project files.

The following example illustrates how to configure base layers in a GeoView Project file:

```
# GeoView project file for Rio Grande basin.

# Main GeoView properties.

[GeoView]

# Main home for data
# If a directory is not specified, the directory will be determined when the
# GeoView project file is selected.
#GeoDataHome = "C:\cdss\statemod\data\rgtwday\gis"
# ArcView/ArcExplorer Default...
#SelectColor = Yellow
# Arc 8...
#SelectColor = Cyan
# All-purpose (magenta/pink)
SelectColor = "255,120,255"
MaximumExtent = "266400,4090475 503060,4260700"

# Now list the layer views. A layer view consists of specifying a data layer
# (e.g., shapefile) and view information (e.g., symbol). This is equivalent to
# the ESRI "theme" concept. The layers specified first are drawn on the bottom.
# Start with number 1 and increase the layer number sequentially as layers are
# added on top.

[GeoLayerView 1]
GeoLayer = div3_districts.shp
Name = "Water Districts"
# RGB 153 204 50 - green-yellow
#Color = "0x99CC32"
# tan
Color = "255,240,190"
OutlineColor = black
ReferenceLayer = true
AppLayerType = "BaseLayer"

[GeoLayerView 2]
GeoLayer = div3_lakes.shp
#GeoLayer = div3_lakes.shp
Name = "Lakes"
# - blue
Color = "165,250,254"
OutlineColor = "0,130,254"
AppLayerType = "BaseLayer"

[GeoLayerView 3]
Name = "Rivers"
GeoLayer = div3_rivers.shp
#GeoLayer = div3_rivers.shp
# RGB - blue
Color = "0,188,253"
AppLayerType = "BaseLayer"

[GeoLayerView 4]
GeoLayer = div3_highways.shp
Name = "Roads and Highways"
Color = "255,0,0"
AppLayerType = "BaseLayer"

[GeoLayerView 5]
GeoLayer = div3_cities.shp
Name = "Cities and Towns"
SymbolStyle = "Square-Filled"
SymbolSize = 6
Color = "red"
LabelField = "Name"
LabelPosition = RightCenter
AppLayerType = "BaseLayer"
```


The following example illustrates how to display point data layers. These properties should be inserted at the appropriate location in a GeoView Project file.

```
[GeoLayerView 15]
#SkipLayerView = true
GeoLayer = div3_flowstations_2001-10-24.shp
Name = "Stream Gages"
# orange
Color = "254,167,0"
SymbolStyle = "Circle-Filled"
SymbolSize = 6
AppLayerType = "Streamflow"
AppJoinField = "STATION_ID"
#LabelField = "STATION_NA, STATION_NA"
#LabelFormat = "%s, %s"

[GeoLayerView 18]
#SkipLayerView = true
GeoLayer = div3_reservoirs_2001-10-24.shp
Name = "Reservoirs (WDID)"
# black
Color = "black"
SymbolStyle = "Triangle-Up-Filled"
SymbolSize = 6
AppLayerType = "Reservoir"
AppJoinField = "ID_LABEL_6"
```

This page is intentionally blank.

Appendix: Spatial Data Format – ESRI Shapefile

2004-05-24, Acrobat Distiller

Overview

ESRI Shapefiles are a relatively simple format for spatial data, consisting of a file containing shape information (*.shp*), a file containing attribute data (*.dbf*), and an index file (*.shx*). The GeoView package currently supports the following shapefile shape types:

- Point (shape type 1)
- Multi-point (shape type 8)
- Arc/Line (shape type 3)
- Polygon type (shape type 5)
- Null shape (shape type 0)

Support for additional shape types may be added in the future, consistent with the shapefile specifications.

This page is intentionally blank.

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

