2 Introduction

Version 11.06.00, 2015-07-23

TSTool can be thought of as a time series calculator, and additionally TSTool processes data tables and to a limited degree spatial data. TSTool reads, displays, manipulates, analyzes, and writes time series data, either interactively or in batch (automated) mode. The TSTool graphical user interface (GUI) 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 volumes:

Volume 1 – User Manual (for overall information about understanding and using TSTool)
Volume 2 – Command Reference (detailed documentation for every command)
Volume 3 – Datastore Reference (detailed documentation for every datastore, including file formats, databases, and web services)

This **User Manual** documentation is divided into the following chapters:

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.

Chapter 10 – Spreadsheet Integration provides information about using TSTool with spreadsheet software.

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

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

The **TSView Time Series Viewing Tools** appendix provides a general reference for time series viewing features. These features are used throughout TSTool.

The **GeoView Mapping Tools** appendix provides a general reference for the GeoView map interface. The mapping interface is being phased in.

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

2.1 TSTool Data Flow Concepts

Although TSTool can be used simply to browse and view data, the main function of TSTool is to automate data processing using a workflow of commands. Figure 1 illustrates overall data flow concepts.

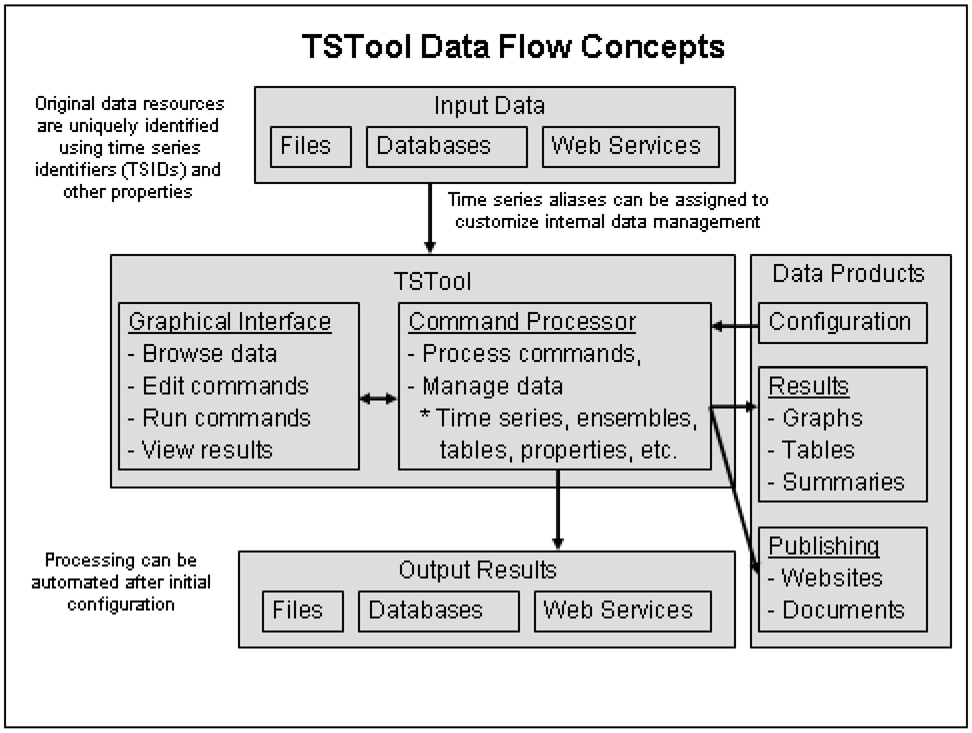

TSTool Data Flow Concepts
Original data resources are uniquely identified using time series identifiers (TSIDs) and other properties
Input Data
Files
Databases
Web Services
Time series aliases can be assigned to customize internal data management
TSTool
Graphical Interface
- Browse data
- Edit commands
- Run commands
- View results
Command Processor
- Process commands,
- Manage data
* Time series, ensembles, tables, properties, etc.
Data Products
Configuration
Results
- Graphs
- Tables
- Summaries
Publishing
- Websites
- Documents
Processing can be automated after initial configuration
Output Results
Files
Databases
Web Services

Figure 1 – TSTool Data Flow Concepts

Input data are retrieved using unique identifiers, such as time series identifiers discussed in the next section, and table identifiers. The TSTool graphical user interface (GUI) is used to browse data, edit commands, run commands (by calling the command processor), and view results. The command processor internally manages data including lists of time series, time series ensembles, tables, and processor properties, and provides access to the data in interactions with the GUI. The command processor also can output results to files, databases, web services, and can create data products. Once configured, processing can be automated.

The interpretation of input data and definition of workflow and data products depends heavily on standards and conventions, some of which are defined by data providers, and some of which are defined by the TSTool design. Although standards define specific features, TSTool allows users a great deal of flexibility in defining workflows and producing data products.

2.2 Time Series Objects and Identifiers

TSTool handles time series as objects that are read (or internally created), manipulated, viewed, and output. Time series data include properties (also referred to as attributes or metadata) and a series of date/time and data value pairs (and optionally value data flags). Data generally consist of floating point values; however, time series may contain other data such as strings or images. TSTool primarily focuses on numerical time series. Time series are defined either as regular interval (equal spacing of date/time) or irregular interval (e.g., infrequent measurements). 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 properties 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 `NaN` [Not a Number])
- Comments (often station comments, if available)
- Processing history (a list of comments about how the time series was created and manipulated)

To identify time series in the original data and manage time series internally, TSTool assigns each time series a time series identifier (TSID) that uses the notation:

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

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

The TSID can be thought of as a unique resource identifier, similar to a URL for web resources. The first five parts (`Location.Source.Type.Interval.Scenario`) are used to identify time series within the original data (e.g., to find the time series in a file or database):

- `LocationType` – optionally used where necessary to uniquely identify locations (e.g., use a location type of `Basin` or `Subbasin` where other identifier information would result in ambiguous interpretation of the identifier parts
- `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 a location changes over time or a database may contain time series from multiple data providers
- `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 sequence identifier used in cases where multiple time series traces in an ensemble 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). Where historical data are used as input for an analysis, the sequence identifier typically is a four-digit year corresponding to the data input start year.

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

The following table lists datastore types that are supported or are under development. See the datastore appendices for information about how time series identifiers are formatted for specific datastores. TSTool features for a specific datastore may only be available if at least one datastore for a specific type is configured and enabled. Datastores are available for databases and web services because the "connection" information can be configured once and used throughout the TSTool session. User-assigned datastore names are used in TSTool commands to ensure that command files are transportable and datastore configurations can be updated without requiring commands to be updated.

Datastores Supported by TSTool

Datastore Type	Category	Description
ColoradoWaterHBGuest	Web service	State of Colorado's historical data.
ColoradoWaterSMS	Web service	State of Colorado's real-time data.
Generic Database	Database	A generic datastore to work with the `ReadTableFromDataStore()`, `WriteTableToDataStore()`, `ReadTimeSeriesFromDataStore()`, and `WriteTimeSeriesToDataStore()` commands, which can be configured to use an Open Database Connectivity (ODBC) Data Source Name (DSN).
HydroBase	Database	State of Colorado database (see also legacy and default treatment as an input type in following table). This is not available by default due to the legacy practice of allowing HydroBase to be selected at TSTool startup.
NRCS AWDB	Web service	Natural Resources Conservation Service (NRCS) Air and Water Database.
RCC-ACIS	Web service	Regional Climate Center Applied Climate Information System.
ReclamationHDB	Database	United States Bureau of Reclamation HDB database.
RiversideDB	Database	Riverside Technology, inc. database used for real-time and historical time series data as part of RiverTrak® System software.
USGS NWIS Daily	Web service	United States Geological Survey (USGS) National Water Information System (NWIS) daily values.

Datastore Type	Category	Description
USGS NWIS Groundwater	Web service	USGS NWIS groundwater values.
USGS NWIS Instantaneous	Web service	USGS NWIS instantaneous (real-time) values.
WaterOneFlow	Web service	WaterOneFlow web services (uses WaterML format for time series) – under development.

The following table lists input types that are supported or are under development. The legacy "input type" terminology continues to be used but in the future may be replaced with "file datastore" or similar. See the input type appendices for information about how time series identifiers are formatted for specific input types. TSTool features for a specific input type may only be available if the input type is enabled in the TSTool configuration file. Input types may utilize an "input name", for example when a time series identifier needs to include the input type and the name of a file being read. Reading time series from a single file using this approach makes sense because there is no need to configure a datastore with a configuration file for every data file. Most legacy input types that could be migrated to datastores have been migrated.

Input Types Supported by TSTool

Input Type	Category	Description
DateValue	Single file	General delimited date/value file with extended header information, able to store one or more time series.
Delimited	Single file	Generic column-delimited format (see the `ReadDelimitedFile()` command).
DIADvisor	Database	DIADvisor real-time environmental monitoring software, from OneRain, Inc. Earlier versions of TSTool provided DIADvisor integration; however, this capability has not been actively maintained.
HEC-DSS	Binary file database	Army Corp of Engineers binary time series database file used with Hydrologic Engineering Center (HEC) software.
HydroBase	Database	State of Colorado database. HydroBase also can be accessed as a datastore (see previous table and HydroBase datastore appendix). The input type convention is being retained in the short term until a decision can be made about how to configure HydroBase datastores consistently between multiple users and projects.
MODSIM	Single file	Colorado State University MODSIM model, version 7. MODSIM version 8 uses a database but full support for this database has not been added (see Generic Database datastore as one option).
NWSCard	Single file	National Weather Service River Forecast System (NWSRFS) card file format for hourly data.
NWSRFS_ESPTraceEnsemble	Binary file	NWSRFS Ensemble Streamflow Prediction binary file.
NWSRFS_FS5Files	Binary file database	NWSRFS binary FS5Files preprocessor and processed database.

Input Type	Category	Description
RiverWare	Single file	University of Colorado Center for Advanced Decision Support for Water and Environmental Systems (CADSWES) RiverWare model data format.
SHEF	Single file	Standard Hydrologic Exchange Format, a common data format used by United States government agencies. Only limited support is provided.
StateCU	Single file	State of Colorado consumptive use model time series input and output file formats.
StateCUB	Binary file database	State of Colorado consumptive use model binary output file.
StateMod	Single file	State of Colorado StateMod model time series input and output file formats.
StateModB	Binary file database	State of Colorado StateMod model output binary file.
USGS NWIS Rdb	Single file	USGS NWIS RDB format files.
WaterML	Single file	WaterML format file.

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 daily streamflow file might be identified using:

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

In these examples, the optional scenario (fifth part) and sequence identifier are not used. TSID strings can be saved in a command file or time series product description file to indicate the time series to process. The TSID with input information allows TSTool to determine how to access the time series and is useful for managing time series. The TSTool GUI typically handles creation of all time series identifiers; however, command files can be edited with a text editor. The path to files can be absolute or relative to the command file. The latter is recommended to improve portability of files between computers.

TSTool only shows the input type and input name parts of the identifier when editing 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 (see the next section).

2.3 Using Time Series Aliases

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:

```
ReadTimeSeries(Alias=”09010500”,TSID=”09010500.USGS.Streamflow.Month~HydroBase”)
```

The following older syntax was phased out in TSTool version 10.00.00, and is automatically converted to the above syntax when a command file is read:

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

If an alias is defined for a time series, the alias will be used instead of the alias and will be displayed in command editors and results (although the TSID is often also shown in output). TSTool ignores upper/lower case when comparing identifiers, aliases, and other commands, although it is good practice to be consistent. The general term "TSID", when used as a command parameter to identify time series to process, allows a TSID or alias to be specified.

TSTool commands allow the alias to be defined using time series properties. For example, the parameter `Alias="%L"` will assign the alias as the location part of the time series identifier. A combination of the special formatting strings and literal characters can be used to create unique aliases for time series. It is recommended that aliases not contain spaces or other whitespace and that periods be avoided because they are used in TSIDs.

When defining a series of commands as a workflow to perform a task, a user generally will develop a rough draft of the process by using TSTool's interactive data browsing and command editing features. However, it is useful to develop a data flow description more identifiers that are more appropriate than the default TSIDs provided by TSTool. Consequently, at an appropriate time in the workflow definition, it is useful to take a step back and review the identifiers being used, and insert aliases where possible. For example, an inventory of time series may be defined to describe time series used in a process, similar to the following (where the bold indicates information that will be replaced with a specific value):

Alias	Description
Location-Streamflow-Month-Original	Original monthly streamflow.
Location-Streamflow-Month-QC	Monthly streamflow data after quality control.
Location-Streamflow-Month	Monthly streamflow data after filling data gaps, ready for analysis.
Location-Streamflow-Year	Annual streamflow data, accumulated from **Location-**Streamflow-Month.

By using aliases, it is possible to switch the data inputs with minor changes to the processing logic. If it is important to indicate the source of information, such as when comparing the same data stored in two different databases, then the alias can be defined to include the input name. Regardless of whether an alias is used, the original TSID also is maintained with the time series and is available for reports and displays.

In summary, if an alias is assigned to a time series, it will take precedence over the TSID when identifying the time series.

2.4 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.
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`, `Hour`). Consequently `24Hour` data has a base interval of `Hour` and a multiplier of `24`.
3. The period of a time series is defined by start and end date/time values, using 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) or water year (October to November). Additionally, a year type of NovToOct has been implemented to represent November to October and additional similar year types may be implemented in the future. Calendar year is the default but can be changed in some commands.

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 but generally are not used (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 formats commonly used in the United States and other locales. 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 input type and datastore appendices discuss date/time issues with various data 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 screen pixels or page coordinates. 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. Some commands will allow an automatic conversion of 24Hour interval data to Day interval data to avoid hour offset issues.

TSTool understands leap years and days per month. Consequently data formats that do not properly implement leap years or simplify time by assuming a constant number of days per month may result in missing values in data when read into TSTool.

TSTool does have the capability to handle time zone in small interval (hour, minute) data. However, fully representing time zone and daylight savings offsets is somewhat complex and TSTool for the most part does not perform time zone conversions or normalization.

Input formats that have different conventions are handled by converting the data to TSTool conventions when reading the data and converting from TSTool conventions when writing the data.

2.5 Time Scale for Time Series Data

The time scale for time series data gives an indication of how data values 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 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.

Input formats that have different conventions are handled by converting the data to TSTool conventions when reading the data and converting from TSTool conventions when writing the data.

2.6 Time Series Commands and Processing Sequence

Although TSTool can be run in batch mode (see **Chapter 3 – Getting Started**), it is possible 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.

TSTool commands fall into several categories:

1. Time series identifiers (see Section **2.2 – Time Series Objects and Identifiers**), which are equivalent to time series "read" commands (where the identifier input type or datastore is used to determine how to read from the original data format)
2. General commands, which are used to set properties like the period for output, and control processing such as `If()` and `For()` commands
3. Time series commands, which are used to read and manipulate time series and output results
4. Ensemble commands, which process ensembles of time series,
5. Table commands, which process tables of information.

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. Command syntax is 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, which makes the command more readable as text.
- Because the parameter name is included, the parameters can be in any order. The command editor dialogs will enforce a default older.
- 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.

Older commands used a fixed notation where parameters were required to be in a specific order. TSTool will attempt to update older command syntax to the current syntax when a command file is read. The **Command Reference** describes each command's function and parameters.

The following sequence occurs when processing commands:

1. **Check the command for run-time initialization errors** . Commands initially are parsed when a command file is opened or new commands are added in the GUI. When commands are run, additional checks are performed based on run-time data, such as the period that has been specified for the run. Example commands are shown below. If the command results in reading or creating a time series, step 3 is executed, as described below.

```
# Example commands
08235350.USGS.Streamflow.Month~HydroBase
```

```
08236000.DWR.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 or create time series.** TSTool recognizes that certain commands should read or create 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. By default, the entire time series will be read unless the `SetInputPeriod()` or `SetOutputPeriod()` commands have been specified.
    a. **Read data.** If the input type, and if needed, input name, are specified in the identifier, they are used to read the data. Time series properties such as units are assigned.
    b. **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.
    c. **Save time series in processor.** The time series that are read or created are managed by the processor, using the TSID and alias. The time series can then be manipulated by other commands, as per the following step.

3. **Manipulate time series or other data.** 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 using the TSID or alias and are manipulated as per the command functionality. 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.DWR.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 time series (those that have been read or created 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).

TSTool commands generally use the `TSList` parameter to locate time series in memory, where the default value depends on the command. For example, `TSList=AllTS` will process all time series. For commands that require a single input time series, TSTool often will default to `TSList=LastMatchingTSID`, which finds the first time series prior to the current command. It is possible to create more than one time series with the same identifier while allowing localized processing of each time series. For 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"). However, this may lead to confusion and generally should be avoided by using unique identifiers for each time series that is being processed.

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. Output commands also may use the `TSList` parameter to indicate which time series are to be output.

4. 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 also can 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 is not displayed. Processing the example shown above results in three time series in memory:

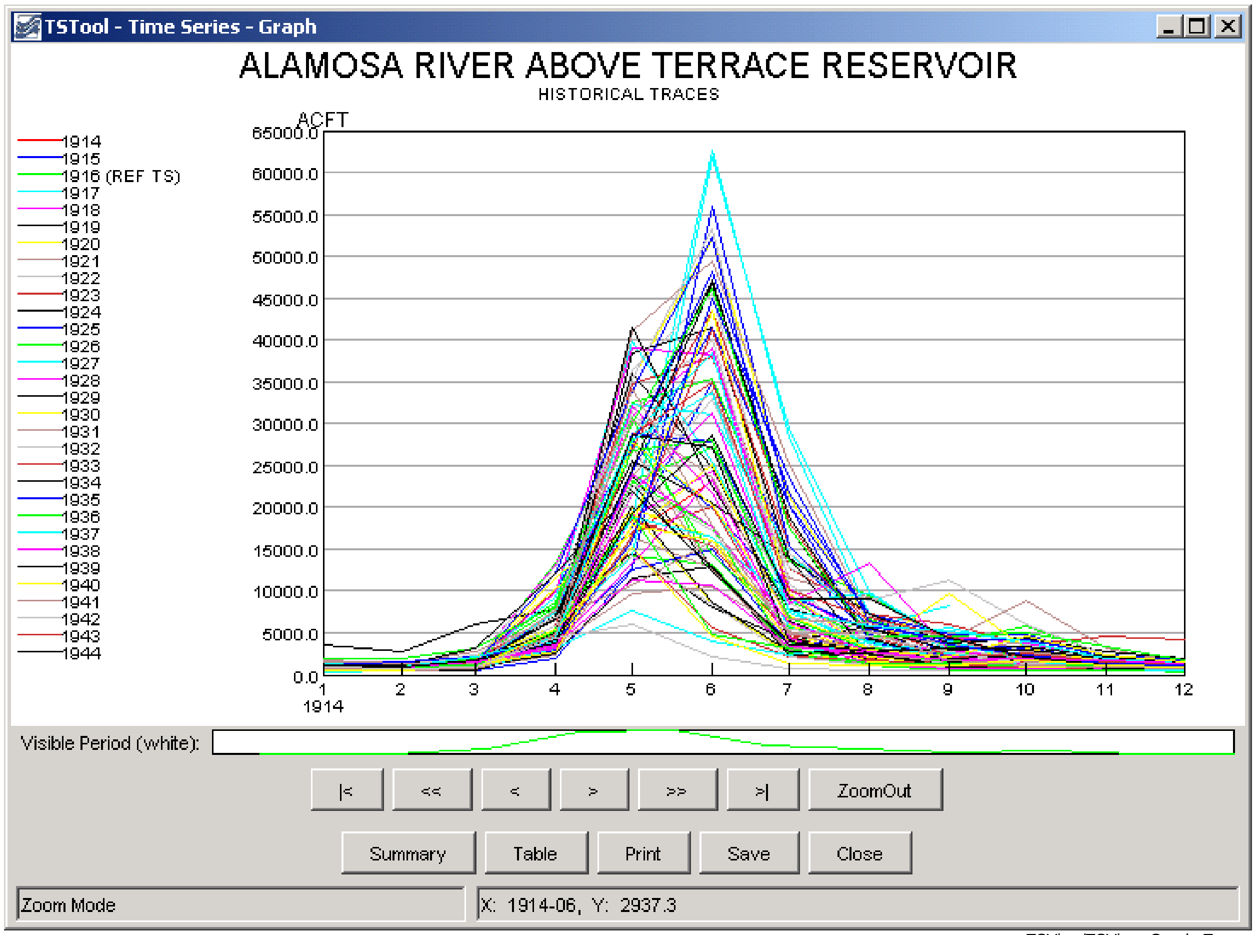
1. A time series identified by `08235350.USGS.Streamflow.Month`, containing the sum of the two time series.
2. A time series identified by `08236000.DWR.Streamflow.Month`, containing the input to the `Add()` command.
3. A time series **also** identified by `08235350.USGS.Streamflow.Month`, containing 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.

2.7 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 the ***Ensembles*** results tab. Each time series in an ensemble is also available for individual processing and is listed in the ***Time Series*** results tab. Time series in ensembles often have a `[Seq]` sequence identifier in the TSID corresponding to the starting year of the ensemble trace time series, or other unique identifier, as discussed in **Section 2.2**.

The following figure illustrates an ensemble of annual time series created from a long historical time series.



TSView/TSView_Graph_Traces

Example Ensemble Plot Showing Historical Years

In general, ensembles provide a way to process a group of time series. Refer to the command reference for specific features and limitations related to ensemble processing.

2.8 Time Series Processor Properties

The time series processor performs data management tasks such as tracking the list of time series and providing requested time series to commands for processing (using the time series identifier and alias to match time series). The processor also has an understanding of standard global properties that serve as defaults for all commands, such as the input and output periods and working directory. In addition to global properties that are assigned by default, the `SetProperty()` command can be used to provide user-specified properties to the processor. For example, the user might want to define a property corresponding to a folder on the system for data files. The property then can be used in command parameters to insert the folder name. The syntax for using a property in a command parameter is:

```
Command(InputFile="${PropertyName}/SomeOtherInformation")
```

These processor properties should not be confused with time series properties, which are specific to each time series and are set with the `SetTimeSeriesProperty()` command. Command editors may be updated over time to provide drop-down choices of defined global properties; however, it may be necessary to enter the property name as text. The following table lists standard global properties that can be used in command parameters that recognize the global properties (refer to specific command documentation to determine whether a command supports global properties). The internal representation of each property varies depending on the meaning of the property. For example, date/times are represented internally as DateTime objects. Regardless of internal representation, the value is converted to a string when used in string parameters such as filenames. Because some data types (such as date/times) can be formatted in a variety of ways, the default representations (such as the ISO 8601 YYYY-MM-DD representation for dates) may not be appropriate. The `FormatDateTimeProperty()` command can be used to format date/time string properties.

Command Processor Global Properties

Property Name	Type	Description
`CreateOutput`	Boolean	Indicates whether commands should create output files (true) or not (false) – this is useful for speeding processing during initial testing.
`InitialWorkingDir`	String	The initial working directory (folder), initially defaulted to the main command file folder. This property is used with the `RunCommands()` command to help the processor keep track of changing working directories.
`InputStart`	DateTime	The global input period start, for read commands.
`InputEnd`	DateTime	The global input period end, for read commands.
`OutputStart`	DateTime	The global output period start, for commands that create or write time series.
`OutputEnd`	DateTime	The global output period end, for commands that create or write time series.
`OutputYearType`	String	The output year type, for commands that create or write time series.
`WorkingDir`	String	The working directory (folder), initially defaulted to the command file folder. All other relative paths will be relative to this location. The `SetWorkingDir()` can be used to modify the value; however, this is not recommended because it can lead to hard-coding paths in command files, which limits portability.

2.9 Using Templates and Logic Control Commands

TSTool processes a workflow of commands sequentially from the first command in the command file to the last command in the command file. This is generally appropriate for simple data processing workflows. However, if processing a large amount of data, it can be cumbersome to create long command files. Consequently, two options (templates and logic control commands) have been implemented to allow scaling processing for large datasets. The benefit and downside of using these techniques should be evaluated when implementing command workflows. Note also that both approaches can be used together.

2.9.1 Templates

Templates are text files that include templating language supported by the FreeMarker template library (see *freemarker.org*). Text files can include TSTool command files, TSTool time series product files, data files, HTML files for websites, and other files. The `ExpandTemplateFile()` command (see the ***Commands… Template Processing*** menu) is used to expand the template into a final TSTool command file that can be run as usual. Conceptually the process to expand a template TSTool command file is as follows.

1. Edit the template file with a text editor, including FreeMarker syntax to implement logic and looping. If the template is a TSTool command file, it is helpful to copy and paste commands

from the bottom part of command editor dialogs. It is also helpful to develop the core logic of a TSTool command file as usual with TSTool, and then edit the command file with a text editor to convert to a template. Include FreeMarker syntax as documented in the `ExpandTemplateFile()` command documentation and on the FreeMarker website.

2. Create a small TSTool command file that includes a `ExpandTemplateFile()` command. The purpose of this command file is to process the template command file created in the previous step into the final TSTool command file that can be run in the next step. The `ExpandTemplateFile()` command utilizes data from the TSTool processor to expand the template, including:
    a. Processor properties including built-in properties (e.g., global input and output periods) and user-defined properties (e.g., set with `SetProperty()` command and other commands).
    b. One-column tables that can be used to iterate in template lists. If necessary, one-column tables can be created from larger tables using the `CopyTable()` command.
3. Open and run the expanded TSTool command file to run the final commands. Any errors in logic will need to be addressed by editing the template file and re-expanding to the final command file.

The benefits of using a template for a TSTool command file are as follows, and should be contrasted with the downside and the next section on logic control commands:

- A relatively small template file can be expanded into a file that represents a complete workflow or data product.
- Because the final file is completely expanded, any errors in that file can be reviewed at the specific location of the error. For example, if a block of TSTool commands is reproduced 100 times, an error in the 98th block will be very evident because all the commands are present in the final command file.
- Developing a robust template command file can encompass the primary logic of a workflow.

The downside of using a template for a TSTool command file are as follows:

- The final expanded command file may be very large. Consequently, the file may take a long time to load into TSTool and may actually cause TSTool to run out of memory. One reason that TSTool can be slow loading the file is that TSTool runs commands in "discovery" mode to ensure that data are correct and can be used by other commands. For example, commands that read time series will generally attempt to read the time series metadata so that time series can be listed in choices for other commands. The following work-arounds may be helpful:
    - Minimize the length of the template command file so as to minimize the length of the expanded command file.
    - Use the ***File… Open… Command File (no discovery)…*** menu to open the command file. This will avoid running the discovery step, which his appropriate since the expanded command file should generally not be edited after being auto-generated.
- The FreeMarker template language syntax may be difficult to understand and master competency.
- The `${property}` notation used by FreeMarker conflicts with the similar notation used by TSTool. Consequently, where appropriate, the notation needs to be "escaped" by using FreeMarker literal string notation. See the `ExpandTemplateFile()` command documentation for examples.

2.9.2 Logic Control Commands

TSTool 10.26.00 introduced the `If()` command and TSTool 10.31.00 introduced the `For()` command. It has never been the intent of TSTool developers to create a full programming language. However, these basic commands provide fundamental capability that allow logic control in TSTool command files. These commands can be used in place of templates in order to expand a block of command logic to process a list of input data. The command functionality is limited to simple logic constructs.

The `If()` and `For()` commands work by using processor properties to evaluate and manage the state of processing. Processor properties are set with the `SetProperty()` command and can also be set with a number of other commands. See the **Commands… General – Running and Properties** command menu and also look for command parameters used to set a property. For example, the `CopyTable()` command sets a property for the row count of the copied table, which is useful for error checks.

The benefits of using logic commands (in particular the `For()` command) are listed below, and should be contrasted with the downside and the previous section on templates:

- Because logic is repeated within the `For()` and `EndFor()` command block, command files can be much shorter than if using templates. Consequently, command files load faster and in most cases also run faster.
- The syntax of TSTool logic commands is controlled by TSTool editors and parameters are edited with command editor dialogs, thereby providing guidance to users.

The downside of using logic commands are listed below:

- `For()` commands essentially compress logic into a repeating loop of commands. This makes it more difficult to troubleshoot errors. Additional TSTool features will be added over time to help.
- There is more reliance on processor properties, which cannot be checked until the commands are run. Consequently, users may have more difficulty troubleshooting command file logic.