US Army Corps
of Engineers
Hydrologic Engineering Center

HEC-DSSVue
HEC Data Storage System Visual Utility Engine

User's Manual

Version 2.0
July 2009

CPD-79

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to the Department of Defense, Executive Services and Communications Directorate (0704-0188). Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE *(DD-MM-YYYY)* July 2009	**2. REPORT TYPE** Computer Program Documentation	**3. DATES COVERED** *(From - To)*

4. TITLE AND SUBTITLE HEC-DSSVue HEC Data Storage System Visual Utility Engine User's Manual	**5a. CONTRACT NUMBER** **5b. GRANT NUMBER** **5c. PROGRAM ELEMENT NUMBER**
6. AUTHOR(S) CEIWR-HEC	**5d. PROJECT NUMBER** **5e. TASK NUMBER** **5F. WORK UNIT NUMBER**
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army Corps of Engineers Institute for Water Resources Hydrologic Engineering Center (HEC) 609 Second Street Davis, CA 95616-4687	**8. PERFORMING ORGANIZATION REPORT NUMBER** CPD-79
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**10. SPONSOR/ MONITOR'S ACRONYM(S)** **11. SPONSOR/ MONITOR'S REPORT NUMBER(S)**

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for Public Release. Distribution of this document is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The Hydrologic Engineering Center's Data Storage System, or HEC-DSS, is a database system designed to efficiently store and retrieve scientific data that is typically sequential. Such data types include, but are not limited to, time series data, curve data, spatial-oriented gridded data, textual data (such as this manual), and others. The system was designed to make it easy for users and application programs to retrieve and store data.

HEC-DSSVue (HEC-DSS Visual Utility Engine) is a graphical user interface program for viewing, editing, and manipulating data in HEC-DSS database files.

15. SUBJECT TERMS
HEC-DSS, HEC-DSSVue, Data Storage System, database, computer program, USACE, US Army Corps of Engineers, Hydrologic Engineering Center, HEC, database, system, store, retrieve, scientific data, curve data, spatial-oriented gridded data, textual data, application, viewing, editing, software, Java, DSSMATH, function, graphics, graphical user interface, GUI, plot, tabulate, mathematical functions, datasets, sequential, time series data, hydrologic studies, analysis, study data, results, alternative plans, conditions, independent, design, rapid, optimal, hashing, algorithm, flexible, daily flow values, precipitation, rating tables, conditional, relational, block, basic, unit, values, variable, pathname, parts, array

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	490	**19b. TELEPHONE NUMBER**

HEC-DSSVue
HEC Data Storage System
Visual Utility Engine

User's Manual

Version 2.0
July 2009

US Army Corps of Engineers
Institute for Water Resources
Hydrologic Engineering Center
609 Second Street
Davis, CA 95616

(530) 756-1104
(530) 756-8250 FAX
www.hec.usace.army.mil

CPD-79

HEC Data Storage System Visual Utility Engine, HEC-DSSVue
User's Manual

Use of the software described by this document is controlled by certain terms and conditions. The user must acknowledge and agree to be bound by the terms and conditions of usage before the software can be installed or used. For reference, a copy of the terms and conditions of usage are included in Appendix L of this document so that they may be examined before obtaining and loading the software. The software described by this document can be downloaded for free from our internet site (www.hec.usace.army.mil).

HEC cannot provide technical support for this software to non-Corps users. In the past, for non-Corps users, HEC had provided a list of possible vendors for assistance or support for HEC software. By direction of USACE counsel HEC has discontinued this practice and has removed the list from our web site. Non-Corps individuals and organizations should use any internet search engine to locate a vendor that can provide support for the HEC software of interest. However, we will respond to all documented instances of program errors. Documented errors are bugs in the software due to programming mistakes not model problems due to user-entered data.

This document contains references to product names that are trademarks or registered trademarks of their respective owners. Use of specific product names does not imply official or unofficial endorsement. Product names are used solely for the purpose of identifying products available in the public market place.

Microsoft, *Window*s, and *Excel* are registered trademarks of Microsoft Corp.

Solaris and *Java* are trademarks of Sun Microsystems, Inc.

RedHat, is a trademark of Red Hat, Inc.

Table of Contents

Chapters

Table of Contents

Chapters

Table of Contents

Chapters

Table of Contents

Chapters

Table of Contents

Chapters

Table of Contents

Chapters

Table of Contents

Chapters

Table of Contents

Chapters

Table of Contents

Chapters

Table of Contents

Chapters

Table of Contents

Chapters

8 **Scripting (continued)**

Table of Contents

Chapters

Table of Contents

Chapters

Appendices

Table of Contents

Appendices

Appendix C Example Scripts

List of Figures

Figure Number

List of Figures

Figure Number

List of Figures

Figure Number

List of Figures

Figure Number

List of Figures

Figure Number

List of Figures

Figure Number

List of Figures

Figure Number

List of Figures

Figure Number

List of Tables

Table Number

Foreword

The U.S. Army Corps of Engineers' Hydrologic Engineering Center Data Storage System, or HEC-DSS, is a database system designed to efficiently store and retrieve scientific data that is typically sequential. Such data types include time series data, curve data, spatial-oriented gridded data, and textual data (such as this manual). The system was designed to make it easy for users and application programs to retrieve and store data. HEC-DSSVue (HEC-DSS Visual Utility Engine) is a graphical user interface program for viewing, editing, and manipulating data in HEC-DSS database files.

HEC-DSS originated at the Hydrologic Engineering Center in 1979 under the direction of Dr. Art Pabst. Since that time, many have worked on the development of the HEC-DSS software and the HEC-DSS utility programs, including William Charley, Al Montalvo, Carl Franke, Paul Ely, Robert Carl, Dennis Huff, Mike Perryman and numerous others.

Mr. William Charley led the software design and development team, as well as created the main interface screen and other components. Resource Management Associates (RMA) of Fairfield, California, under the direction of Dr. John DeGeorge, aided in the development of HEC-DSSVue. The RMA staff included Richard Rachiele, who translated the DSSMATH functions into Java, Mark Ackerman, who wrote the math function screens and plug-in capabilities, Shannon Newbold, who was responsible for the graphics and Peter Morris, who designed and implement the graphical editor. Mike Perryman of HEC led the development of the scripting capabilities.

Dr. Cassie Carter, under contract with RMA, wrote the initial draft of this manual. Shannon Larson, William Charley, and Mike Perryman wrote updates for the current version of this document. Amanda Waller and Penni Baker tirelessly reviewed, corrected and formatted the document for publication.

CHAPTER 1

Introduction

The Hydrologic Engineering Center's (HEC) Data Storage System Visual Utility Engine (HEC-DSSVue) is a graphical user interface program for viewing, editing, and manipulating data in the HEC Data Storage System (HEC-DSS) database files. With HEC-DSSVue, you may plot, tabulate, and edit data, as well as manipulate data with over fifty mathematical functions. Along with these functions, HEC-DSSVue provides several utility functions that allow you to enter datasets into a database, rename dataset names, copy data sets to other HEC-DSS database files, and delete datasets.

Typically, you will select datasets from a sorted/filtered list of names in a HEC-DSS database file. HEC-DSSVue also incorporates the "Jython" standard scripting language that allows you to specify a routine sequence of steps in a text format and then execute the sequence from a user-defined button or a "batch" process.

HEC-DSSVue was written using the Java programming language that allows it to be run under a variety of different operating systems. Fully supported systems include Microsoft Windows, Sun Solaris (UNIX), RedHat Linux and other systems (contact HEC for the availability of these.)

1.1 Overview of the HEC Data Storage System

HEC-DSS is a database system designed to efficiently store and retrieve scientific data that is typically sequential. Such data types include, but are not limited to, time series data, paired (curve) data, spatial-oriented gridded data, and textual data (such as this manual). The system was designed to make it easy for users and application programs to retrieve and store data.

1.1.1 Background

HEC-DSS was the result of a need that emerged in the late 1970s. Before HEC-DSS, most hydrologic studies were performed in a step-wise fashion by passing data from one analysis program to another in a manual mode.

While this was functional, it was not very productive. Programs that used the same type of data, or that were sequentially related, did not use a common data format. Also, each program had to have its own set of graphics routines or other such functions to aid in the program's use.

The Kissimmee River study was performed by HEC for the Jacksonville District beginning in 1978 and required an orderly approach to properly manage the study data and analysis results. A large number of alternative plans and conditions were to be processed in this project. This study gave birth to the first version of HEC-DSS. The basic design provided for the storage of data in a standard form that was independent of any particular program. The data would be provided to the programs when it was needed and results would be stored in the same independent form for use by utilities and other applications programs. The early design of HEC-DSS was conceived to support files containing up to a few thousand data sets. As the use of HEC-DSS expanded into real-time data storage applications, data files were written to manage hundreds of thousands of data sets.

The current HEC-DSS version is designed for rapid storage and retrieval of datasets from files containing as few as forty to fifty datasets to files containing more than several million data sets. HEC-DSS Version 6-Qx and later can have database files up to 8 Gigabytes.

1.1.2 HEC-DSS Contrasted with Other Database Systems

HEC-DSS is designed to be optimal for storing and retrieving large sets, or series, of data. HEC-DSS incorporates a modified hashing algorithm and hierarchical design for database accesses. This algorithm provides quick access to data sets and an efficient means of adding new data sets to the database. Additionally, HEC-DSS provides a flexible set of utility programs and is easy to add to a user's application program. These are the features that distinguish HEC-DSS from most commercial database programs and make it optimal for scientific applications.

HEC-DSS is designed specifically for the storage and retrieval of large sets, or series, of data. These include daily flow values, hourly precipitation measurements, rating tables, and pages of text information. HEC-DSS is not optimized for dealing with small data sets or single data values, nor is it effective at conditional data searches common to relational database systems. In contrast, most commercial databases are designed for small sets of elemental data. Such elemental data includes employee records, accounting data, and inventory of stock.

Commercial databases usually employ a relational model in which data is stored in a related manner. A relational database can be viewed essentially as a collection of tables. This type of system requires the construction of a data definition or data dictionary file. Although a relational database requires some initial setup, it can effectively store short data sets comprised of both characters and numbers. Relational database systems use the ANSI ratified Structured Query Language (SQL) to access data.

While relational databases are ideal for elemental data sets, they are not as practical for longer series of data. HEC-DSS, however, is designed for such sets of data. HEC-DSS database files are not defined by a data definition file like the relational model requires so there is no set up required by the user. (HEC-DSS data is defined by the pathname and conventions used.) The type of data generally stored in HEC-DSS does not lend itself well to a query language such as SQL although the selective catalog feature has some similar capabilities.

Also unlike many commercial database systems, the HEC-DSS was designed to be easily added to a user's application program. In traditional "C" and FORTRAN programs, only two or three function calls are needed to access data. Those calls identify the database, the pathname, and a time window (if desired). Besides these languages, an extensive set of classes are available in both C++ and Java languages (including some access through Visual Basic), which are the languages used to develop most HEC programs.

1.2 General Concepts for HEC-DSS

HEC-DSS uses a block of sequential data as the basic unit of storage. This concept results in a more efficient access of time series or other sequentially related data. Each block contains a series of values of a single variable over a time span appropriate for most applications. The basic concept underlying HEC-DSS is the organization of data into records of continuous, applications-related elements as opposed to individually addressable data items. This approach is more efficient for scientific applications than a conventional database system because it avoids the processing and storage overhead required in assembling an equivalent record from a conventional system.

Data is stored in blocks, or records, within a file and each record is identified by a unique name called a "pathname". Each time data is stored or retrieved from the file, the data's pathname must be given. The data and information about the data (e.g., units) is stored in a "header array". HEC-DSS automatically stores the name of the program writing the data, the number of times the data has been written to, and the last written date and time. HEC-DSS documents stored data completely via information contained in the pathname and stored in the header so no additional information is required to identify the data. The self-documenting nature of the database allows information to be recognized and understood months or years after data has been stored.

The pathname is the key to the data's location in the database. HEC-DSS analyzes each pathname to determine a "hash" index number. This index determines where the data set is stored within the database. The design ensures that very few disk accesses are made to retrieve or store datasets. One data set is not directly related to another so there is no need to update other areas of the database when a new dataset is stored.

Because of the self-documenting nature of the pathname and the conventions adopted, there is no need for a data dictionary or data definition file as required with other database systems. In fact, there are no database creation tasks or any database setup. Both HEC-DSS utility programs and applications that use HEC-DSS will generate and configure HEC-DSS database files automatically. There is no pre-allocation of space; the software automatically expands the file size as needed.

A HEC-DSS database file has a user-specified conventional name with an extension of "dss". As many database files as desired may be generated and there are no size limitations, apart from available disk space. Corps offices have HEC-DSS files that range from a few datasets to thousands. HEC-DSS adjusts internal tables and hash algorithms to match the database size so as to access both small and very large databases efficiently.

HEC-DSS database files are "direct-access" binary files with no published format. Only programs linked with the HEC-DSS software library can be used to access HEC-DSS files. Direct access files allow efficient retrieval and storage of blocks of data compared to sequential files.

A principal feature of HEC-DSS is that many users can read and write data to a single database at the same time. This multi-user access capability is implemented with system record locking and flushing functions. There is no daemon or other background program managing accesses to a database. A database may exist on a Windows or UNIX server machine, which can be accessed by users on PC's or other computers via NFS or the Microsoft network, as long as locking and flushing functions are implemented.

1.2.1 Pathnames

HEC-DSS references datasets, or records, by their *pathnames*. A pathname may consist of up to 391 characters and is, by convention, separated into six parts, which may be up to 64 characters each. Pathnames are automatically translated into all upper case characters.

Pathnames are separated into six parts (delimited by slashes "/") labeled "A" through "F", as follows:

/A/B/C/D/E/F/

The naming convention for pathname parts is listed in the table below:

Part	Description
A	Project, river, or basin name
B	Location
C	Data parameter
D	Starting date of block, in a nine-character military format
E	Time interval
F	Additional user-defined descriptive information

An example pathname for regular-interval time series might be:

```
/RED RIVER/BEND MARINA/FLOW/01JAN1995/1DAY/OBS/
```

1.2.2 Catalogs

HEC-DSS utility programs, including HEC-DSSVue, will generate a list of the pathnames in a HEC-DSS file and store that list in a "catalog" file. The catalog file is a list of the record pathnames in the file, their last written date and time, and the name of the program that wrote that record. The catalog is usually sorted alphabetically by pathname parts. Each pathname has a record tag and a reference number, either of which may be used in place of the pathname in several of the utility programs. The name given to the catalog file is the HEC-DSS file's name with an extension of ".dsc".

A special catalog file, the "condensed catalog", is useful mainly for time series data. In this type of catalog, pathname parts display in columns, and pathnames for time series data, differing only by the date (Part D), are referenced with one line.

1.2.3 Data Conventions

HEC-DSS can store data of most types using a pathname of any structure. To facilitate the ability of application and utility programs to work with and display data, standard record conventions were developed. These conventions define what should be contained in a pathname, how data is stored, and what additional information is stored along with the data. For regular-interval time series data (e.g., hourly data), the conventions specify that data is stored in blocks of a standard length, uniform for that time interval, with a pathname that contains the date of the beginning of the block and the time interval. The conventions identify how a pathname for the data should be constructed. Conventions have been defined for regular and irregular interval time series data, paired (curve) data, gridded data (such as NEXRAD radar data), and text (alphanumeric) data.

Regular-interval time series data is data that occurs at a standard time interval. This data is divided into blocks whose length depends on the time interval. For example, hourly data is stored with a block length of a month, while daily data is stored with a block length of a year. Only the date and time of the first piece of data for a block is stored; the times of the other data elements are implied by their location within the block. If a data element, or a set of elements, does not exist for a particular time, a missing data flag is placed in that element’s location. Data quality flags may optionally be stored along with a regular-interval time series record.

Irregular-interval time series data does not have a constant time interval between values. This type of data is stored with a date/time stamp for each element. The user-selectable block size is based on the amount of data that is to be stored. For example, the user may select a block length of a month or a year. Because a date/time stamp is stored with each data element, approximately twice the amount of space is required compared with regular-interval time series data. Data quality flags may optionally be stored along with an irregular-interval time series record.

A convention for paired data has been defined for data that generally defines a curve. Paired data is for rating tables, flow-frequency curves, and stage-damage curves. One paired data record may contain several curves within it as long as the record has a common set of ordinates. For example, a stage-damage curve will contain a set of stages and may have associated residential damages, commercial damages, and, agricultural damages; however, a stage-damage curve and a stage-flow curve cannot be stored in the same record.

Conventions for spatially gridded data can be found in the "GridUtil" User's Manual. Text conventions are reviewed in Section 1.4.

1.3 Time Series Conventions

This section covers the conventions for both regular- and irregular-interval time series data. There are four data types that are recognized by DSS, and are listed in the table (see following page).

Data Type	Example
PER-AVER	Monthly Flow
PER-CUM	Incremental Precipitation
INST-VAL	Stages
INST-CUM	Precipitation Mass Curve

1.3.1 Default Pathname Parts

Both regular- and irregular-interval time series record pathnames have the same A, B, C, and F parts. Data blocks are labeled with a six part pathname. The parts are referenced by the characters A, B, C, D, E, and F, and are separated by a slash "/", so that a pathname would look as follows:

/A/B/C/D/E/F/

The default conventions for the pathname parts are outlined in the following sections:

Part A – Group

Part A is required for regular- and irregular-interval time series data and is used as a way to group records. Usually this information is a watershed name, study name, project, river, or basin name; a name that allows the user to group associated records.

Part B - Location

Part B is required for regular- and irregular-interval time series data and is the basic location identifier, which is generally the site name. A similar identifier, such as a project ID, USGS gage ID, or NWS station ID may be used. Part B is required.

When a hydrograph is routed from one location to another, the recommended Part B is *LOC1.LOC2*, where *LOC1* is the identifier to which the flows are routed, and *LOC2* is an identifier for the location from which flows are routed. The second location (*.LOC2)* is optional. For example, Part B may be left out in situations where there is only one routed hydrograph for a location.

Part C – Parameter

Part C identifies the basic data parameter for regular- and irregular-interval time series data. A dash is used as a sub-separator if further identification is needed. Additional information about the parameter, such as how it was obtained (e.g., OBSERVED), should be given in Part F. Examples of valid parameters are:

```
FLOW
ELEV
PH
PRECIP-INC
STAGE
TEMP-AIR
TEMP-WATER
```

Recommended C-parts for flows associated with stream locations are as follows:

Part C	Description
FLOW	Total flow
FLOW-LOC	Local flow; that is, flow generated only from the subbasin that has an outlet at the specified location.
FLOW-CUM LOC	Cumulative local flow. This is the flow from all subbasins downstream from the nearest upstream reservoirs.
FLOW-COMB	Combined flow. This is the total flow minus the local flow.
FLOW-DIVERT	Flow diverted out of the river at this location.
FLOW-mod	*mod* is a user-specified modifier for the flow. For example, FLOW-POWER would designate a hydrograph from a power plant, and, FLOW-IN would be for a component of reservoir inflow.
FLOW-IN	Reservoir inflow.
FLOW-OUT	Reservoir outflow.
ELEV-RES	Reservoir elevation.
STORAGE	Reservoir storage.

Part D - Block Start Date

Part D identifies the starting date of the data block. For Part D the conventions are different for regular-interval (see Section 1.3.2) and irregular-interval (see Section 1.3.3) time series data.

Part E - Time Interval or Block Length

Part E defines the time interval for regular-interval data (see Section 1.3.2), or the block length for irregular-interval data (see Section 1.3.3).

Part F – Descriptor

Part F is used to provide any additional information about the regular- and irregular-interval time series data. The use of Part F may vary from application to application as appropriate, and may contain several additional qualifying pieces of information separated by a dash "-". If several forms of data exist, such as *OBSERVED or FORECAST*, *PLAN A* or *TEST 2*, they may be reflected in Part F. Generally, the order of multi-descriptors of Part F should be from most to least significant.

1.3.2 Regular-Interval Time Series Conventions

Regular-interval time series data is stored in "standard size" blocks whose length depends upon the time interval of the data. For example, daily time interval data are stored in blocks of one year (365 or 366 values) while monthly values are stored in blocks of ten years (120 values). If data does not exist for a portion of the full block, the missing values are set to the missing data flag "-901.0".

The starting and ending times of a block correspond to standard calendar conventions. For example, for period average monthly data in the 1950's, Part D (date part) of the pathname would be *01JAN1950*, regardless of when the first valid data occurred (e.g., it could start in 1958). The 1960's block starts on *01JAN1960*.

Average period data values are stored at the end of the period over which the data is averaged. For example, daily average values are given a time label of *2400* hours for the appropriate day and average monthly values are labeled at *2400* hours on the last day of the month. If values occur for times other than the end-of-period time, that time offset is stored in the header array. For example, if daily average flow reading's are recorded at *6:00 a.m.* (i.e., the average flow from *6:01 a.m.* of the previous day to *6:00 a.m.* of the current day), then an offset of *360* (minutes) will be stored in the header array.

Part D - Block Start Date

Part D should be a nine-character military style date for the start of the standard data block (not necessarily the start of the first piece of data). Valid dates include *01JAN1982*, *01MAR1982*, and *01JAN1900* for daily data, hourly data, and yearly data. Invalid dates include *01JAN82* (seven characters) and *14APR1982* for daily data (*14APR1982* is not the start of a standard daily block).

Part E - Time Interval

Part E consists of an integer number and an alphanumeric time interval specifying the regular data interval. Valid alpha entries are **MIN, HOUR, WEEK, MON,** and **YEAR**. The valid intervals and block lengths are:

Valid Data Intervals	Block Length
1MIN, 2MIN, 3MIN, 4MIN, 5MIN, 6 MIN, 10MIN, 12MIN	One Day
15MIN, 20MIN, 30MIN, 1HOUR, 2HOUR, 3HOUR, 4HOUR, 6HOUR, 8HOUR, 12HOUR	One Month
1DAY	One Year
1WEEK, 1MON, SEMI-MONTH, TRI-MONTH	One Decade
1YEAR	One Century

Examples of regular-interval pathnames are:

a. Daily USGS observed flow for station *0323150* for calendar year *1954* might be named:
```
/USGS/0323150/FLOW/01JAN1954/1DAY/OBS/
```

b. Six-hourly forecasted flow may have a pathname of:
```
/RED RIVER/DENISION/FLOW/01JUN2010/6HOUR/FORECAST/
```

1.3.3 Irregular-Interval Time Series Conventions

The irregular-interval time series conventions are similar to the regular-interval conventions except that an explicit date and time is stored with each piece of data whereas in regular-interval time series the date and time are implied by the location of the data within the block. Irregular-interval data is stored in variable length blocks while regular-interval data is stored in fixed length blocks. The block lengths are days, months, years, decades, and centuries.

The number of values that may be stored in one record is indefinite although it is prudent to choose a size that will be less than 3,000. The user selects the appropriate block length. For example, if the data to be stored occurred once every one to two hours, a monthly block would be appropriate. If data were recorded once or twice a day, use a yearly block. One would not want to store data that occurred eight or more times a day in a yearly block (about 3,000 values) because that may exceed dimension limits in some DSS programs.

All data are stored in variable length blocks that are incremented a set amount when necessary. Initial space for 100 data values is allocated and additional increments are for fifty data values unless otherwise set. (When the 101st data value is added to the record, a new record with a length of 150 values is written.)

Part D - Block Start Date

Part D should be a nine-character military style date for the first day of the standard data block (not necessarily the start of the first piece of data). For example, data stored in a daily block beginning on *March 23, 1952* at *3:10 p.m.* would have a Part D of *23MAR1952*. If the same data were stored in a monthly block, Part D would be *01MAR1952*. The same data in a yearly block would have a Part D of *01JAN1952*, and as a decade block, *01JAN1950*.

Part E - Block Length

Part E indicates the length of the time block for irregular-interval data, whether it is a day, a month, a year, decade, or a century. For irregular-interval data, Part E consists of **IR-** concatenated with the block length:

```
IR-DAY
IR-MONTH
IR-YEAR
IR-DECADE
IR-CENTURY
```

The same data may be stored in blocks of different lengths. DSS stores these as different records and treats them as a completely different dataset.

An example of a pathname for irregular-interval data is:

```
/SANTA ANA/PRADO/FLOW/01JAN2008/IR-MONTH/OBS/
```

1.4 Paired Data (Curve Data) Conventions

Paired data is a group of data that represents a two variable relationship. Typical examples are data that make up a curve (e.g., a rating table or a flow-frequency curve). Several curves may be stored in the same record if one of the variables is the same. For example several frequency-damage curves may be stored in the same record, where the curves may be residential, commercial, etc. A scale associated with the variable may be one of three types: linear, logarithmic, or probability. The pathname part identifiers are as follows:

Part A – Group

Part A for paired data is used as a way to group records. Usually this information is a watershed name, study name, project, river, or basin name; a name that allows the user to group associated records.

Part B – Location

Part B is the basic location identification of the paired data and could be a control point, damage reach ID, station ID, or other identifier.

Part C – Parameters

Because paired data represents a relationship between two parameters, Part C should contain the two parameter names separated by a hyphen (-). Examples of parameters are:

```
ELEV-DAMAGE
ELEV-FLOW
FREQ-FLOW
STATION-ELEV
```

In the above examples, *ELEV, FREQ*, and *STATION* are referred to as the first, independent variable while *DAMAGE, FLOW* and *ELEV* are the second, dependent variable.

Part D - Optional Descriptor

Part D of the pathname is used to provide any further descriptions of the data. Part D may vary from application to application as appropriate, and is often null.

Part E - Time Descriptor

Part E of the pathname is used only if the paired data is representative of a specific point in time.

Part F - General Descriptor

This part identifies a unique descriptor of the data such as the situation, condition, or alternative plan name associated with the data. This part is included in labeling of data in some output utilities.

An example of a pathname for paired data is:

```
/ALLEGHENY/NATRONA/ELEV-DAMAGE//2020/FLOOD PROOF PLAN B/
```

1.5 Text Data Conventions

Text data is defined as generic alpha-numeric lines of text where each line is preceded by a carriage return character and ends with a line feed character. It does not, at this time, include other types of characters such as those that would be used to create a graphical display. There are no definitive size limitations for a DSS text record but it is recommended that a record contain no more than about 1,000 lines of text. There are no conventions set for the structure of the pathname; however, it is recommended that the pathname parts be labeled in a descending order of importance and that the pathname indicate that the record contains text data and not one of the other types of data.

1.6 General FILE Conventions

Files of a certain type can be stored in an HEC-DSS database file. The allowable types are: .jpg files, .mp3 files, .pdf files, .xls files, and .doc files. This allows the storage of various metadata along with regular data. Photographs, fact sheets, sounds, and other metadata can be presented along with time series and paired data.

In HEC-DSSVue on Microsoft Windows computers, assessing these types of metadata will launch the native application associated with the extension of that dataset. For example, accessing a .pdf dataset will launch Adobe Acrobat, a .xls data set will launch Excel, and so forth. One exception is that an image file will launch a Java window with the image displayed.

Part A – Group

Part A for "file" data is used as a way to group records. Usually this information is a watershed name, study name, project, river, or basin name; a name that allows the user to group associated records.

Part B – Location

Part B is the basic location identification of the "file" data If the "file" is a photograph, it is the location of the picture.

Part C – File Name

Part C contains the filename of the "file" data and excludes directory information.

Part D – Meta Data Type

This part of the pathname can only be either FILE or IMAGE to reflect what the metadata is.

Part E – File Extension

Part E contains the original extension of the file.

Part F - General Descriptor

The F part identifies a unique descriptor about what the file contains.

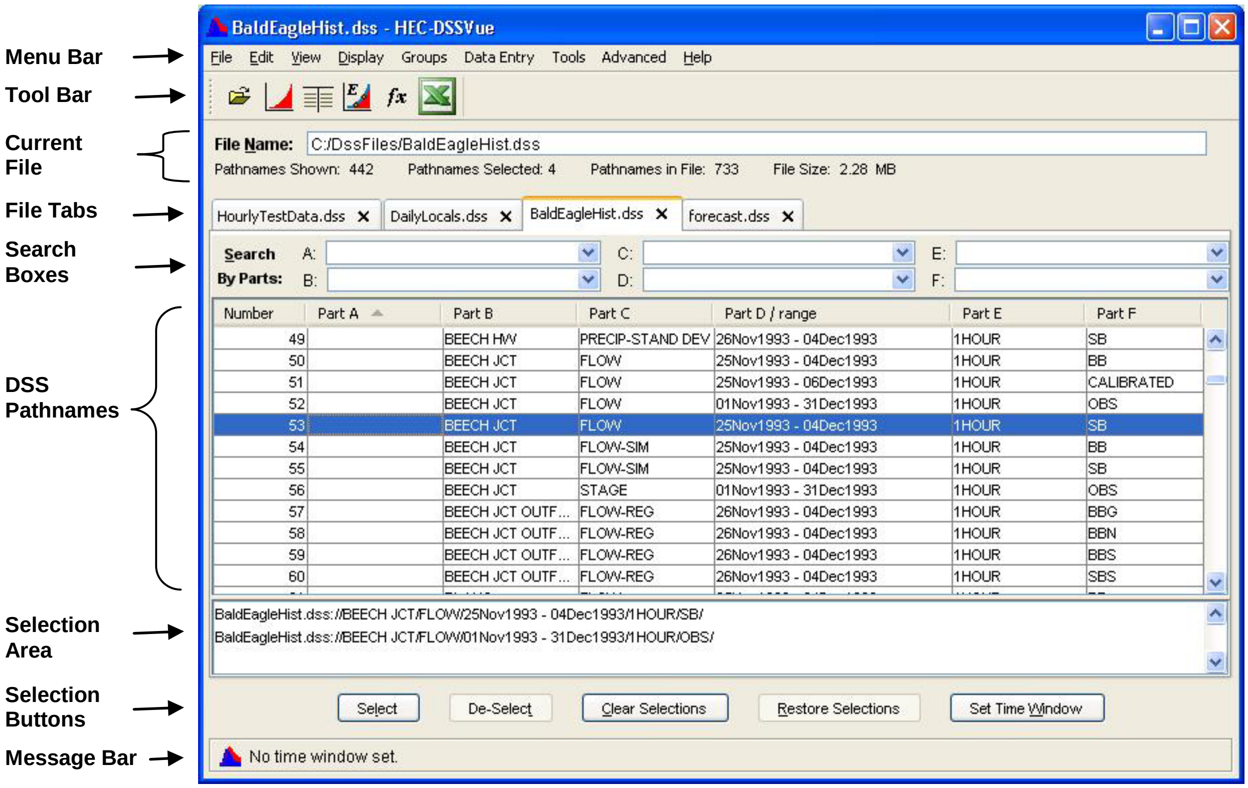
Example pathnames that include "file" data are listed below:

```
/SACRAMENTO/SHASTA/IMG_0009.JPG/IMAGE/JPG/SPILLWAY – AUG 2006/

/LOWER COLORADO/TRAVIS/TRAVIS.PDF/FILE/PDF/FACT SHEET/

/STONES/NO SECURITY/GIMME SHELTER.MP3/FILE/MP3/CARLS FAVORITES/
```

CHAPTER 2

Using HEC-DSSVue: An Overview

With HEC-DSSVue, you can access, view, and manipulate data stored in HEC-DSS database files using a variety of utilities and functions available from the main HEC-DSSVue screen, shown in Figure 2.1.



File Name: C:/DssFiles/BaldEagleHist.dss

Pathnames Shown: 442 Pathnames Selected: 4 Pathnames in File: 733 File Size: 2.28 MB

Number	Part A	Part B	Part C	Part D / range	Part E	Part F
49		BEECH HW	PRECIP-STAND DEV	26Nov1993 - 04Dec1993	1HOUR	SB
50		BEECH JCT	FLOW	25Nov1993 - 04Dec1993	1HOUR	BB
51		BEECH JCT	FLOW	25Nov1993 - 06Dec1993	1HOUR	CALIBRATED
52		BEECH JCT	FLOW	01Nov1993 - 31Dec1993	1HOUR	OBS
53		BEECH JCT	FLOW	25Nov1993 - 04Dec1993	1HOUR	SB
54		BEECH JCT	FLOW-SIM	25Nov1993 - 04Dec1993	1HOUR	BB
55		BEECH JCT	FLOW-SIM	25Nov1993 - 04Dec1993	1HOUR	SB
56		BEECH JCT	STAGE	01Nov1993 - 31Dec1993	1HOUR	OBS
57		BEECH JCT OUTF...	FLOW-REG	26Nov1993 - 04Dec1993	1HOUR	BBG
58		BEECH JCT OUTF...	FLOW-REG	26Nov1993 - 04Dec1993	1HOUR	BBN
59		BEECH JCT OUTF...	FLOW-REG	26Nov1993 - 04Dec1993	1HOUR	BBS
60		BEECH JCT OUTF...	FLOW-REG	26Nov1993 - 04Dec1993	1HOUR	SBS

```
BaldEagleHist.dss://BEECH JCT/FLOW/25Nov1993 - 04Dec1993/1HOUR/SB/
BaldEagleHist.dss://BEECH JCT/FLOW/01Nov1993 - 31Dec1993/1HOUR/OBS/
```

No time window set.

Figure 2.1 HEC-DSSVue Main Window

This chapter provides an overview of the HEC-DSSVue's main window components and their basic functions.

2.1 HEC-DSSVue Main Window

The main window of HEC-DSSVue (Figure 2.1) consists of a **Menu Bar**, **Tool Bar**, **Current File Information**, **Files Tabs**, **Search Boxes**, a list of **DSS Pathnames**, **Selection Area**, **Selection Buttons**, and **Message Bar**. The following sections describe these features in detail.

2.1.1 Menu Bar

Main menu options in HEC-DSSVue (Figure 2.2) allow you to search for, select, and edit HEC-DSS data sets; control the display of pathnames and access plots and tables among other tasks. The HEC-DSSVue main menu options are as follows:

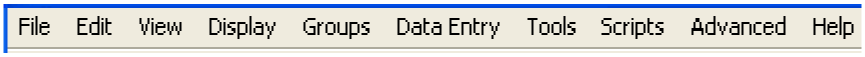


Figure 2.1 HEC-DSSVue Menu Bar

File
File menu commands include **New, Open, Close DSS File(s), Print Catalog Preview, Print Catalog**, and **Exit**. The file menu also lists the last six opened files.

Edit
Edit menu commands are **Tabular Edit, Graphical Edit, Edit in Excel, Select All, Rename Records, Delete Records, Undelete, Duplicate, Copy To**, and **Merge Records**.

View
The **View** menu allows you to customize the display of HEC-DSS pathnames, refresh the catalog and search pathnames. Available commands are **Pathname List, Pathname Parts, Condensed Catalog, No Pathnames, Unsorted List, Search pathnames by string, Search pathnames by parts,** and **Refresh Catalog**.

Display
Use the **Display** menu to view data. Options in this menu are **Plot, Plot Individual Data Sets, Tabulate, Tabulate in Excel, Display Data Options, Supplemental Information, Time Window, Charts**, and **NC DWR Duration Hydrograph**.

Groups
Use the **Group** menu to organize selected records into groups or view and edit existing Groups. **Group** menu commands include **Save Selected, Get, Plot, Plot Individual Sets, Tabulate, Math**, and **Merge**.

Data Entry
The **Data Entry** menu is used to add data and files to records. Commands include **Manual Time Series, Manual Paired Data, Manual Text, Import**, and **Export**.

Tools
The **Tool**s menu provides tools for manipulating DSS records. These tools include **Math Functions, Compare, Search for Value, Check File Integrity, Squeeze, Script Editor**, and **Script Selector**.

Custom (Optional) If your office has written their own plug-ins, those plug-ins can be made available under the **Custom** menu. The menu name is set from the plug-in. See the Writing Plug-ins Appendix for more information.

Scripts (Optional) If you use scripts, you can activate the **Scripts** menu from the **Script Editor** (under **Tools**) and have quick access to your selected scripts from this menu.

Advanced Through the **Advanced** menu you can create, view, or print a **Condensed Disk Catalog, Abbreviated Disk Catalog**, or a **Full Disk Catalogs**. Additional options are **Console Output, DSS Output, Status, Debug**, and **Program Options**.

Help The **Help** menu contains the **About** option, which displays the HEC-DSSVue version information and gives access to the terms and conditions of the software.

2.1.2 Tool Bar

The **Tool Bar** buttons provide shortcuts to frequently used menu commands:

Opens the **Open HEC-DSS File** browser that allows you to select and open a HEC-DSS file. This button would be equivalent to selecting **Open** in the **File** menu.

Plots the selected data, equivalent to selecting **Plot** in the **Display** menu.

Displays the selected data in a table, equivalent to selecting **Tabulate** in the **Display** menu.

Graphically edit the selected data, equivalent to selecting **Graphical Edit** in the **Edit** menu.

Open the **Math Functions** dialog with the selected data, equivalent to selecting **Math Functions** in the **Tools** menu.

(Optional) Open the selected records in Microsoft Excel, equivalent to selecting **Tabulate in Excel** in the **Display** menu. This function relies on the Excel plug-in to be installed.

2.1.3 File Tabs

Multiple DSS files can be opened and viewed in HEC-DSSVue at the same time (Figure 2.3). Each time you open a new dss file, a tab is created with the name of the dss file shown on it. The pathnames displayed are from the selected tab's DSS file. If more than one file is opened, selected pathnames will include the name of the DSS file they are from. For example:

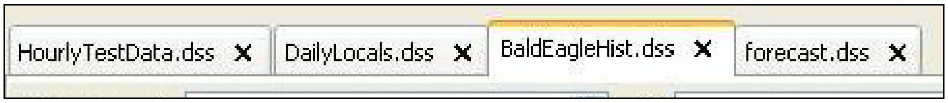
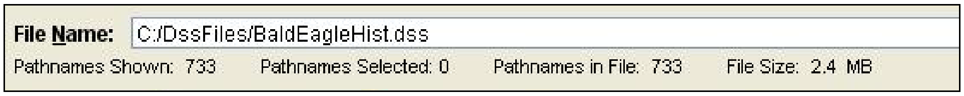


Figure 2.2 File Tabs

BaldEagleHist.dss://BALDE/FLOW/01DEC1993/1HOUR/BB/.

2.1.4 Current File Name

The **Current File** area displays the information of the selected tab’s DSS file. The **File Name** box (Figure 2.4) displays the file name and location of the selected HEC-DSS file.



File Name: C:/DssFiles/BaldEagleHist.dss

Pathnames Shown: 733 Pathnames Selected: 0 Pathnames in File: 733 File Size: 2.4 MB

Figure 2.3 File Name Box

> ***Tip:*** *If you know the exact location and name of the HEC-DSS database file you wish to open, you can type it directly into the* ***File Name*** *box (Figure 2.4) and then click the* ***Enter*** *key to open it.*

Beneath the **File Name** box (Figure 2.4) is statistical information on the pathnames contained in that file. **Pathnames Shown** displays the number of pathnames available in the DSS file and shown in the DSS Pathname list. **Pathnames Selected** provides the number of pathnames currently selected from the file. **Pathnames in File** shows the total number of records in the file, including files and images. And finally, **File Size** displays the amount of disk space the file uses.

2.1.5 Search Boxes

Searches of the DSS pathnames can be made by full pathnames or by pathname parts. These search methods are set through the **View** menu.

The selected search method will display a checkmark next to the menu option, see Figure 2.5.

To search and display all pathnames containing a specific text, from the **View** menu click **Search pathnames by string** (Figure 2.6). Then type the text into the **Search Pathnames** box and click **Search**. The pathnames in the file that contain the text will be displayed in the **DSS Pathname** table, see Figure 2.6.

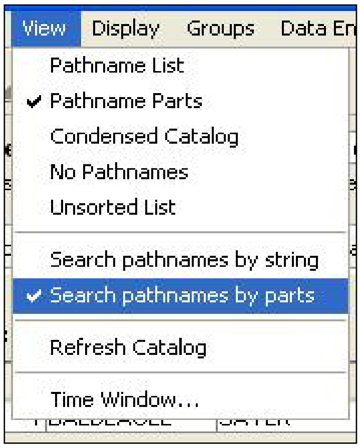
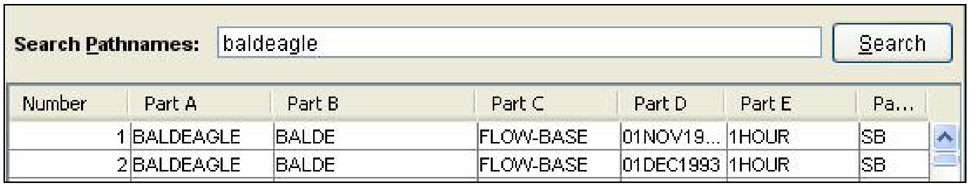


Figure 2.5 View Menu

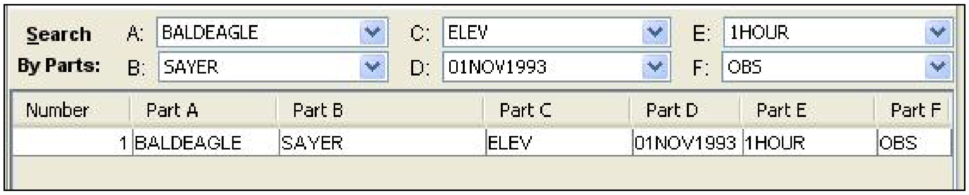
To search using DSS parts, select from the **View** menu click **Search pathnames by parts** (Figure 2.7). Six **Search By Parts** filter boxes will display, one for each DSS part. The available part names contained in the selected DSS file are listed in the filter boxes. When a part is selected in a filter box, only the pathnames that contain the filter text are shown in the **DSS Pathname** table (Figure 2.7).



Search Pathnames: baldeagle

Number	Part A	Part B	Part C	Part D	Part E	Pa...
1	BALDEAGLE	BALDE	FLOW-BASE	01NOV19...	1HOUR	SB
2	BALDEAGLE	BALDE	FLOW-BASE	01DEC1993	1HOUR	SB

Figure 2.6 Search Pathnames by String



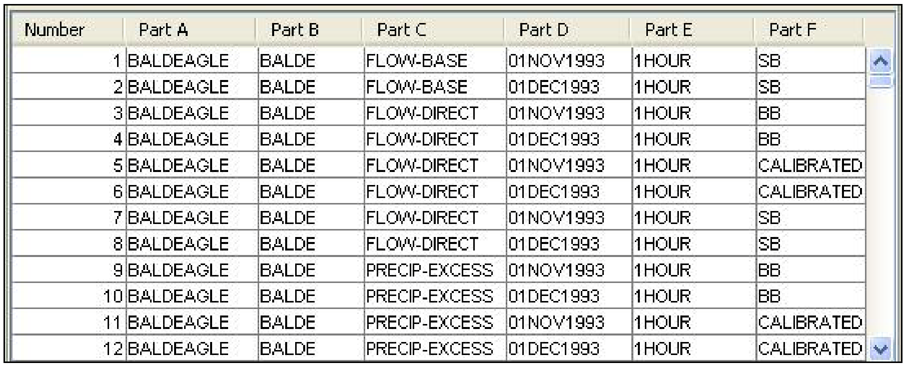
Search By Parts: A: BALDEAGLE B: SAYER C: ELEV D: 01NOV1993 E: 1HOUR F: OBS

Number	Part A	Part B	Part C	Part D	Part E	Part F
1	BALDEAGLE	SAYER	ELEV	01NOV1993	1HOUR	OBS

Figure 2.7 Search Pathnames by Parts

2.1.6 List of HEC-DSS Pathnames

Once you have opened an HEC-DSS file, its pathnames appear in a pane beneath the **Search By** filters (Figure 2.8). The pathnames can be displayed as a pathname sorted list (**Pathname List**), by DSS Parts (**Pathname Parts**), in the **Condensed Catalog** format, with **No Pathnames*** displayed, or as an **Unsorted List***. These options are selected from the **View** menu (Figure 2.5), as described in Section 2.5.1. An example of **Pathname by Parts** is shown in Figure 2.8.



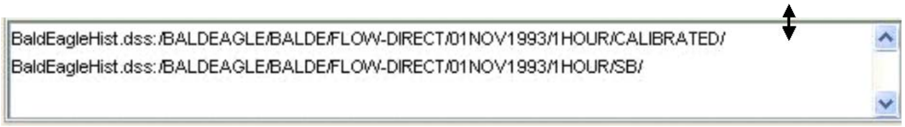
Number	Part A	Part B	Part C	Part D	Part E	Part F
1	BALDEAGLE	BALDE	FLOW-BASE	01NOV1993	1HOUR	SB
2	BALDEAGLE	BALDE	FLOW-BASE	01DEC1993	1HOUR	SB
3	BALDEAGLE	BALDE	FLOW-DIRECT	01NOV1993	1HOUR	BB
4	BALDEAGLE	BALDE	FLOW-DIRECT	01DEC1993	1HOUR	BB
5	BALDEAGLE	BALDE	FLOW-DIRECT	01NOV1993	1HOUR	CALIBRATED
6	BALDEAGLE	BALDE	FLOW-DIRECT	01DEC1993	1HOUR	CALIBRATED
7	BALDEAGLE	BALDE	FLOW-DIRECT	01NOV1993	1HOUR	SB
8	BALDEAGLE	BALDE	FLOW-DIRECT	01DEC1993	1HOUR	SB
9	BALDEAGLE	BALDE	PRECIP-EXCESS	01NOV1993	1HOUR	BB
10	BALDEAGLE	BALDE	PRECIP-EXCESS	01DEC1993	1HOUR	BB
11	BALDEAGLE	BALDE	PRECIP-EXCESS	01NOV1993	1HOUR	CALIBRATED
12	BALDEAGLE	BALDE	PRECIP-EXCESS	01DEC1993	1HOUR	CALIBRATED

Figure 2.8 List of HEC-DSS Pathnames by Parts

> ***Note:** The **No Pathnames** option is typically used to access very large databases using only scripted functions. The **Unsorted List** option can be used to view pathnames in very large databases without the time required to sort the pathnames.*

2.1.7 Selected Pathnames

When you select a pathname (see Section 2.6), it appears beneath the list of all pathnames in the **Selection Area** (Figure 2.9). It includes the name of the DSS file followed by the full DSS pathname. You can change the height of the **Selection Area** by grabbing the separator line between the two lists with your mouse, and moving it up or down. Your mouse pointer will change to a double ended arrow when you are over the separator line, indicating you can slide the line up or down.



BaldEagleHist.dss:/BALDEAGLE/BALDE/FLOW-DIRECT/01NOV1993/1HOUR/CALIBRATED/
BaldEagleHist.dss:/BALDEAGLE/BALDE/FLOW-DIRECT/01NOV1993/1HOUR/SB/

Figure 2.9 Selected Pathnames

You can also **Quick Select** pathnames if the **Selection Area** is empty by just highlighting pathnames from the main list with your mouse, and then selecting a function such as plot or tabulate. The **Selection Area** must be empty to use **Quick Select**.

2.1.8 Selection Buttons

There are five buttons located below the **Selection Area** (Figure 2.10). They are as follows:

Select You add a pathname to the selection list by highlighting it and then clicking the **Select**

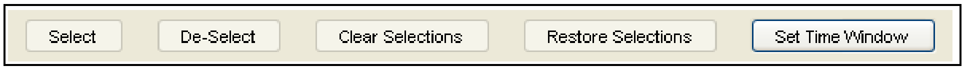


Figure 2.10 Select, De-Select, Clear Selections, Restore Selections, and Set Time Window Buttons

button. The **Select** button will remain inactive until a pathname is highlighted.

> ***Tip:*** *You can also select pathnames by double-clicking on a pathname in the list or clicking on a pathname to highlight it, then click and hold the right mouse key and drag the pathname to the **Selection Area**.*

De-Select To remove a pathname from the **Selection Area** list, highlight the pathname in the **Selection Area** then click the **De-Select** button.

> ***Tip:*** *You can de-select by double-clicking on a pathname in the **Selection Area** list.*

Clear Selections You can remove all pathnames from the **Selection Area** list by clicking the **Clear Selections** button.

Restore Selections The **Restore Selections** button restores all selections you have cleared or de-selected from the **Selection Area** list.

Set Time Window You can set the time window of the data to view with the **Set Time Window** button.

2.2 Message Bar

The **Message Bar** (Figure 2.11) appears at the bottom of the DSSVue main window and displays the current status. Messages displayed in the bar include information on current processes or information regarding settings within the current DSSVue main window, such as the Time Window. For example, if the Time Window is set, the Time Window range will be displayed or if you are copying a large amount of data, the

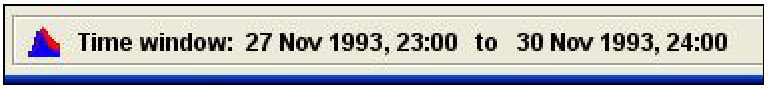
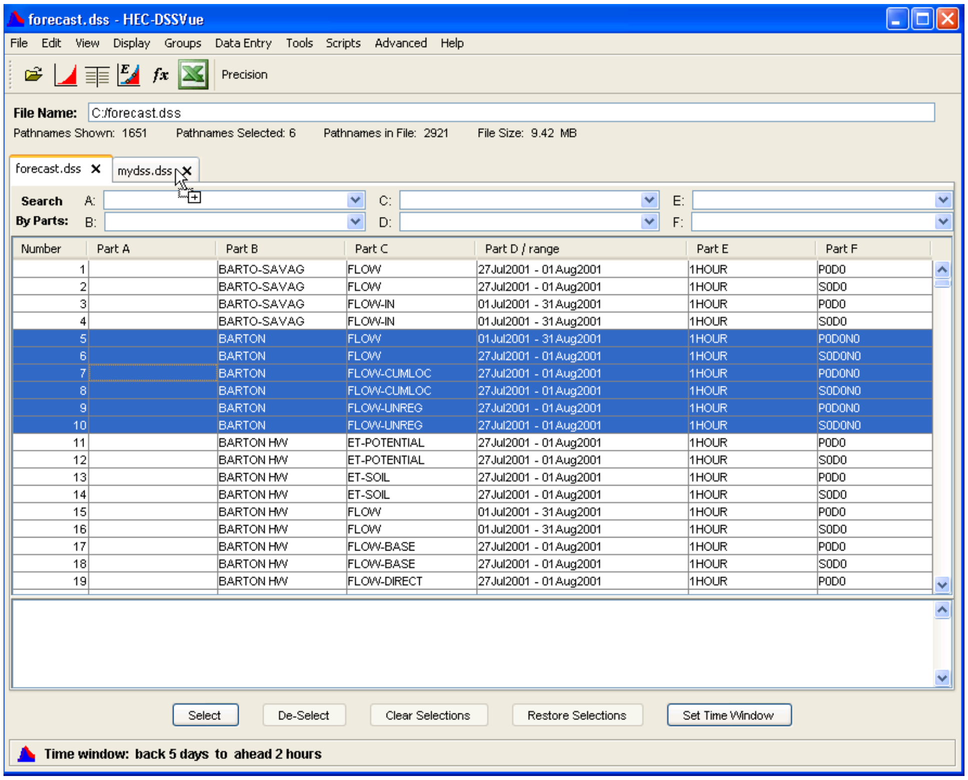

Time window: 27 Nov 1993, 23:00 to 30 Nov 1993, 24:00

Figure 2.11 Message Bar

status of the copy will be displayed in the **Message Bar** during the copy process.

2.3 Drag and Drop

HEC-DSSVue supports a variety of Drag and Drop operations. You can drag pathnames from the main screen to any open table or plot to add those data sets to the table or curve. You can copy data from one DSS file to another by dragging the pathnames of the data sets you want to copy to the tab of the file you want to copy to, or to the main window of another instance of HEC-DSSVue. An example is shown in Figure 2.12.


forecast.dss - HEC-DSSVue
File Edit View Display Groups Data Entry Tools Scripts Advanced Help
Precision
forecast.dss ✕
mydss.dss ✕
Search By Parts: A: B: C: D: E: F:
Select
De-Select
Clear Selections
Restore Selections
Set Time Window

File Name: C:/forecast.dss

Pathnames Shown: 1651 Pathnames Selected: 6 Pathnames in File: 2921 File Size: 9.42 MB

Number	Part A	Part B	Part C	Part D / range	Part E	Part F
1		BARTO-SAVAG	FLOW	27Jul2001 - 01Aug2001	1HOUR	P0D0
2		BARTO-SAVAG	FLOW	27Jul2001 - 01Aug2001	1HOUR	S0D0
3		BARTO-SAVAG	FLOW-IN	01Jul2001 - 31Aug2001	1HOUR	P0D0
4		BARTO-SAVAG	FLOW-IN	01Jul2001 - 31Aug2001	1HOUR	S0D0
5		BARTON	FLOW	01Jul2001 - 31Aug2001	1HOUR	P0D0N0
6		BARTON	FLOW	27Jul2001 - 01Aug2001	1HOUR	S0D0N0
7		BARTON	FLOW-CUMLOC	27Jul2001 - 01Aug2001	1HOUR	P0D0N0
8		BARTON	FLOW-CUMLOC	27Jul2001 - 01Aug2001	1HOUR	S0D0N0
9		BARTON	FLOW-UNREG	27Jul2001 - 01Aug2001	1HOUR	P0D0N0
10		BARTON	FLOW-UNREG	27Jul2001 - 01Aug2001	1HOUR	S0D0N0
11		BARTON HW	ET-POTENTIAL	27Jul2001 - 01Aug2001	1HOUR	P0D0
12		BARTON HW	ET-POTENTIAL	27Jul2001 - 01Aug2001	1HOUR	S0D0
13		BARTON HW	ET-SOIL	27Jul2001 - 01Aug2001	1HOUR	P0D0
14		BARTON HW	ET-SOIL	27Jul2001 - 01Aug2001	1HOUR	S0D0
15		BARTON HW	FLOW	01Jul2001 - 31Aug2001	1HOUR	P0D0
16		BARTON HW	FLOW	01Jul2001 - 31Aug2001	1HOUR	S0D0
17		BARTON HW	FLOW-BASE	27Jul2001 - 01Aug2001	1HOUR	P0D0
18		BARTON HW	FLOW-BASE	27Jul2001 - 01Aug2001	1HOUR	S0D0
19		BARTON HW	FLOW-DIRECT	27Jul2001 - 01Aug2001	1HOUR	P0D0

Time window: back 5 days to ahead 2 hours

Figure 2.12 Copying data from one DSS file to another using Drag and Drop

You can import data into a DSS file by dragging the data or files from Windows Explorer onto the HEC-DSSVue screen. If you have a plug-in loaded that recognizes the file extension, that data will automatically be imported. For example, you can drag a ".shef" file or ".ncdc" file to import shef data or import climate data from the National Climatic Data Center.

You can drag regular files onto HEC-DSSVue and they will be stored in the FILE convention in that DSS file. For example, you can drag a .pdf file or .xls file onto HEC-DSSVue to be stored in the DSS file. When you plot or tabulate those data sets, HEC-DSSVue will launch the native application for that data set.

You can also drag pictures (e.g., "*.jpg*") files on to the main screen to store them, and then view them back as a slide show.

2.4 File Menu Operations

HEC-DSSVue's **File** menu allows you to create a **New** HEC-DSS file, **Open** an existing HEC-DSS file, **Close DSS File(s)** you have opened, show the **Print Catalog Preview**, **Print Catalog**, open a recently viewed file, and **Exit** the program. If you are running HEC-DSSVue in a client-server mode, you can **Open a Remote** file or create a **New Remote** file. The following sections discuss these operations in detail.

2.4.1 Creating a New HEC-DSS File

To create a new HEC-DSS database file, from the **File** menu, click **New**. When the **Create new HEC-DSS File** dialog box opens (Figure 2.13), select where you would like to create the file using the **Look in** list or file icons. Type in a name for the new file in the **File name** box and then click

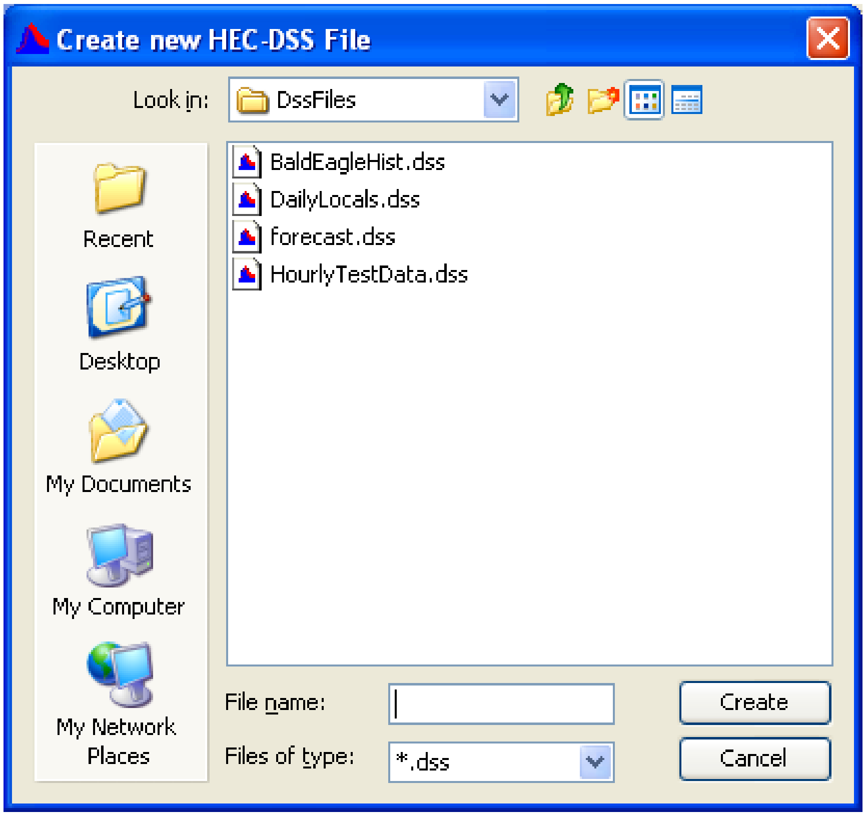


Figure 2.13 Create new HEC-DSS File Dialog Box

the **Create** button. A new HEC-DSSVue file will be created in the location you selected. Data can then be added using the tools located under the **Data Entry** menu, copied in through the many different plug-ins available, or copied from other DSS files*.

> ***Note**: If you copy records from an existing file, a new DSS file will be created automatically if one does not already exist.*

2.4.2 Opening an HEC-DSS File

To open a DSS file, from the **File** menu, click **Open**. The **Open HEC-DSS File** browser will display, shown in Figure 2.14.

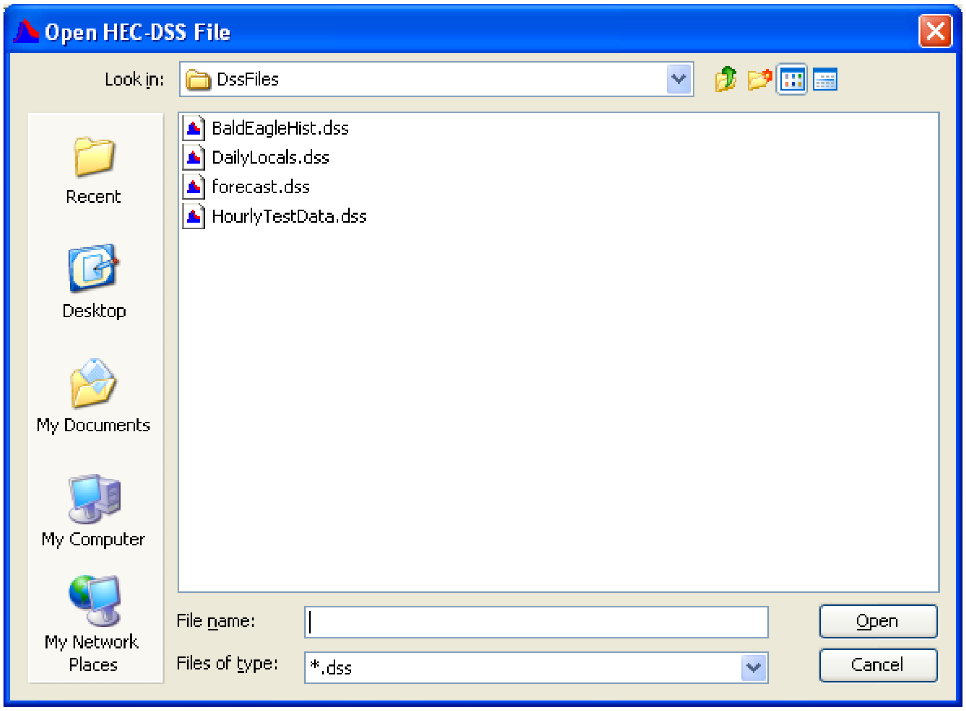


Figure 2.14 Open HEC-DSS File Browser

> ***Note:** If you know the name of the HEC-DSS database file you wish to open, you can type the file name (including the path) directly into the **File Name** box on the main DSSVue window (see Figure 2.1, page 2-1).*

From the **Open HEC-DSS File** browser, use the standard Windows controls to browse to the HEC-DSS file that you wish to open, click **Open**.

The name of the file you have selected will now appear in the **File name** box (see Figure 2.1, page 2-1), and the pathnames of the records contained in the file will display in the **List of HEC-DSS Pathnames**.

2.4.3 Closing an HEC-DSS File

To close the currently selected HEC-DSS database file, from the **File** menu, click **Close**. This will allow some older MS-DOS based programs to access the file or for you to rename or delete the file. HEC-DSS will automatically close un-accessed file files after about a minute. They will be automatically re-opened on the next access.

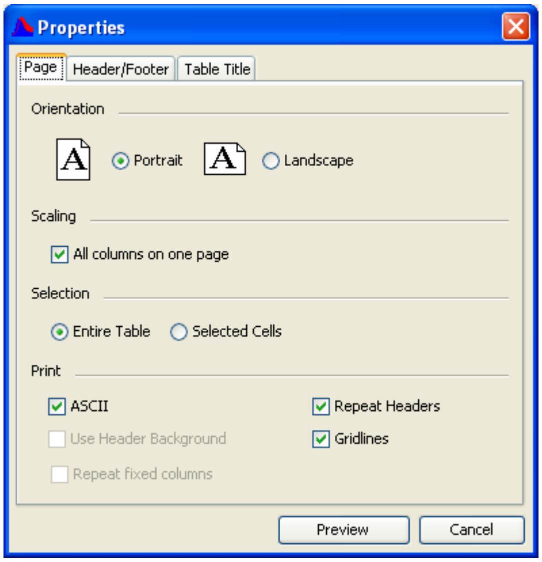
> ***Note:*** *HEC-DSS is a multi-user database system. Some older MS-DOS programs exclusively locked HEC-DSS files. HEC-DSS will automatically close un-accessed file files after about a minute. They will be automatically re-opened on the next access.*

2.4.4 Printing the Catalog

You can print the catalog of a DSS file directly (**Print Catalog**) or you can preview what the catalog will look like before you print (**Print Catalog Preview**). For further printing of DSS catalogs, see Chapter 4.

A catalog for the DSS file consists of a list of the pathnames contained in the file, similar to what is displayed in the **DSS Pathname** list area. From the **View** menu the user can determine how the catalog will display the pathnames. For example, the catalog could be displayed as a sorted list or by pathname parts. The catalog for the currently selected DSS file can be printed or viewed in a separate window.

To preview the catalog in a separate window and then print:



Orientation

● Portrait ○ Landscape

Scaling

☑ All columns on one page

Selection

● Entire Table ○ Selected Cells

Print

☑ ASCII ☑ Repeat Headers

□ Use Header Background ☑ Gridlines

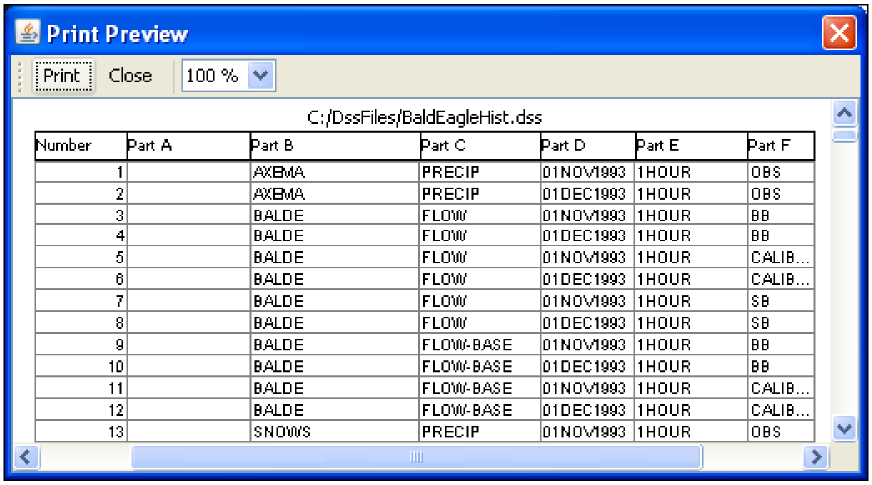
□ Repeat fixed columns

Figure 2.15 Properties Dialog Box

1. From the **File** menu, click **Print Catalog Preview**, a **Properties** dialog box will open (Figure 2.15) that is specific for setting up a report for printing. The **Properties** dialog box offers various print options, on three separate tabs.
2. The **Page** tab allows you to specify the page **Orientation, Scaling**, and **Selection**; you can also choose to print the

table as **ASCII, Repeat Headers** on every page, and print the **Gridlines**.

3. On the **Header/Footer** tab, you can type in the header and footer you want to appear on your printed pages.
4. The **Table Title** tab offers a default title for the table based on the data source. You may edit this title.
5. Once the user has the report setup, click **Preview**, the **Print Preview** dialog box will open (Figure 2.16).



C:/DssFiles/BaldEagleHist.dss

Number	Part A	Part B	Part C	Part D	Part E	Part F
1		AXBMA	PRECIP	01NOV1993	1HOUR	OBS
2		AXBMA	PRECIP	01DEC1993	1HOUR	OBS
3		BALDE	FLOW	01NOV1993	1HOUR	BB
4		BALDE	FLOW	01DEC1993	1HOUR	BB
5		BALDE	FLOW	01NOV1993	1HOUR	CALIB...
6		BALDE	FLOW	01DEC1993	1HOUR	CALIB...
7		BALDE	FLOW	01NOV1993	1HOUR	SB
8		BALDE	FLOW	01DEC1993	1HOUR	SB
9		BALDE	FLOW-BASE	01NOV1993	1HOUR	BB
10		BALDE	FLOW-BASE	01DEC1993	1HOUR	BB
11		BALDE	FLOW-BASE	01NOV1993	1HOUR	CALIB...
12		BALDE	FLOW-BASE	01DEC1993	1HOUR	CALIB...
13		SNOWS	PRECIP	01NOV1993	1HOUR	OBS

Figure 2.16 Print Preview Dialog Box

6. To print the catalog, click **Print**, a **Print** dialog box will open, click **OK**, and the report will print to the selected printer.

To preview the catalog in a separate window and then print:

1. From the **File** menu, click **Print Catalog**, a **Properties** dialog box will open (see Figure 2.15, page 2-11) that is specific for setting up a report for printing. The **Properties** dialog box offers various print options, on three separate tabs.
2. From above, review Steps 2 thru 4 for available print options.
3. Click **Print,** a **Print** dialog box will open, click **OK**, and the report will print to the selected printer.

2.4.5 Exiting HEC-DSSVue

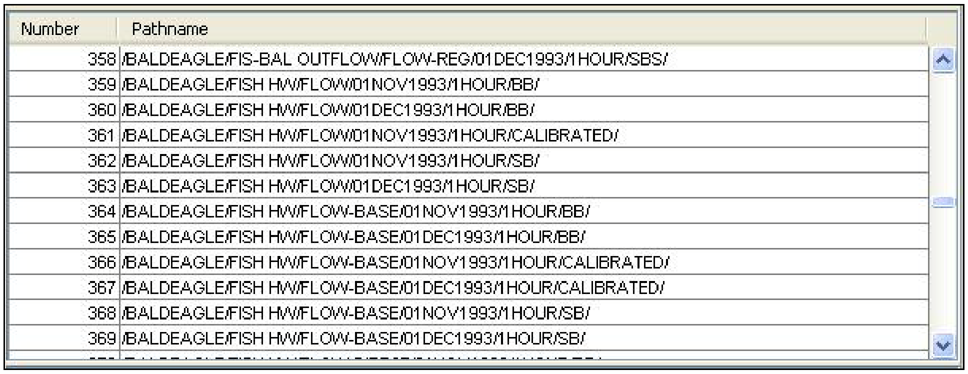
To exit HEC-DSSVue, from the **File** menu, click **Exit**. (If you have opened HEC-DSSVue from another application, such as CWMS or ResSim, click **Close** from HEC-DSSVue's **File** menu.)

2.5 View Catalog

The **View** menu allows you to change the appearance the catalog that is associated with a DSS file.

2.5.1 Choosing a Display Mode for HEC-DSS Pathnames

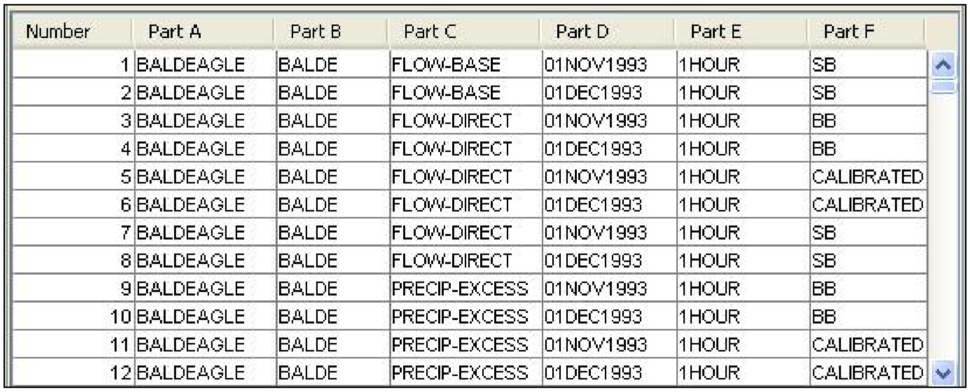
To specify the display mode for HEC-DSS Pathnames, choose **Pathname List, Pathname Parts, Condensed Catalog**, or **No Pathn**ames or **Unsorted List** from the **View** menu. A **Pathname** List of HEC-DSS files displayed is shown in Figure 2.17.



Number	Pathname
358	/BALDEAGLE/FIS-BAL OUTFLOW/FLOW-REG/01DEC1993/1HOUR/SBS/
359	/BALDEAGLE/FISH HW/FLOW/01NOV1993/1HOUR/BB/
360	/BALDEAGLE/FISH HW/FLOW/01DEC1993/1HOUR/BB/
361	/BALDEAGLE/FISH HW/FLOW/01NOV1993/1HOUR/CALIBRATED/
362	/BALDEAGLE/FISH HW/FLOW/01NOV1993/1HOUR/SB/
363	/BALDEAGLE/FISH HW/FLOW/01DEC1993/1HOUR/SB/
364	/BALDEAGLE/FISH HW/FLOW-BASE/01NOV1993/1HOUR/BB/
365	/BALDEAGLE/FISH HW/FLOW-BASE/01DEC1993/1HOUR/BB/
366	/BALDEAGLE/FISH HW/FLOW-BASE/01NOV1993/1HOUR/CALIBRATED/
367	/BALDEAGLE/FISH HW/FLOW-BASE/01DEC1993/1HOUR/CALIBRATED/
368	/BALDEAGLE/FISH HW/FLOW-BASE/01NOV1993/1HOUR/SB/
369	/BALDEAGLE/FISH HW/FLOW-BASE/01DEC1993/1HOUR/SB/

Figure 2.17 HEC-DSSVue - Data Selection List window, Pathname List displayed

You can also view the pathnames as a **Pathname Parts** List (Figure 2.18).

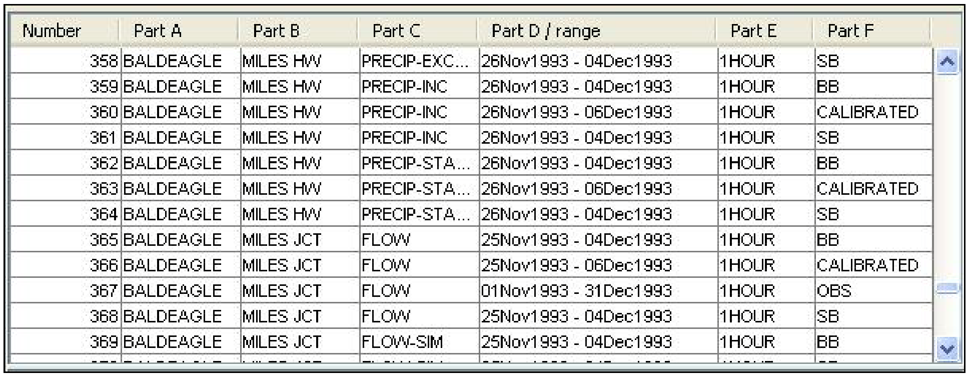


Number	Part A	Part B	Part C	Part D	Part E	Part F
1	BALDEAGLE	BALDE	FLOW-BASE	01NOV1993	1HOUR	SB
2	BALDEAGLE	BALDE	FLOW-BASE	01DEC1993	1HOUR	SB
3	BALDEAGLE	BALDE	FLOW-DIRECT	01NOV1993	1HOUR	BB
4	BALDEAGLE	BALDE	FLOW-DIRECT	01DEC1993	1HOUR	BB
5	BALDEAGLE	BALDE	FLOW-DIRECT	01NOV1993	1HOUR	CALIBRATED
6	BALDEAGLE	BALDE	FLOW-DIRECT	01DEC1993	1HOUR	CALIBRATED
7	BALDEAGLE	BALDE	FLOW-DIRECT	01NOV1993	1HOUR	SB
8	BALDEAGLE	BALDE	FLOW-DIRECT	01DEC1993	1HOUR	SB
9	BALDEAGLE	BALDE	PRECIP-EXCESS	01NOV1993	1HOUR	BB
10	BALDEAGLE	BALDE	PRECIP-EXCESS	01DEC1993	1HOUR	BB
11	BALDEAGLE	BALDE	PRECIP-EXCESS	01NOV1993	1HOUR	CALIBRATED
12	BALDEAGLE	BALDE	PRECIP-EXCESS	01DEC1993	1HOUR	CALIBRATED

Figure 2.18 HEC-DSSVue - Data Selection List window, Pathname Parts displayed

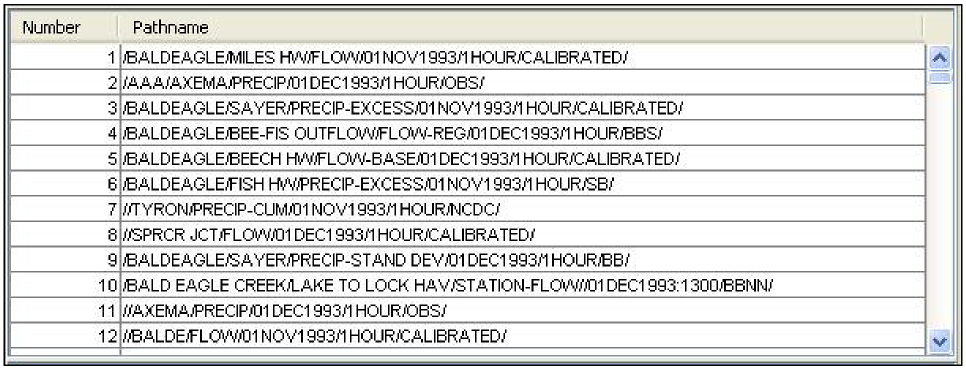
The **Condensed Catalog** style (see Figure 2.19, page 2-14) abridges time series data sets so that the date span for the entire data set displays in place of the "D" part.

The **Unsorted List** style (see Figure 2.20, page 2-14) displays the pathnames without sorting them. This is useful to see the pathnames in a very large database without using the overhead of sorting. This is also the fastest method to show the pathnames in a database.



Number	Part A	Part B	Part C	Part D / range	Part E	Part F
358	BALDEAGLE	MILES HW	PRECIP-EXC...	26Nov1993 - 04Dec1993	1HOUR	SB
359	BALDEAGLE	MILES HW	PRECIP-INC	26Nov1993 - 04Dec1993	1HOUR	BB
360	BALDEAGLE	MILES HW	PRECIP-INC	26Nov1993 - 06Dec1993	1HOUR	CALIBRATED
361	BALDEAGLE	MILES HW	PRECIP-INC	26Nov1993 - 04Dec1993	1HOUR	SB
362	BALDEAGLE	MILES HW	PRECIP-STA...	26Nov1993 - 04Dec1993	1HOUR	BB
363	BALDEAGLE	MILES HW	PRECIP-STA...	26Nov1993 - 06Dec1993	1HOUR	CALIBRATED
364	BALDEAGLE	MILES HW	PRECIP-STA...	26Nov1993 - 04Dec1993	1HOUR	SB
365	BALDEAGLE	MILES JCT	FLOW	25Nov1993 - 04Dec1993	1HOUR	BB
366	BALDEAGLE	MILES JCT	FLOW	25Nov1993 - 06Dec1993	1HOUR	CALIBRATED
367	BALDEAGLE	MILES JCT	FLOW	01Nov1993 - 31Dec1993	1HOUR	OBS
368	BALDEAGLE	MILES JCT	FLOW	25Nov1993 - 04Dec1993	1HOUR	SB
369	BALDEAGLE	MILES JCT	FLOW-SIM	25Nov1993 - 04Dec1993	1HOUR	BB

Figure 2.19 HEC-DSSVue - Data Selection List window, Condensed Catalog displayed



Number	Pathname
1	/BALDEAGLE/MILES HW/FLOW/01NOV1993/1HOUR/CALIBRATED/
2	/AAA/AXEMA/PRECIP/01DEC1993/1HOUR/OBS/
3	/BALDEAGLE/SAYER/PRECIP-EXCESS/01NOV1993/1HOUR/CALIBRATED/
4	/BALDEAGLE/BEE-FIS OUTFLOW/FLOW-REG/01DEC1993/1HOUR/BBS/
5	/BALDEAGLE/BEECH HW/FLOW-BASE/01DEC1993/1HOUR/CALIBRATED/
6	/BALDEAGLE/FISH HW/PRECIP-EXCESS/01NOV1993/1HOUR/SB/
7	//TYRON/PRECIP-CUM/01NOV1993/1HOUR/NCDC/
8	//SPRCR JCT/FLOW/01DEC1993/1HOUR/CALIBRATED/
9	/BALDEAGLE/SAYER/PRECIP-STAND DEV/01DEC1993/1HOUR/BB/
10	/BALD EAGLE CREEK/LAKE TO LOCK HAV/STATION-FLOW//01DEC1993:1300/BBNN/
11	//AXEMA/PRECIP/01DEC1993/1HOUR/OBS/
12	//BALDE/FLOW/01NOV1993/1HOUR/CALIBRATED/

Figure 2.20 HEC-DSSVue – Data Selection List window, Unsorted List displayed

2.5.2 Searching and Filtering HEC-DSS Pathnames

With HEC-DSSVue, you can select specific data sets from a list of pathnames (or catalog) in the database. You can refine the list by searching for either a string in the pathnames or for pathname parts. The following sections describe searching and filtering options.

Searching Pathname Strings

The **Search Pathnames by String** feature allows you to search for records whose pathnames contain specific strings of characters. For example, if you search for "FLOW" the list will display only pathnames containing the string "FLOW" in the pathname.

To **Search Pathnames by String**:

1. From the **View** menu, click **Search Pathnames by string**.
2. In the **Search Pathnames** box (see Figure 2.21, page 2-15), type the character string you want to filter the pathnames with.
3. Click **Search**.

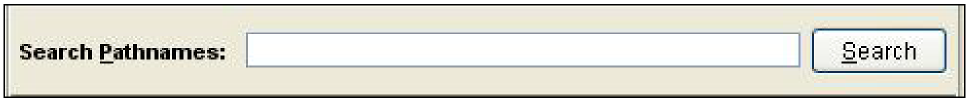


Figure 2.21 Search Pathnames Box

Filtering HEC-DSS Pathnames by Parts

The **Search by Parts** feature lets you choose specific pathname parts from those available in a HEC-DSS file.

To **Search by Parts**:

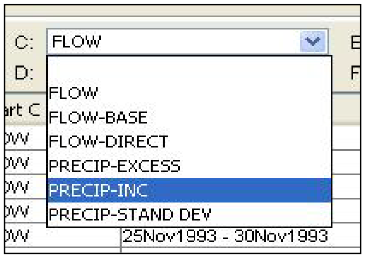


Figure 2.22 Selecting Pathname Parts

1. From the **View** menu, click **Search pathnames by parts**.
2. Click on the down arrows beside the pathname parts you want to search. This allows you to view lists of available names in the file (Figure 2.22).
3. When you have a pathname part list open, you can scroll through the list to locate the pathname part you want.
4. Click on a pathname part to select it.
5. To return to the previous list, click on the blank field at the top of the list.

2.5.3 Refreshing the Catalog

The catalog automatically refreshes whenever you add, delete, or rename records. For very large files or network files, a dialog box will be shown asking if you want to create a new catalog. When the catalog is refreshed, it re-inventories all records on disk to get a pathname for every record. To generate a new catalog manually, click **Refresh Catalog** from the **Edit** menu.

2.6 Selecting Pathnames

Once you have filtered a HEC-DSS file for the pathnames you want, you select specific pathnames to visualize data for or perform a utility function on. When you select a Pathname, it will appear in the **Selected Pathnames** List.

There are several ways to select pathnames:

- Double-click on an individual pathname in the HEC-DSS Pathname List.
- Highlight a pathname in the HEC-DSS Pathname List then click **Select** (Figure 2.10). Until you select a pathname, the **Select** button remains inactive.
- Click and drag your mouse to select a series of pathnames, and then click **Select**. You can also use **Ctrl**+click to select multiple, non-consecutive pathnames.
- Hold down the **Ctrl** key and click on pathnames to highlight them, or hold down on the shift key to highlight all pathnames between mouse clicks, and then click **Select**.
- If you wish to select all of the pathnames, click **Select All** from the **Edit** menu.
- If no pathnames are selected, you can just highlight one or more pathnames and then perform the function that you intended. If pathnames have already been added to the selection box, those pathnames will be used instead of the highlighted ones.

2.7 De-Selecting Pathnames

To de-select a pathname currently in the **Selected Pathnames** List:

1. Click the record's pathname in the **Selected Pathnames** List to select it.
2. Click **De-Select**, the pathname will no longer appear in the **Selected Pathnames** List.

You can also de-select pathnames by double-clicking on a pathname in the **Selection Area** list.

To remove all pathnames from the **Selected Pathnames** List, click **Clear Selections**. To restore selections you have cleared, use **Restore Selections**.

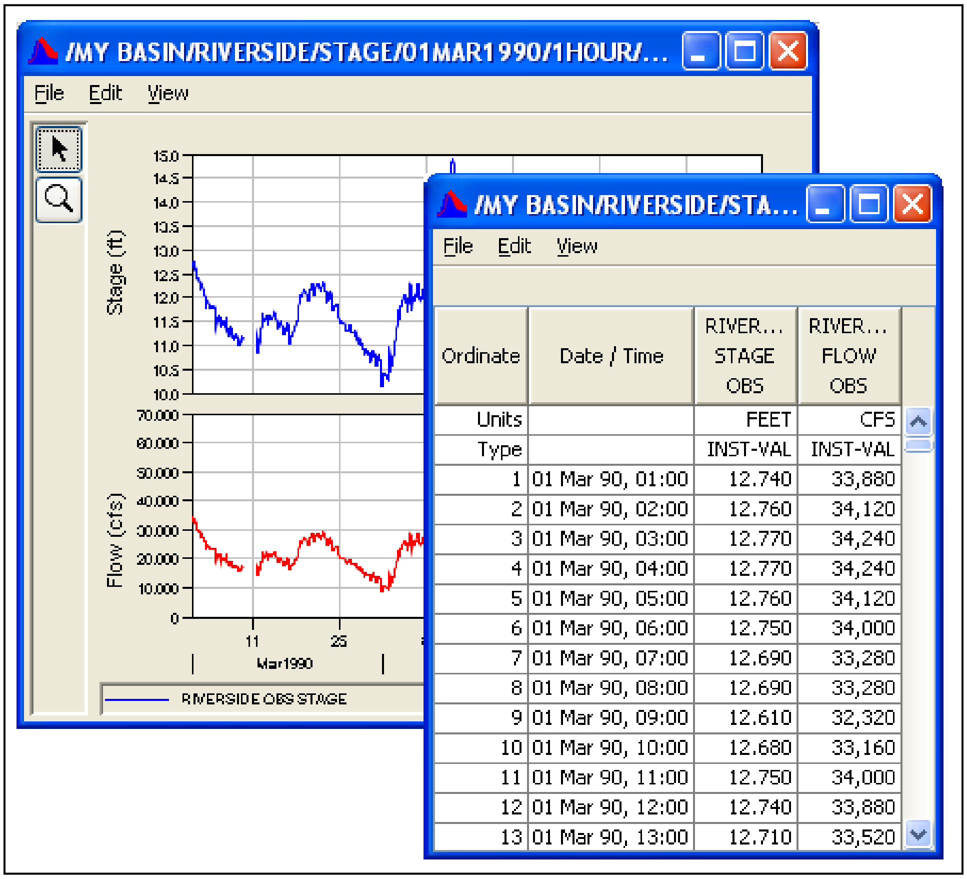
2.8 Visualizing HEC-DSS Data with Plots and Tables

Once the **Selected Pathnames** List contains all the pathnames you wish to display, you can generate plots and tables.

To create a plot or table from the **Display** menu, click **Plot** or **Tabulate**. You can also click the **Plot** button or the **Tabulate** button.

From a plot window, you can also tabulate data using the **Tabulate** command in the plot's **File** menu. Likewise, to open a plot from a table, you can use the **Plot** command in the table's **Fil**e menu.

Examples of the plots and tables HEC-DSSVue can produce are shown in Figure 2.23.



Ordinate	Date / Time	RIVER... STAGE OBS	RIVER... FLOW OBS
Units		FEET	CFS
Type		INST-VAL	INST-VAL
1	01 Mar 90, 01:00	12.740	33,880
2	01 Mar 90, 02:00	12.760	34,120
3	01 Mar 90, 03:00	12.770	34,240
4	01 Mar 90, 04:00	12.770	34,240
5	01 Mar 90, 05:00	12.760	34,120
6	01 Mar 90, 06:00	12.750	34,000
7	01 Mar 90, 07:00	12.690	33,280
8	01 Mar 90, 08:00	12.690	33,280
9	01 Mar 90, 09:00	12.610	32,320
10	01 Mar 90, 10:00	12.680	33,160
11	01 Mar 90, 11:00	12.750	34,000
12	01 Mar 90, 12:00	12.740	33,880
13	01 Mar 90, 13:00	12.710	33,520

Figure 2.23 Example Plot and Table

HEC-DSSVue plots are highly customizable. You can add titles, specify colors and patterns of backgrounds and lines, change line styles, add markers and callouts, add borders to labels, and customize legends, axes, and tic marks. For more information refer to Chapters 6.

2.9 Groups

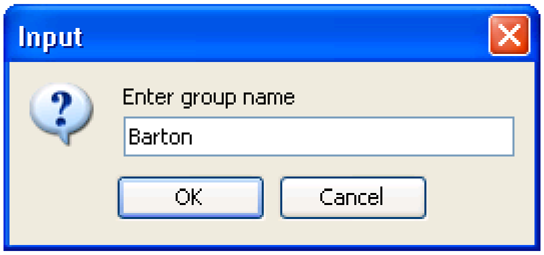
Data sets can be assembled together in Groups. Groups provide easy access of plotting, tabulating, and editing frequently accessed records without having to search for them in the DSS file. The **Groups** capability is often used when the same time series data sets with different time windows are accessed frequently. The Group information is stored in a text file in the User's Application Data area, under HEC-DSSSVue.

The **Groups** menu option gives you the ability to create groups and access previously created groups. The following sections describe the **Groups** menu options in detail.

2.9.1 Creating and Saving Groups

To create a **Group**:

1. Select all the records you would like in the group in the **Selected Area**. If you set a time window, that will be saved along with the group
2. From the **Groups** menu, click **Save Selected**. The **Input** dialog will appear (Figure 2.24). Type the group's name in the **Enter group name** box and click **OK**.



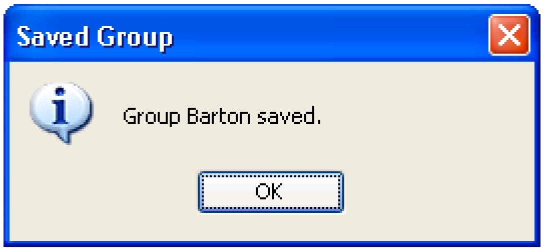
Input

Enter group name

Barton

Figure 2.24 Input Dialog Box

3. The **Saved Group** message (Figure 2.25) will appear when the g has been saved. Click **OK**.



Saved Group

Group Barton saved.

Figure 2.25 Saved Group Message

4. The group will be saved and listed in the **Groups** submenus, see Figure 2.26.

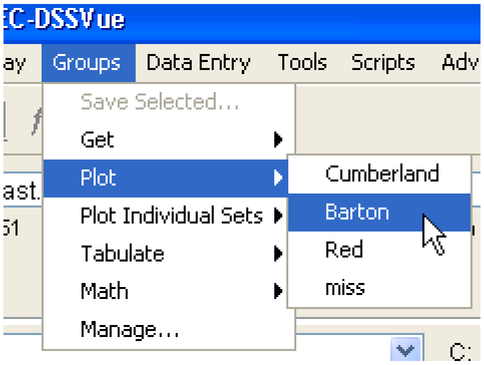


Figure 2.26 Group Submenus

2.9.2 Selecting Groups

When a **Group** is selected, all the records contained in the group are added to the **Selected Pathnames** List. To select a previously created **Group**, from the **Groups** menu, point to **Get**, select the name of the **Group**.

2.9.3 Plotting Groups

To plot a group, from the **Groups** menu, point to **Plot**, select the name of the group. A plot window will appear plotting all the records contained in the group as shown in Figure 2.27 below.

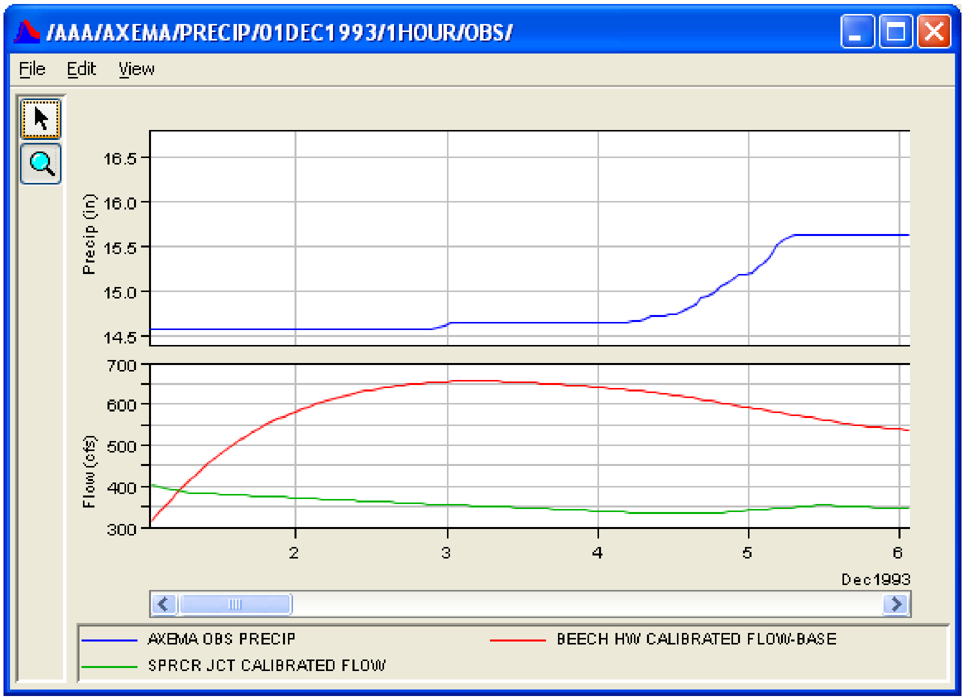


Figure 2.27 Group Plot

If you would like to plot a group, but only display one record at a time in the plot, from the **Groups** menu, point to **Plot Individual Sets**, then the name of the group. A plot window will appear that has a **Select Data Set** list (see Figure 2.28, page 2-20), listing all the records in the group. The first record in the group will be selected in the **Select Data Set** list and the records data will be plotted. To plot a different record in the group, simply select the record in the **Select Data Set** list and the current data in the plot will be replaced with the newly selected data.

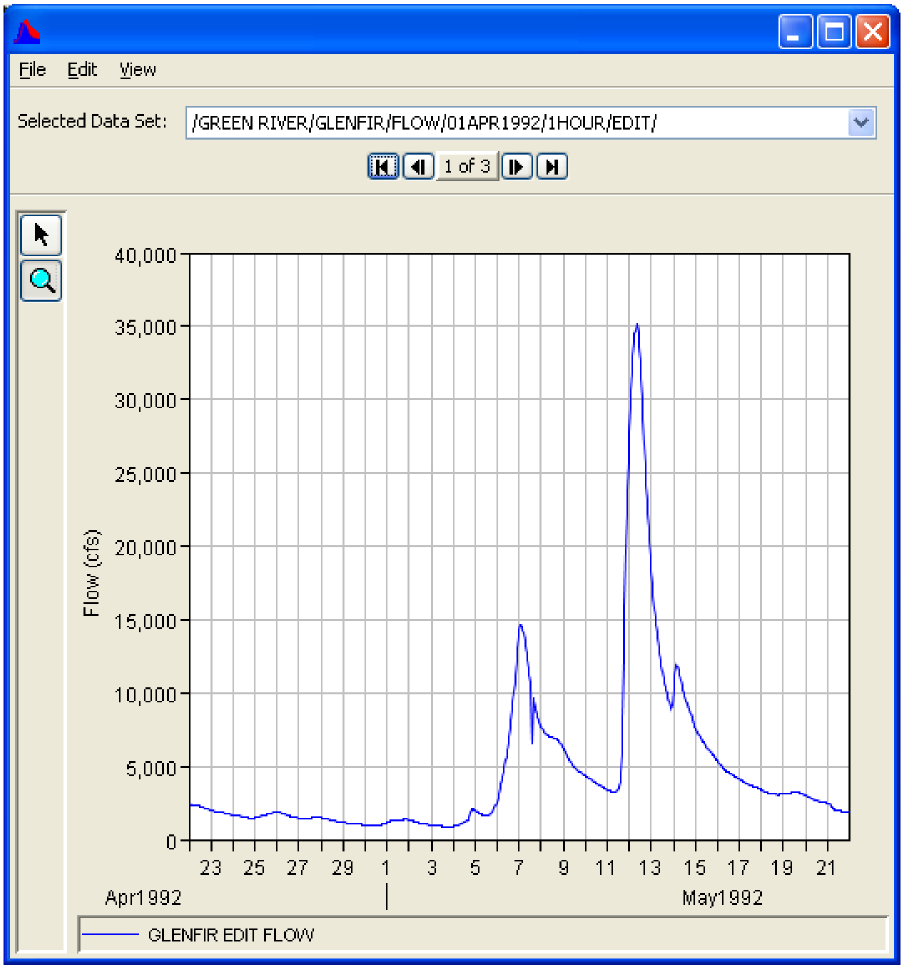


Figure 2.28 Individual Data Set Plot

2.9.4 Tabulate Groups

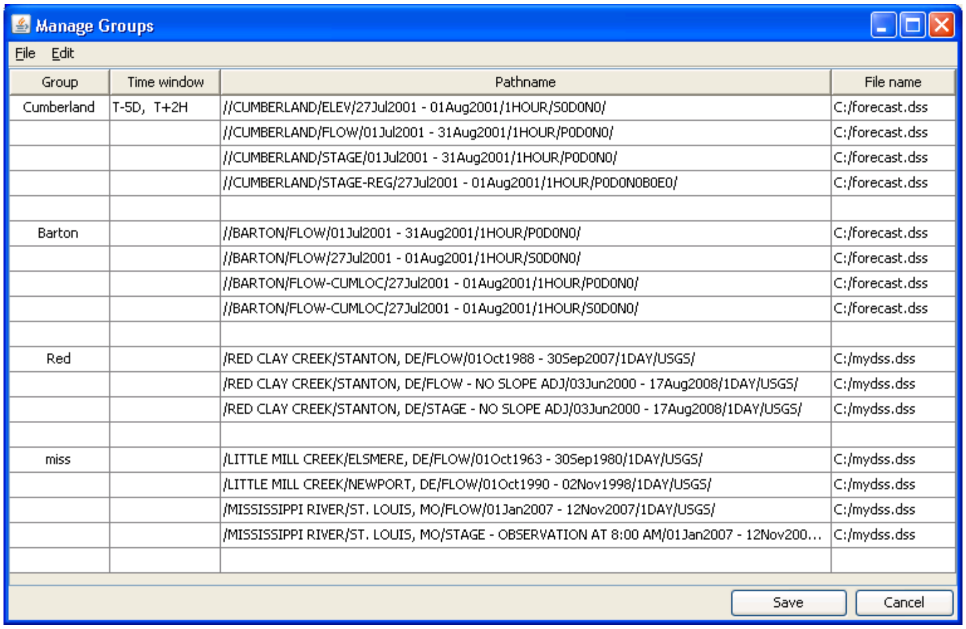
To tabulate a group, from the **Groups** menu, point to **Tabulate**, select the name of the group. A tabulate window will appear tabulating all the records contained in the group.

2.9.5 Math Groups

You can open the **Math Functions Editor** for a particular group of pathnames (data set) to perform math operations on the data sets within the group. For more detailed information on the **Math Functions Editor**, see Chapter 7. To use the **Math Functions Editor**, from the **Groups** menu, point to **Math**, select the name of the group. The **Math Functions Editor** will open with the group's pathnames populating the **Selected Data Set** list.

2.9.6 Manage Groups

Groups can be edited in the **Manage Groups Editor** (Figure 2.29). In the **Manage Groups Editor** you can delete groups, add pathnames to a group, change the order of a group's pathnames, remove pathnames, and rename groups. To access this editor, from the **Groups** menu, click **Manage,** the **Manage Groups Editor** will open.



Group	Time window	Pathname	File name
Cumberland	T-5D, T+2H	//CUMBERLAND/ELEV/27Jul2001 - 01Aug2001/1HOUR/S0D0N0/	C:/forecast.dss
		//CUMBERLAND/FLOW/01Jul2001 - 31Aug2001/1HOUR/P0D0N0/	C:/forecast.dss
		//CUMBERLAND/STAGE/01Jul2001 - 31Aug2001/1HOUR/P0D0N0/	C:/forecast.dss
		//CUMBERLAND/STAGE-REG/27Jul2001 - 01Aug2001/1HOUR/P0D0N0B0E0/	C:/forecast.dss
Barton		//BARTON/FLOW/01Jul2001 - 31Aug2001/1HOUR/P0D0N0/	C:/forecast.dss
		//BARTON/FLOW/27Jul2001 - 01Aug2001/1HOUR/S0D0N0/	C:/forecast.dss
		//BARTON/FLOW-CUMLOC/27Jul2001 - 01Aug2001/1HOUR/P0D0N0/	C:/forecast.dss
		//BARTON/FLOW-CUMLOC/27Jul2001 - 01Aug2001/1HOUR/S0D0N0/	C:/forecast.dss
Red		/RED CLAY CREEK/STANTON, DE/FLOW/01Oct1988 - 30Sep2007/1DAY/USGS/	C:/mydss.dss
		/RED CLAY CREEK/STANTON, DE/FLOW - NO SLOPE ADJ/03Jun2000 - 17Aug2008/1DAY/USGS/	C:/mydss.dss
		/RED CLAY CREEK/STANTON, DE/STAGE - NO SLOPE ADJ/03Jun2000 - 17Aug2008/1DAY/USGS/	C:/mydss.dss
miss		/LITTLE MILL CREEK/ELSMERE, DE/FLOW/01Oct1963 - 30Sep1980/1DAY/USGS/	C:/mydss.dss
		/LITTLE MILL CREEK/NEWPORT, DE/FLOW/01Oct1990 - 02Nov1998/1DAY/USGS/	C:/mydss.dss
		/MISSISSIPPI RIVER/ST. LOUIS, MO/FLOW/01Jan2007 - 12Nov2007/1DAY/USGS/	C:/mydss.dss
		/MISSISSIPPI RIVER/ST. LOUIS, MO/STAGE - OBSERVATION AT 8:00 AM/01Jan2007 - 12Nov200...	C:/mydss.dss

Figure 2.29 Manage Groups Editor

2.10 Data Entry

The **Data Entry** menu provides tools to add data to HEC-DSS files. Commands available through this menu allow you to manually create and add time series, paired and text data, and **Import** or **Export** data sets to and from an HEC-DSS file.

2.11 Utilities

The **Edit** menu provides a means for renaming, deleting, duplicating and copying data sets. Data sets that have been deleted before the HEC-DSS file is squeezed can be recovered from the **Un-delete** dialog in the **Edit** menu. Two HEC-DSS files can be merged into one using the **Merge** feature.

The **Tools** menu facilitates additional tools to help manipulate and manage the data contained in HEC-DSS files. These tools include: **Math Functions, Data and File Comparing, Search, Integrity Check**, and

Scripting. You can also remove excess disk space and optimize the file with **File Squeeze**.

The **Advanced** menu provides access to the traditional HEC-DSS catalogs. From here you can generate, view, and print an abbreviated catalog, a condensed catalog or a complete catalog. You can also view the output log, reset the program defaults, and set HEC-DSS options, as well as several internal tools.

2.12 Custom and Scripting Menus

Menus can be added to HEC-DSSVue's main menu for plug-ins and scripting. The **Script Editor** has a checkbox to activate the **Script** menu and add the script you are editing to that menu. This is a convenient way to access a large number of scripts that are used frequently.

If you have several customized operations that are preformed through plug-ins, you can have those functions show up on a menu that you name.

2.13 Plug-Ins

HEC-DSSVue has the capability of accepting Java jar plug-ins. A plug-in is essentially compiled Java code that has been written to extend the capability of HEC-DSSVue. It is similar to a script, but can be more powerful and more customizable. For example, a plug-in might retrieve data, decode it and store it into a HEC-DSS file. Another use might be to perform sophisticated math functions on data selected from the main screen.

Plug-ins are added by placing the jar plug-in file in the HecDssVue/*Plugins* directory. When HEC-DSSVue is executed, it searches this directory for available plug-ins and loads them into the program. Most plug-ins will add a button to the main toolbar, adjacent to any script buttons. To remove a plug-in, simply remove its jar file from the *Plugins* directory.

CHAPTER 3

Displaying & Editing Data

HEC-DSSVue allows you to plot, tabulate, and edit data in HEC-DSS files. This chapter covers setting a time window to access and view time series data with, tabulating time series and paired data, editing data and displaying supplemental data associated with a data set. Plotting is covered in the chapter "Customizing Plots".

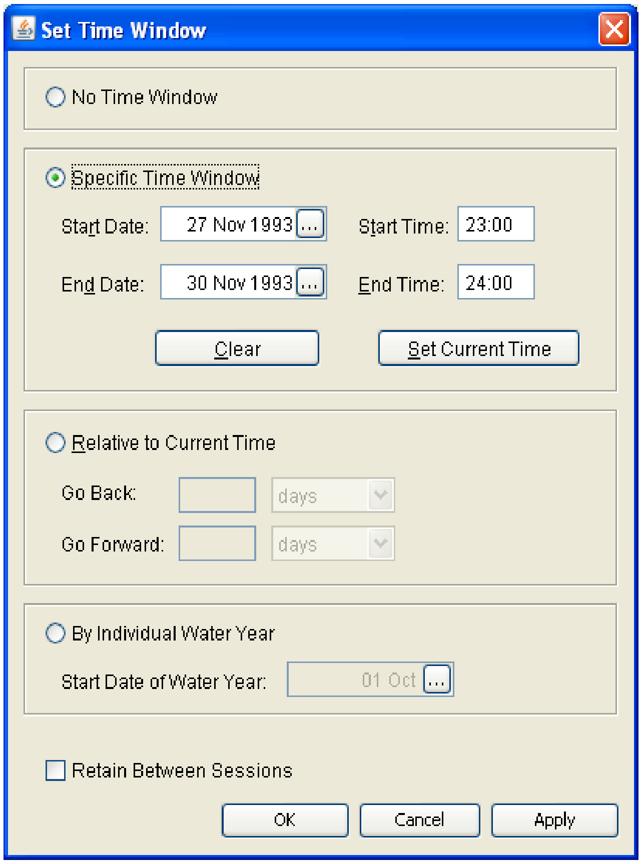
For HEC-DSSVue Version 2.0, you can "drag" data sets from the main window onto a plot or table to add that data set to the plot or table. The file that you are dragging from does not need to be the same as where the original data came from; in fact you can drag a data set from a different instance of HEC-DSSVue.

3.1 Setting a Time Window

You can specify the time window for data you want displayed in plots and tables and to perform math functions on. For example, if you have a set of HEC-DSS records ranging from October 1, 1993, to December 30, 1993, you can specify a narrower time window such as 05OCT1993 to 25OCT1993 for plots, tables, and operations with math functions.

To set the Time Window:

1. From the **Display** menu, click **Time Window**. The **Set Time Window** dialog box will open (see Figure 3.1, page 3-2).
2. You may select no time window, which will retrieve all the data for a data set, enter specific dates and times, use a time window relative to the current time, or access data by water year.
3. To access all data in a data set, select **No Time Window** (see Figure 3.1, page 3-2).
4. To set a specific time window, select **Specific Time Window**.
    a. If you wish, you can set the fields to the current time by clicking **Set Current Time** and then change what you want manually.
    b. To specify dates and times:
        i. Enter the **Start Date** and **End Date** (see Figure 3.1, page 3-1). Although a variety of formats will be recognized for the date, the typical format is DDMMMYYYY.



Set Time Window

○ No Time Window

● Specific Time Window

Start Date: 27 Nov 1993 Start Time: 23:00

End Date: 30 Nov 1993 End Time: 24:00

○ Relative to Current Time

Go Back: days

Go Forward: days

○ By Individual Water Year

Start Date of Water Year: 01 Oct

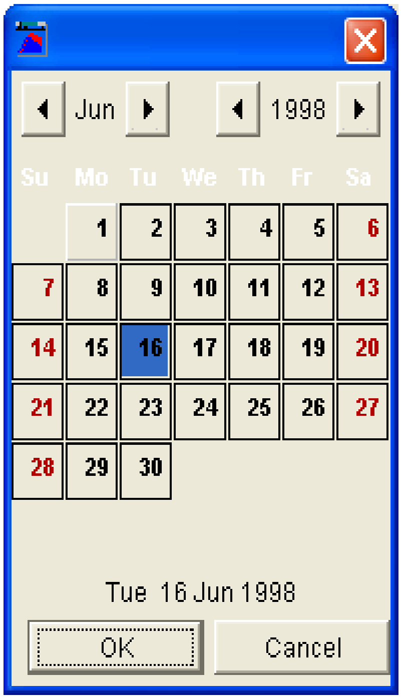
☐ Retain Between Sessions

Figure 3.1 Set Time Window Dialog Box

ii. Enter the **Start Time** and **End Time** in 24-hour format (e.g., for 2:00 PM, type "1400").

c. You may select the starting and ending dates from the calendar tool, From the **Start Date** box, click on […]. The calendar tool in a monthly format will open (Figure 3.2). You may scroll through months or years by using the appropriate arrow key at the top of the box.

5. To set a time window relative to the current time, select **Relative to Current Time** (Figure 3.1).
   a. Select the units you want to go back and forward from the current time. You can select **hours, days, months**, or **years**.
   b. Enter the **Go Back** and **Go Forward** units.
   c. If you do not set a unit, **No Time Window** will be set.



Jun 1998

Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

Tue 16 Jun 1998

Figure 3.2 Calendar Tool

6. Accessing DSS data by water year, from the **Set Time Window** dialog box select (see Figure 3.1, page 3-2) **Individual Water Year**. You can change the default start date of the water year from October first by setting a new date in the **Start Date of Water Year** box. This will read all the data in a data set and then separate it into individual data sets by water year. This setting is primarily useful when using math functions in HEC-DSSVue. If you plot data when this setting is selected, you should choose **Plot Individual Data Sets**.
7. To keep your settings for the next time you run HEC-DSSVue, select **Retain Between Sessions**. The next time you execute HEC-DSSVue, your time window will be set.
8. Click **OK** the **Set Time Window** dialog box will close (see Figure 3.1, page 3-2), and the time window settings will be applied.

3.2 Display Data Options

Two options are provided as a quick way to compare time series data sets for different time periods or for different amounts. The options are turned on or off by selecting and un-selecting them from the **Display** menu, point to **Display Data Options**, and the two available options are - **Normalize** and **Synch data set times to first**.

Normalize will subtract the differences between the values of first and the second or subsequent data sets from the corresponding data set, so that the curves all start at the same value (which is the value of the first data point in the first data set). This is often used to compare cumulative precipitation curves, which usually start at different values.

Synch data set times to first will subtract the differences between the times of first and the second or subsequent data sets from the corresponding data set, so that the curves all start at the same time (which is the time in the first data set). This is often used to directly compare hydrographs for different years or time periods.

Each option will remain on until you turn it off or the program exits. The options can be used in combination with each other. For example, if you want to compare cumulative precipitation curves for different years, set each option to on. Figure 3.3 shows such an example of normalized cumulative precipitation for years 1986 and 1995, synched to the same start time.

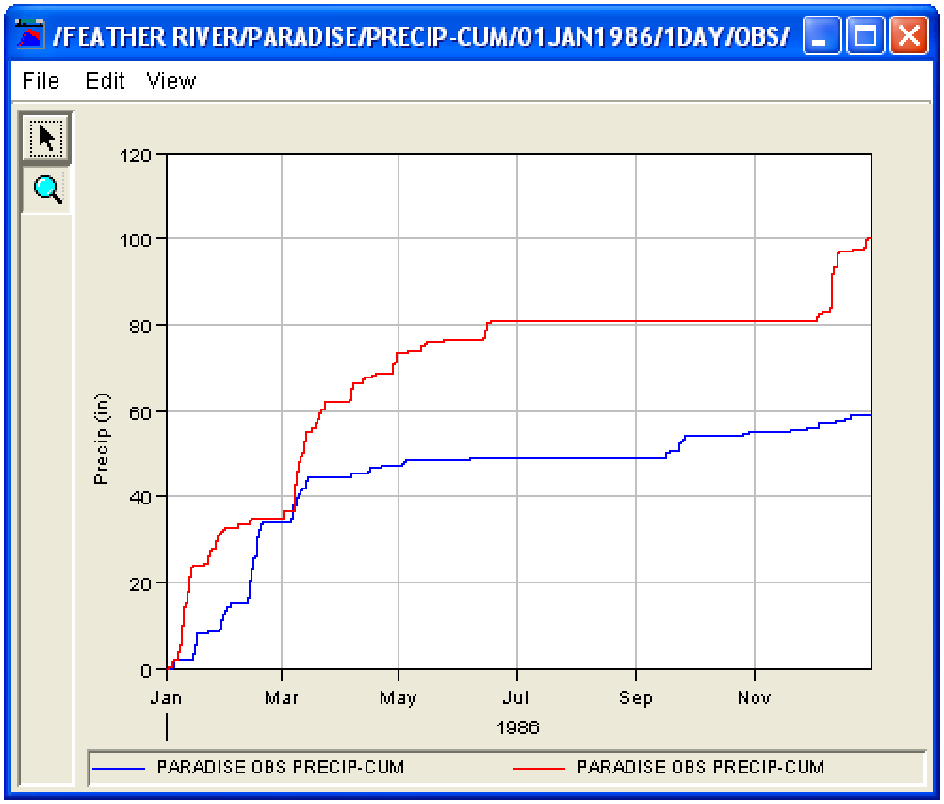
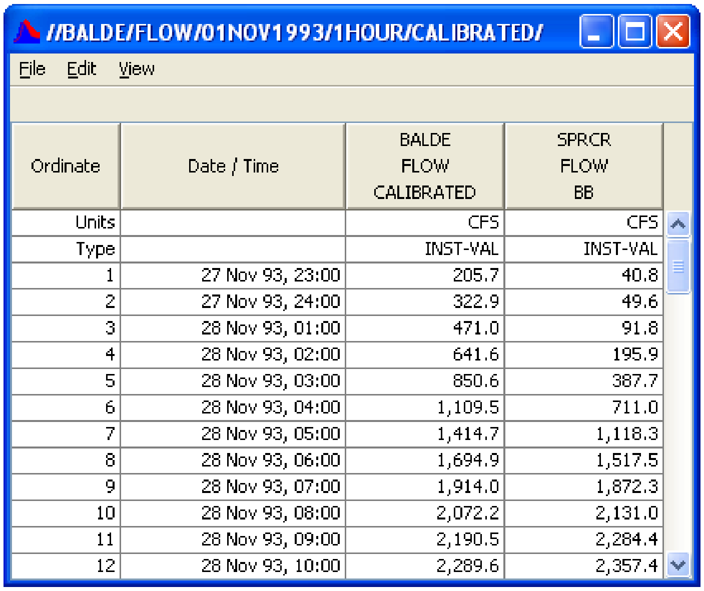


Figure 3.3 Normalized and Time Synched Precipitation Data for Years 1986 and 1995

3.3 Viewing Tabular Data

An example table produced using HEC-DSSVue is shown in Figure 3.4. Tables allow you to view and edit HEC-DSS data in a vertical scrolling



Ordinate	Date / Time	BALDE FLOW CALIBRATED	SPRCR FLOW BB
Units		CFS	CFS
Type		INST-VAL	INST-VAL
1	27 Nov 93, 23:00	205.7	40.8
2	27 Nov 93, 24:00	322.9	49.6
3	28 Nov 93, 01:00	471.0	91.8
4	28 Nov 93, 02:00	641.6	195.9
5	28 Nov 93, 03:00	850.6	387.7
6	28 Nov 93, 04:00	1,109.5	711.0
7	28 Nov 93, 05:00	1,414.7	1,118.3
8	28 Nov 93, 06:00	1,694.9	1,517.5
9	28 Nov 93, 07:00	1,914.0	1,872.3
10	28 Nov 93, 08:00	2,072.2	2,131.0
11	28 Nov 93, 09:00	2,190.5	2,284.4
12	28 Nov 93, 10:00	2,289.6	2,357.4

Figure 3.4 Example Tabulation from HEC-DSSVue

window that shows the ordinate (starting from the start date/time), the date and time stamp, and the values for the selected data sets. From the **File** menu of a table dialog box (see Figure 3.4, page 3-4), you can view the tabular data in plot format by clicking **Plot**. Respectively, you can tabulate data displayed in a plot dialog box, from the **File** menu, click **Tabulate**.

3.3.1 Accessing Tables

To access tables, first select the pathnames of the records you wish to view. There are several ways to select pathnames:

- Double-click on an individual pathname in the **HEC-DSS Pathname** list.
- Highlight a pathname in the **HEC-DSS Pathname** list then click **Select**. Until you select a pathname, the **Select** button remains inactive.
- Click and drag your mouse to select a series of pathnames, and then click **Select**. You can also use **Ctrl+click** to select multiple, non-consecutive pathnames.
- If you wish to select all of the pathnames, click **Select All** from the **Edit** menu.
- If no pathnames are in the selection list, individual pathnames can just be highlighted for quick selection.
- Drag additional datasets from the main screen onto the table to add them to the table.

Once you have selected the pathnames you want to visualize, you can open a table by clicking the **Tabulate** ▤ button from the **Tool Bar**, or from the **Display** menu click **Tabulate**.

3.3.2 Customizing the Display of Tabular Data

From a tabular data dialog box, from the **View** menu, you can choose to display commas in numbers by clicking **Commas**. You can reverse the order of the table by clicking **Reverse Order**. For time series data, this option will show the latest values first. The date and time of the data can be tabulated in separate columns by clicking **Date and Time Separately**. To have the dates display the years with four digits, click **Date With 4 Digit Years**. You can set the precision of decimal places for your data by pointing to **Show Decimal Places** and then clicking the number of decimal places you wish to display (see Figure 3.5, page 3-6). You can also change how missing values are displayed by pointing to **Show Missing As** and then by clicking **blanks** (the default), **-901.0, M**, or **-M-**.

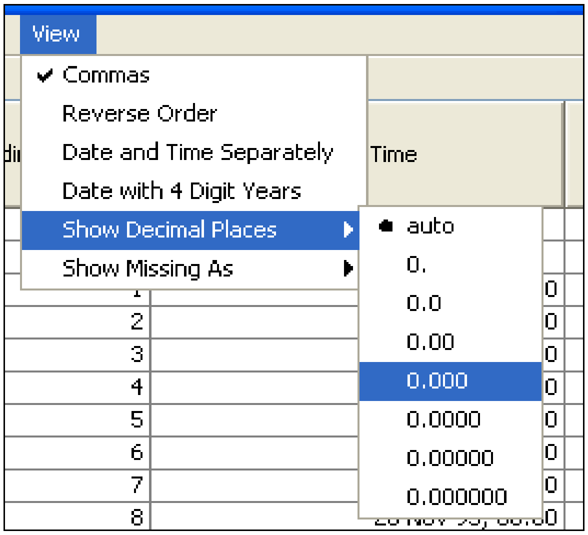


Figure 3.5 HEC-DSSVue Tables have Several Options for Displaying Data

3.3.3 Searching for Values

You can search for specific values in your dataset, from a tabulate dialog box (see Figure 3.4, page 3-4), from the **Edit** menu click **Find** (Figure 3.6), or press the **F3** key. Using either option will open the **Find** dialog box (Figure 3.7).

In the **Find what** box (Figure 3.7), enter the item that you are searching for in your dataset. When you click **Find Next** (Figure 3.7), the next value that matches will be highlighted. You can search up from where your cursor is, or search down. You have the option of searching for the exact number or part of that number. You can also search for a date or time.

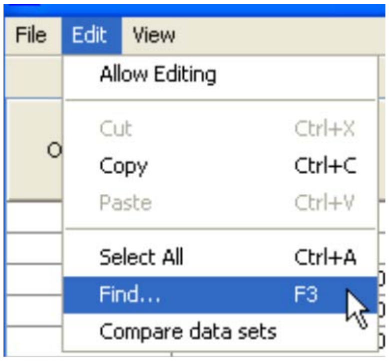
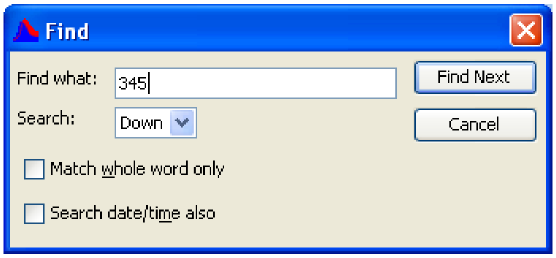


Figure 3.6 Tabulate Edit Menu



Find what: 345

Search: Down

☐ Match whole word only

☐ Search date/time also

Figure 3.7 Find Dialog Box

3.3.4 Comparing Datasets

If you tabulate two or more similar time series data sets, you can compare values from both sets from the table. From a tabulate dialog box (see Figure 3.4, page 3-4), from the **Edit** menu, click **Compare data sets** (Figure 3.8)

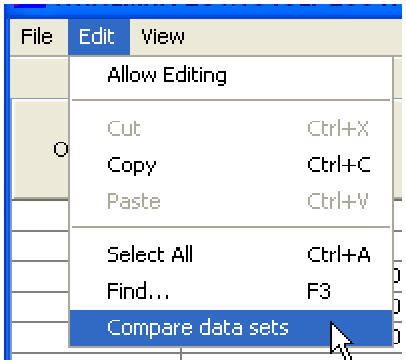
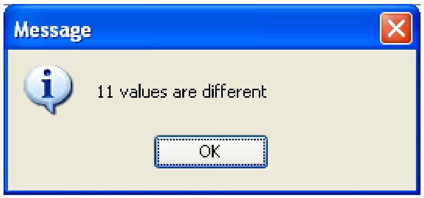


Figure 3.8 Compare Data Sets

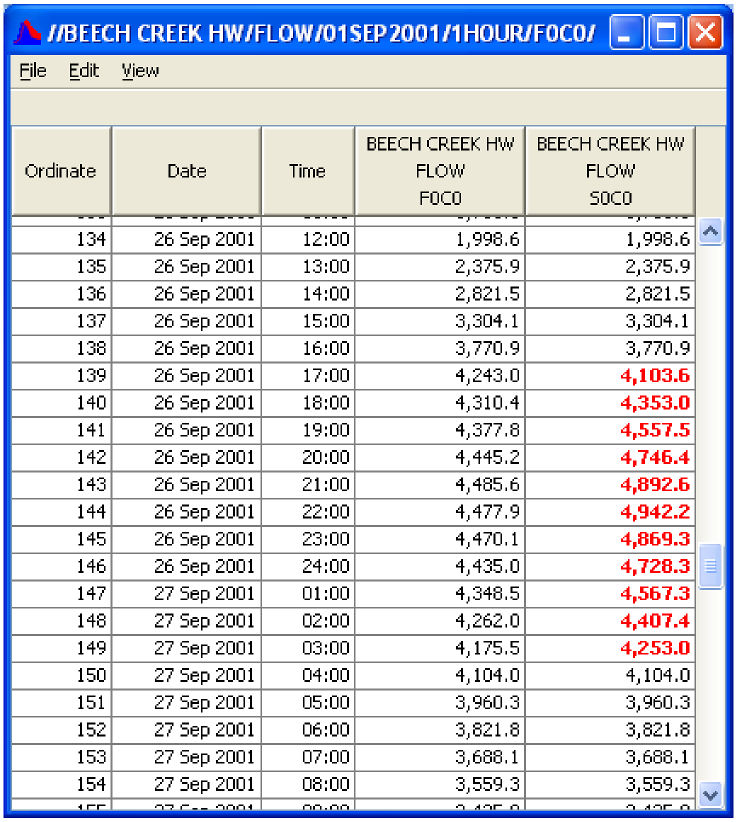
A message will appear displaying the number of values that differ in the selected data sets (Figure 3.9) and the differences between the two data sets will be highlighted in red, as show in Figure 3.10.


Message

11 values are different

Figure 3.9 Display of the number of values that vary

For more information and options to compare datasets, see Chapter 4, Section 4.7.


//BEECH CREEK HW/FLOW/01SEP2001/1HOUR/FOCO/

Ordinate	Date	Time	BEECH CREEK HW FLOW F0C0	BEECH CREEK HW FLOW S0C0
134	26 Sep 2001	12:00	1,998.6	1,998.6
135	26 Sep 2001	13:00	2,375.9	2,375.9
136	26 Sep 2001	14:00	2,821.5	2,821.5
137	26 Sep 2001	15:00	3,304.1	3,304.1
138	26 Sep 2001	16:00	3,770.9	3,770.9
139	26 Sep 2001	17:00	4,243.0	**4,103.6**
140	26 Sep 2001	18:00	4,310.4	**4,353.0**
141	26 Sep 2001	19:00	4,377.8	**4,557.5**
142	26 Sep 2001	20:00	4,445.2	**4,746.4**
143	26 Sep 2001	21:00	4,485.6	**4,892.6**
144	26 Sep 2001	22:00	4,477.9	**4,942.2**
145	26 Sep 2001	23:00	4,470.1	**4,869.3**
146	26 Sep 2001	24:00	4,435.0	**4,728.3**
147	27 Sep 2001	01:00	4,348.5	**4,567.3**
148	27 Sep 2001	02:00	4,262.0	**4,407.4**
149	27 Sep 2001	03:00	4,175.5	**4,253.0**
150	27 Sep 2001	04:00	4,104.0	4,104.0
151	27 Sep 2001	05:00	3,960.3	3,960.3
152	27 Sep 2001	06:00	3,821.8	3,821.8
153	27 Sep 2001	07:00	3,688.1	3,688.1
154	27 Sep 2001	08:00	3,559.3	3,559.3

Figure 3.10 Comparison of Two Data Sets

3.4 Editing Tabular Data

In HEC-DSSVue, you can edit data directly in tables, but first you must set the table to allow editing.

From a tabulate dialog box (see Figure 3.4, page 3-4), from the **Edit** menu, click **Allow Editing** (Figure 3.11). This will set the table editable and allow you to manually edit the listed data. When this option is checked, the background of non-editable table cells display grey and the **Cut, Paste, Insert Rows**, and **Delete Rows** commands become available from the **Edit** menu.

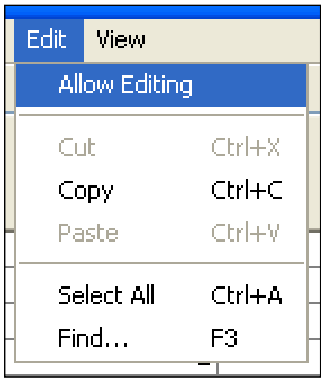


Figure 3.11 Edit Menu - Allow Editing

If the **Allow Editing** option is disabled (grayed-out), you may not have permission to write to the file or the table was started from a dialog that does not have access to write to the file (such as a plot window).

Changes made to a table can be saved, from the **File** menu, click **Save** or click **Save As**. If you do not save your data before you close the table window, HEC-DSSVue will prompt you to save your changes before it closes the window.

3.4.1 Selecting Table Cells

To select an individual table cell, simply click on it.

To select several consecutive cells, click, hold down and drag your mouse over the cells. You can also hold down the **Ctrl** key and click to select multiple, non-consecutive records.

If you wish to select all rows in a table, from the **Edit** menu, click **Select All**.

You can also use shortcut menu items to edit multiple selected cells in tables (Figure 3.12).

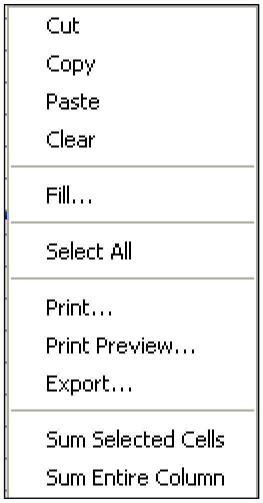


Figure 3.12 Shortcut Menu

3.4.2 Cutting and Pasting Data

The **Cut** command removes data from the cell and places it on the computers clipboard in ASCII format. For regular-interval time series data, the **Cut** command will set the emptied cell to the missing data flag.

You can cut data from one set of cells and paste it into another set of cells (in the same table or another table). To do a cut and paste:

1. From the **View** menu, click **Allow Editing**.
2. Select the cells you want to **Cut** the data from.
3. From either the **View** menu or the shortcut menu (see Figure 3.12, page 3-8), click **Cut**.
4. Select the cells where you want to **Paste** the data.
5. From either the **View** menu or the shortcut menu (see Figure 3.12, page 3-8), click **Paste**.

Tip: *The keyboard shortcut keys,* ***Ctrl + x*** *and* ***Ctrl + v*** *also work for cut and paste.*

3.4.3 Copying and Pasting Data

The **Copy** command copies selected cell data and places it on the computers clipboard in ASCII format.

You can copy data from one set of cells and paste it into another set of cells (in the same table or another table). To do a copy and paste:

1. From the **View** menu, click **Allow Editing**.
2. Select the cells you want to **Copy** the data from.
3. From either the **View** menu or the shortcut menu (see Figure 3.12, page 3-8), click **Copy**.
4. Select the cells where you want to **Paste** the data.
5. From either the **View** menu or the shortcut menu (see Figure 3.12, page 3-8), click **Paste**.

Tip*:* *The keyboard shortcut keys,* ***Ctrl + x*** *and* ***Ctrl + v*** *also work for cut and paste.*

3.4.4 Clearing Table Cells

To delete and remove data from table cells use the **Clear** command. Cleared data is not saved to the computer's clipboard, so you will not be able to paste the data back into the table. However, for time series data, the **Clear** command will remove the data from the cell and add a missing flag to the emptied cell. To clear table cells:

1. From the **View** menu, click **Allow Editing**.
2. Select the cells you want to clear.
3. From the shortcut menu (see Figure 3.12, page 3-8), click **Clear**.

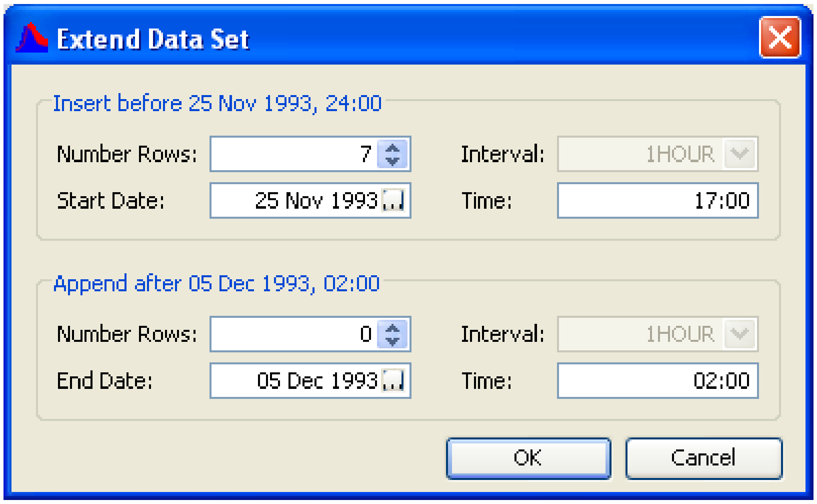
3.4.5 Adding and Inserting Rows

Depending on what type of data is being edited in a table, rows can be inserted, appended, or added to the beginning or end of the table.

Appending rows is automatic when **Allow Editing** is set. When **Allow Editing** is selected, two blank rows automatically appear at the end of the table. When ever you enter data in a row at the bottom of the table, an additional blank row will be appended. When you save the data, any blank rows at the end of the table are removed automatically.

For regular interval time series data, rows can only be inserted at the beginning or end of the data set. For irregular interval time series data and paired data, rows can be inserted between any existing table rows. To insert a row into the table:

1. Using your mouse pointer, click on the row you would like to insert rows next to and then from the **Edit** menu, click **Insert Rows**.
2. The **Extend Data Set** dialog box will open (Figure 3.13). This dialog may be different from what's pictured in Figure 3.13 depending on what type of data is displayed in the table.



Extend Data Set

Insert before 25 Nov 1993, 24:00

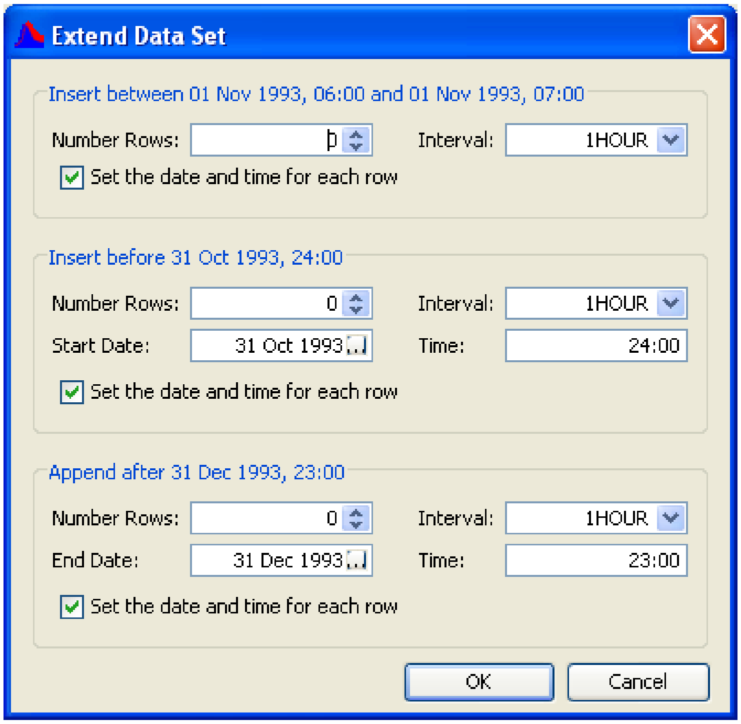
Number Rows: 7 Interval: 1HOUR
Start Date: 25 Nov 1993 Time: 17:00

Append after 05 Dec 1993, 02:00

Number Rows: 0 Interval: 1HOUR
End Date: 05 Dec 1993 Time: 02:00

Figure 3.13 Extend Data Set Dialog Box for Regular Interval Data

3. For irregular interval time series data (see Figure 3.14, page 3-11), you have the option to **Insert between the selected dates**, **Insert before the first date**, and **Append after the last date**. Regular interval data can only have data inserted before and after the existing time steps. Each choice requires you to enter the number of rows that will be added. From the **Number Rows** list, enter or select the number of rows that will be added to the table at the given area.



Extend Data Set

Insert between 01 Nov 1993, 06:00 and 01 Nov 1993, 07:00

Number Rows: 0 Interval: 1HOUR

☑ Set the date and time for each row

Insert before 31 Oct 1993, 24:00

Number Rows: 0 Interval: 1HOUR

Start Date: 31 Oct 1993 Time: 24:00

☑ Set the date and time for each row

Append after 31 Dec 1993, 23:00

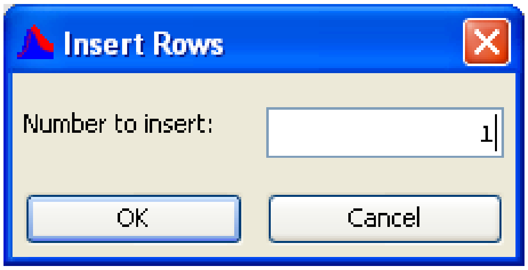
Number Rows: 0 Interval: 1HOUR

End Date: 31 Dec 1993 Time: 23:00

☑ Set the date and time for each row

Figure 3.14 Extend Data Set Dialog Box for Irregular Interval Data

4. Irregular interval time series data requires an interval to be selected. From the **Interva**l list, select an interval, default is *1HOUR*. If you would like the data and time filled for each inserted row, be sure **Set the data and time for each row** is selected.
5. For a paired data table, all that is required is entering the number of rows you would like to insert. From the **Edit** menu, click **Insert Rows,** the **Insert Rows** dialog box will open (Figure 3.15). In the **Number to insert** box, enter the number of rows you would like to add. Paired data tables also allow you to add additional columns, from.



Insert Rows

Number to insert: 1

Figure 3.15 Insert Rows Dialog Box

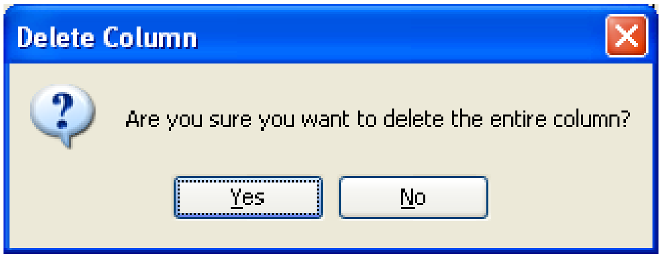
6. Once you are done entering information for the insert, click **OK**.
7. The entered number of blank rows will be inserted at that row, moving the original row that you selected down.

3.4.6 Deleting Rows and Columns

You may delete rows from a table when working with irregular-interval time series data and paired data. To delete a row from a table, select the row and then from the **Edit** menu, click **Delete Rows**.

If you are working with regular interval time series data, you will not be able to delete rows; instead, the **Delete Rows** function will change the data values to missing. This is the same as clicking **Clear** (see Figure 3.12, page 3-8) from the shortcut menu.

In a paired data table you can delete a column. To delete a paired data column, select the column and then from the **Edit** menu, click **Delete Column**. The **Delete Column** message box will open (Figure 3.16), requiring you to verify you would like to delete the entire column. Click **Yes** to finish deleting the column.


Delete Column

Are you sure you want to delete the entire column?

Figure 3.16 Delete Column Message Box

3.5 Printing, Copying, and Exporting Tables

HEC-DSSVue tables offer several commands that allow you to print, copy or export data and then paste tables into other applications such as Microsoft Excel and Word. The following sections describe each in detail.

3.5.1 Printing Tables

You can access the **Print** and **Print Preview** commands from either the **File** menu (Figure 3.17) or from the shortcut menu (see Figure 3.18, page 3-13) of the table window.

In a table, the **Print** and **Print Preview** commands open the **Properties** dialog box (see Figure 3.19, page 3-13), which offers options on three tabs.

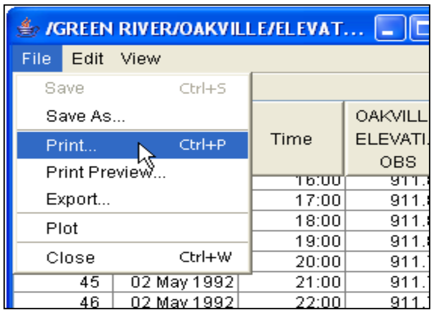


Figure 3.17 File Menu - Tabulate Dialog Box

The **Page** tab allows you to specify the page **Orientation, Scaling**, and **Selection**. You can also choose to print the table as **ASCII, Repeat Headers** on every page, and print the **Gridlines**.

On the **Header/Footer** tab you can type in the header and footer you want to appear on your printed pages.

The **Table Title** tab offers a default title for the table based on the data source, which you can change.

On the **Properties** dialog box, the **Print** command performs two functions, depending on whether you arrived at the dialog box via the **Print** command or the **Print Preview** command.

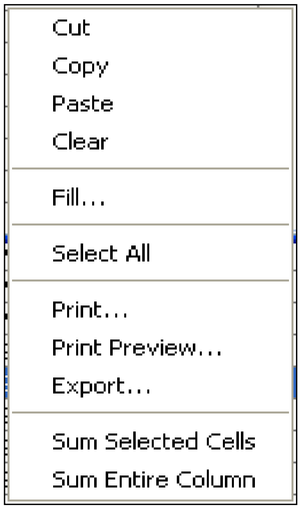
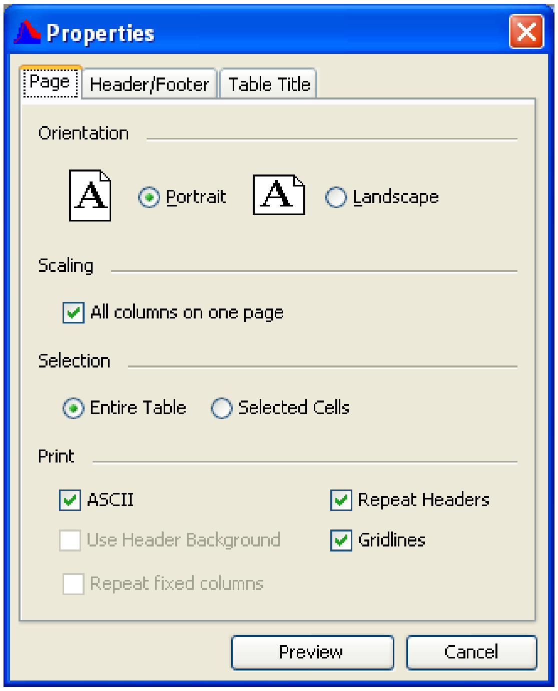


Figure 3.18 Print Shortcut Menu



Orientation

● Portrait ○ Landscape

Scaling

☑ All columns on one page

Selection

● Entire Table ○ Selected Cells

Print

☑ ASCII ☑ Repeat Headers

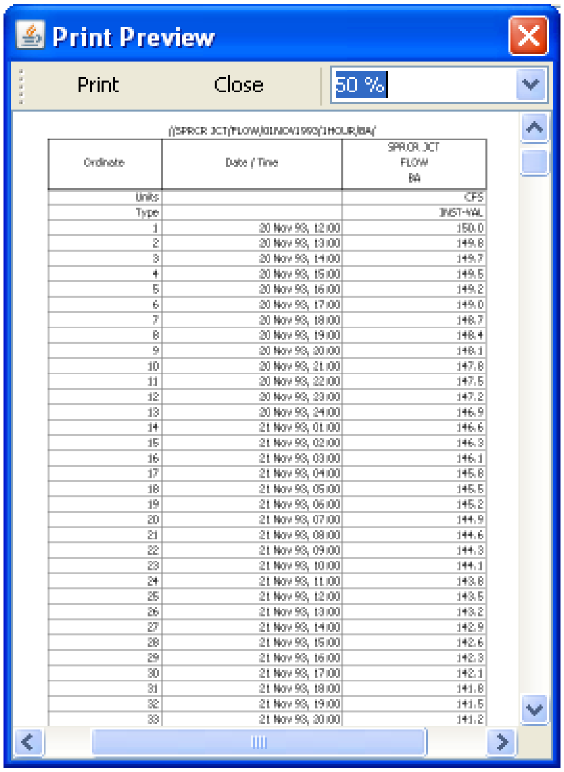
☐ Use Header Background ☑ Gridlines

☐ Repeat fixed columns

Figure 3.19 Properties Dialog Box - Printing

If the **Properties** dialog box was opened using the **Print Preview** menu, the **Preview** button on the **Properties** dialog box opens a **Print Preview** dialog box that allows you to view the table as it will look when it is printed, see Figure 3.20 (see page 3-14). You can click **Print** in the **Print Preview** dialog box to print your table.

From the **Print** command, the **Print** button on the **Properties** dialog box opens a Windows-style print dialog box where you can choose your



//SPRCR JCT/FLOW/01NOV1993/1HOUR/BA/

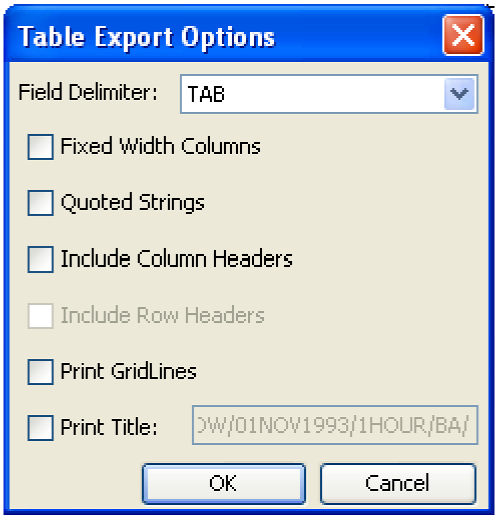
Ordinate	Date / Time	SPRCR JCT FLOW BA
Units		CFS
Type		INST-VAL
1	20 Nov 93, 12:00	150.0
2	20 Nov 93, 13:00	149.8
3	20 Nov 93, 14:00	149.7
4	20 Nov 93, 15:00	149.5
5	20 Nov 93, 16:00	149.2
6	20 Nov 93, 17:00	149.0
7	20 Nov 93, 18:00	148.7
8	20 Nov 93, 19:00	148.4
9	20 Nov 93, 20:00	148.1
10	20 Nov 93, 21:00	147.8
11	20 Nov 93, 22:00	147.5
12	20 Nov 93, 23:00	147.2
13	20 Nov 93, 24:00	146.9
14	21 Nov 93, 01:00	146.6
15	21 Nov 93, 02:00	146.3
16	21 Nov 93, 03:00	146.1
17	21 Nov 93, 04:00	145.8
18	21 Nov 93, 05:00	145.5
19	21 Nov 93, 06:00	145.2
20	21 Nov 93, 07:00	144.9
21	21 Nov 93, 08:00	144.6
22	21 Nov 93, 09:00	144.3
23	21 Nov 93, 10:00	144.1
24	21 Nov 93, 11:00	143.8
25	21 Nov 93, 12:00	143.5
26	21 Nov 93, 13:00	143.2
27	21 Nov 93, 14:00	142.9
28	21 Nov 93, 15:00	142.6
29	21 Nov 93, 16:00	142.3
30	21 Nov 93, 17:00	142.1
31	21 Nov 93, 18:00	141.8
32	21 Nov 93, 19:00	141.5
33	21 Nov 93, 20:00	141.2

Figure 3.20 Print Preview Dialog Box - Table

printer, set printer properties, and specify the number of copies to print. You can also print your table to a file instead of to a printer.

3.5.2 Exporting Tables

Tables can be exported and converted into a *.txt file. Once the table is in the form of a *.txt file, it can be opened in a different application such as Microsoft Excel or Word.



Field Delimiter: TAB

□ Fixed Width Columns

□ Quoted Strings

□ Include Column Headers

□ Include Row Headers

□ Print GridLines

□ Print Title: DW/01NOV1993/1HOUR/BA/

Figure 3.21 Table Export Options Dialog Box

From the shortcut menu or the **File** menu, click **Export**, the **Table Export Options** dialog box will open (Figure 3.21).

In the **Table Export Options** dialog box, you can choose a **Field Delimiter** (**TAB**, **SPACE**, **COMMA**, **COLON**, or **SEMI-COLON**), specify **Fixed-Width Columns**, choose to display **Quoted Strings**, **Include Column Headers**, and opt to **Print Gridlines** and a **Title**.

3.5.3 Copying Tables to the Clipboard for Use in Other Applications

To copy a table to the clipboard, from a table dialog box, from the **Edit** menu, click **Copy** (Figure 3.22). You can also right-click inside the table and click **Copy** from the shortcut menu (Figure 3.23). To copy the entire table from the shortcut menu, click **Select All** and then click **Copy**. You can then paste the table as tab-separated values into another application such as Microsoft Excel or Word. Only the data is copied to the clipboard; the table gridlines are not copied. (Use the export options when you want to copy table gridlines.)

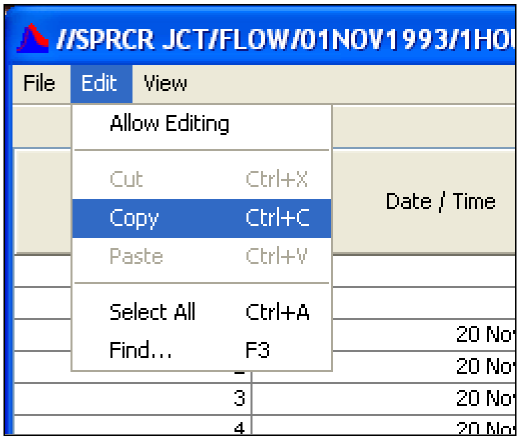


Figure 3.22 Edit Menu - Table Dialog Box

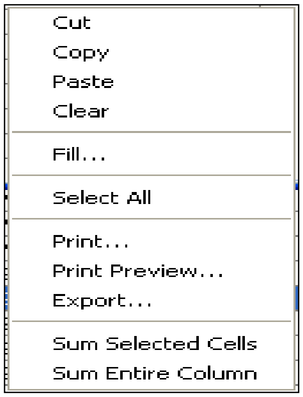


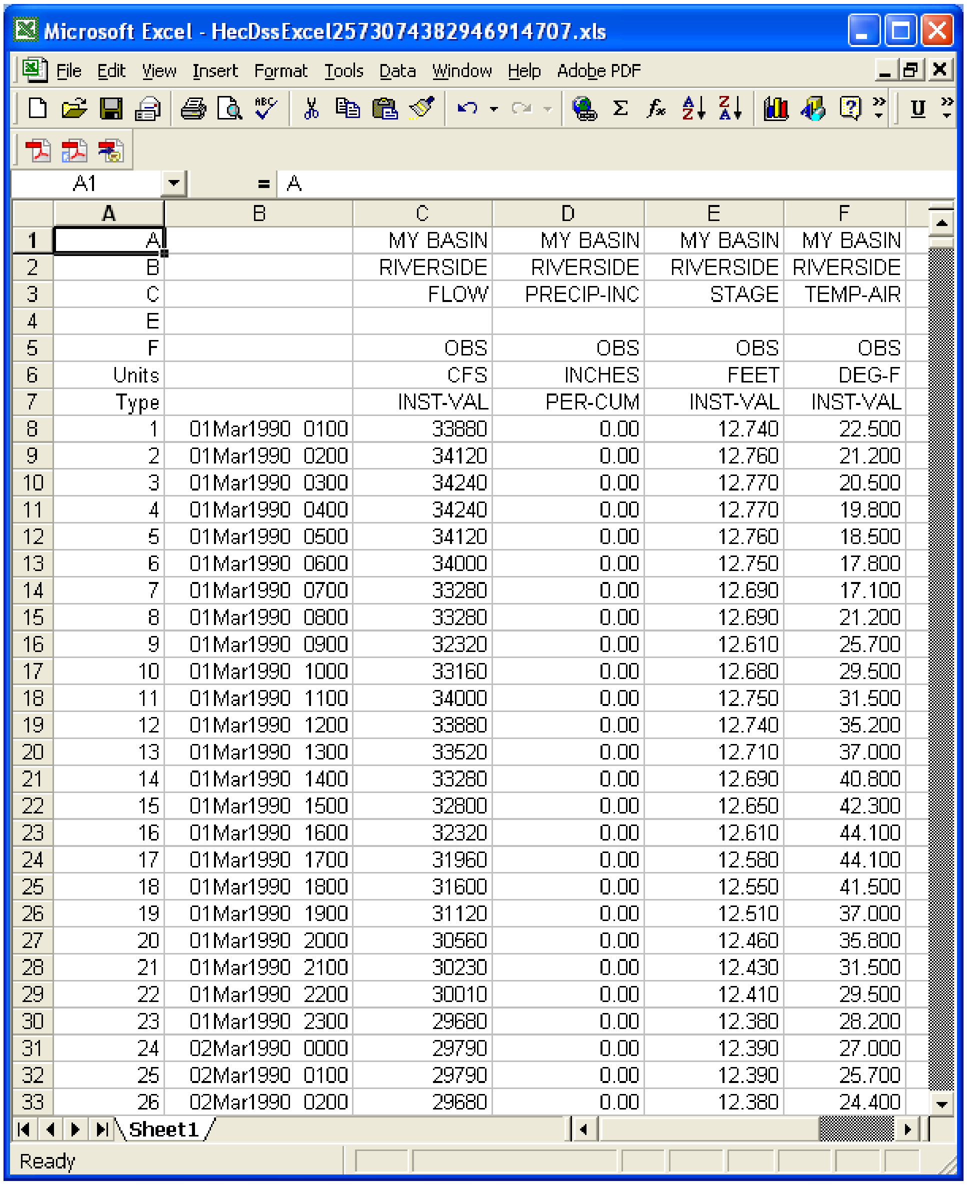
Figure 3.23 Shortcut Menu - Copy

You can also highlight a portion of the table and then click **Copy**. This will copy the portion of the table that you highlighted to the clipboard.

3.5.4 Tabulate and Editing using MS Excel (Optional)

If the Excel plug-in is loaded, you can tabulate (or export) data in Microsoft Excel and edit time series data in Excel. Simply select the data sets you wish to tabulate as you normally would and then either press the **Excel** tool in the toolbar, or from the **Display** menu, click **Tabulate in Exce**l (assuming you have Microsoft Excel installed). Figure 3.24 (see page 3-16) shows how the data is displayed in Excel.

To edit data in Excel, select your data sets and from the **Edi**t menu, click **Edit in Excel**. This will load the selected data into an Excel spreadsheet


Microsoft Excel - HecDssExcel25730743829469147O7.xls
File Edit View Insert Format Tools Data Window Help Adobe PDF
A1
= A
Sheet1
Ready

	A	B	C	D	E	F
1	A		MY BASIN	MY BASIN	MY BASIN	MY BASIN
2	B		RIVERSIDE	RIVERSIDE	RIVERSIDE	RIVERSIDE
3	C		FLOW	PRECIP-INC	STAGE	TEMP-AIR
4	E					
5	F		OBS	OBS	OBS	OBS
6	Units		CFS	INCHES	FEET	DEG-F
7	Type		INST-VAL	PER-CUM	INST-VAL	INST-VAL
8	1	01Mar1990 0100	33880	0.00	12.740	22.500
9	2	01Mar1990 0200	34120	0.00	12.760	21.200
10	3	01Mar1990 0300	34240	0.00	12.770	20.500
11	4	01Mar1990 0400	34240	0.00	12.770	19.800
12	5	01Mar1990 0500	34120	0.00	12.760	18.500
13	6	01Mar1990 0600	34000	0.00	12.750	17.800
14	7	01Mar1990 0700	33280	0.00	12.690	17.100
15	8	01Mar1990 0800	33280	0.00	12.690	21.200
16	9	01Mar1990 0900	32320	0.00	12.610	25.700
17	10	01Mar1990 1000	33160	0.00	12.680	29.500
18	11	01Mar1990 1100	34000	0.00	12.750	31.500
19	12	01Mar1990 1200	33880	0.00	12.740	35.200
20	13	01Mar1990 1300	33520	0.00	12.710	37.000
21	14	01Mar1990 1400	33280	0.00	12.690	40.800
22	15	01Mar1990 1500	32800	0.00	12.650	42.300
23	16	01Mar1990 1600	32320	0.00	12.610	44.100
24	17	01Mar1990 1700	31960	0.00	12.580	44.100
25	18	01Mar1990 1800	31600	0.00	12.550	41.500
26	19	01Mar1990 1900	31120	0.00	12.510	37.000
27	20	01Mar1990 2000	30560	0.00	12.460	35.800
28	21	01Mar1990 2100	30230	0.00	12.430	31.500
29	22	01Mar1990 2200	30010	0.00	12.410	29.500
30	23	01Mar1990 2300	29680	0.00	12.380	28.200
31	24	02Mar1990 0000	29790	0.00	12.390	27.000
32	25	02Mar1990 0100	29790	0.00	12.390	25.700
33	26	02Mar1990 0200	29680	0.00	12.380	24.400

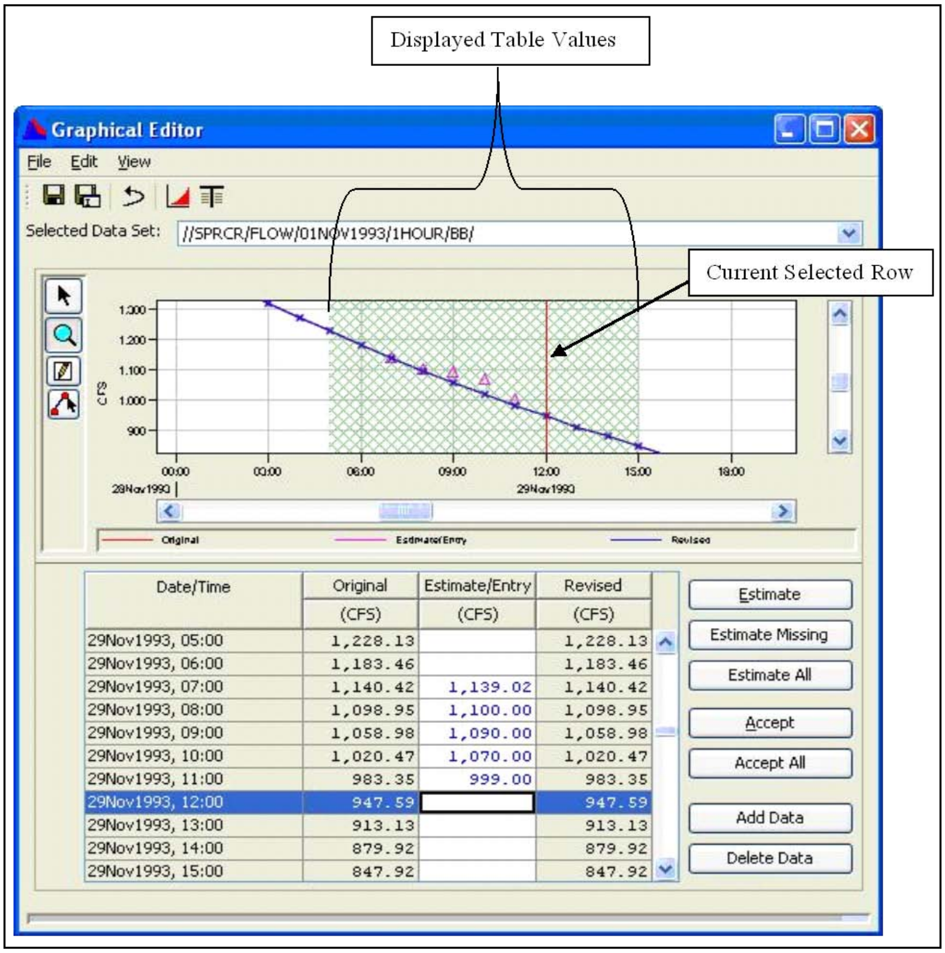
Figure 3.24 Tabular Data Exported to Excel

and then read it back to DSS once you have saved it. In this mode, you may not add or delete rows or columns; you can only change the data. After you are finished, save the Excel spreadsheet and then click **OK** or **Cancel** from the dialog from HEC-DSSVue.

> ***Note*** *- Take caution with dates and time in Excel. Excel shows times as "Beginning of Period", while the HEC-DSS convention shows times as "End of Period". A data value for midnight will show up as 2400 hours at the end of the day in HEC-DSS, while Excel will display that as 0000 hours for the next day. For daily data, the hours might not be shown in Excel and the data can be misinterpreted to be for the following day.*

3.6 Graphical Editing

You can view and edit time series data in a combined plot/table using the **Graphical Editor** (Figure 3.25) for HEC-DSSVue. To access the HEC-DSSVue **Graphical Editor**, select one or more pathnames from the main HEC-DSSVue screen, from the **Edit** menu, click **Graphical Edit** or from the toolbar click the **Graphical Edit** button. Only time series data can be edited through the graphical editor.



Date/Time	Original	Estimate/Entry	Revised
	(CFS)	(CFS)	(CFS)
29Nov1993, 05:00	1,228.13		1,228.13
29Nov1993, 06:00	1,183.46		1,183.46
29Nov1993, 07:00	1,140.42	1,139.02	1,140.42
29Nov1993, 08:00	1,098.95	1,100.00	1,098.95
29Nov1993, 09:00	1,058.98	1,090.00	1,058.98
29Nov1993, 10:00	1,020.47	1,070.00	1,020.47
29Nov1993, 11:00	983.35	999.00	983.35
29Nov1993, 12:00	947.59		947.59
29Nov1993, 13:00	913.13		913.13
29Nov1993, 14:00	879.92		879.92
29Nov1993, 15:00	847.92		847.92

Figure 3.25 Graphical Editor

The **Graphical Editor** displays an editable plot of the data on the top of the editor and a scrollable table of the data on the bottom of the editor. The area of the data that is visible in the table is marked with a green hash region in the plot. As you scroll through the table, the green hash area moves along with the data that is displayed. When you select a point on the curve or in the table, a red vertical line is drawn at that point to show which value has been selected.

You can edit data in the **Graphical Editor** by drawing a curve with your mouse, or by selecting a point on the curve and moving that point up or down, or by changing a value in the associated table. You can also have the editor linear interpolate missing points, or use the other fill options associated with tables.

You can only edit one curve at a time, but you can plot other data sets on the same graph for comparison purposes. You can also select multiple data sets and scroll through them as you edit. The major components of the **Graphical Editor** are shown in Figure 3.25 (page 3-17).

The components of the graphical editor are:

- **Current Selected Row** shows what data value is selected on the plot and its corresponding row in the table.

- **Displayed Table Values** indicates the area in the plot that corresponds to the data showing in the table.

- **Selected Data Set** field displays the pathname for the data set being edited.

- **Estimate** button generates a linear interpolation for selected rows for missing data or copies the original data into the editable field. The **Estimate All** button performs this for all rows in the table. Data must be in the editable field before the point editor can be used.

- The **Accept** button copies data from the **Estimate/Entry** column into the **Revised** column for the selected rows. This is for intermediate saving of the last data edited. If you want to retain all changed values, the **Accept All** button copies all data from the **Estimate/Entry** column into the **Revised** column.

- The **Add Data** button adds rows for irregular interval time series data. A dialog will be displayed to set the date and time for the row added.

- The **Delete Data** button removes rows for irregular-interval time series data. You cannot remove regular-interval data; you can only change it to missing.

- **Original** column contains the values of the original data set. The values are generally depicted with a red curve in the plot with each point marked by an 'x'. If the value is the same as the one in the **Revised** column, that point will be over drawn with the blue 'x' as well.

- **Estimate/Entry** (edit) column is an empty editable column where you enter data manually or edit with the point tool. These values show-up in the plot as magenta triangles. When you are satisfied with the changes you made in this column, press the **Accept** or **Accept All** button.

- **Revised** (edited) column contains the final values that will be saved when **Save** or **Save As** is selected. These values display as a blue 'x' in the plot.

3.6.1 Graphical Editor Tools and Buttons

The **Graphical Editor** contains mouse tools and shortcut icon buttons to help aid with the editing process. Available mouse tools:

Select Point tool selects a point on the curve to edit by moving the mouse to that point and clicking the left mouse button.

Zoom tool put the mouse in a zoom mode. To zoom-in, move the mouse to one corner of the area you want to zoom-in to, click and hold down the left mouse button while dragging the mouse to the opposite corner and the release the button. To zoom-out, click the right mouse button. The graph is zoomed-out in increments.

Single Point edit tool puts the mouse in a point edit mode, as described in Section 3.6.3.

Line Draw is a multi-point edit tool that puts the mouse in a line draw mode, as described in the next section.

Available tool bar buttons:

Save button saves the data in the **Revised** column to the DSS file without changing the pathname.

Save As button saves the data in the **Revised** column to the DSS file after allowing you to change one or more of the pathname parts.

Undo button restores the values to the original data set, undoing all changes.

Plot button plots the original and revised data in a standard plot window.

Tabulate button tabulates the original and revised data in a standard table.

3.6.2 Editing a Curve with the Line Draw Tool

You can edit data by drawing a curve freehand by moving the mouse from left to right while clicking on the left mouse button at selected inflection points. To do this:

1. Select the data set from the drop down box at the top of the screen if it is not selected already.
2. Select the **Line Draw** (multi-point edit) tool.
3. Move the mouse to the left-most position of the curve where you want to start editing.
4. Click on the left mouse button to mark the first data point.
5. Move the mouse right along your desired curve and press the left mouse button at inflection points. The **Graphical Editor** will draw a straight line between points.
6. If you make a mistake, you can move the mouse left past that point and insert a new inflection point. You can backup as many points at one time as you want.
7. When you are finished with a curve segment, press the right mouse button. (That location will not be used as part of the curve.) The line segments just given will be drawn with magenta triangles, and those data points are entered in the **Estimate/Entry** column.
8. You can draw other curve segments elsewhere on the plot using the same procedure.
9. If you need to remove a curve segment that you have drawn, select the values for it in the table and press the **Estimate** button. The original values will be copied into the **Estimate/Entry** column.
10. After completing your curve, press the **Accept** button or the **Accept All** button to accept all edits. This will copy the data from the **Estimate/Entry** column into the **Revised** column and will be ready to save to the HEC-DSS file.
11. At the end of all of your edits, press the **Save** or **Save As** button or select that item from the **File** menu to store your changes in the HEC-DSS file.

3.6.3 Editing a Point with the Single-Point Edit Tool

You can edit single data values by double clicking on the new point that you want or clicking on the existing point and moving it vertically. To do this:

1. Select the data set from the drop down box at the top of the screen if it is not selected already.
2. Click the **Single-Point** edit tool.

3. Find the new value of the point on the graph that you want to edit and double click the mouse at that location. This will enter the value that you selected into the **Estimate/Entry** column.
4. Edit the remaining points that you desire to have changed and when you are complete, press the **Accept** button or the **Accept All** button to accept all edits. This will copy the data from the **Estimate/Entry** column into the **Revised** column and will be ready to save to the HEC-DSS file.
5. At the end of all of your edits, press the **Save** or **Save As** button or select that item from the **File** menu to store your changes in the HEC-DSS file.

You can also grab the point that you want to modify and move it vertically. To do this:

1. Click the left mouse button on the data point you want to edit. The table row containing that point will be highlighted.
2. Press the **Estimate Missing** button to provide a value to start from, unless there is already a value in the Estimate column. A magenta triangle will be drawn at that point on the curve. To edit existing points, double click the point and a magenta triangle will appear.
3. Move the mouse over the triangle and press and hold down the left mouse button. Move the mouse up or down to the desired value.
4. Edit the remaining points that you desire to have changed and when you are complete, press the **Accept** button or the **Accept All** button to accept all edits. This will copy the data from the **Estimate/Entry** column into the **Revised** column and will be ready to save to the HEC-DSS file.
5. At the end of all of your edits, press the **Save** or **Save As** button or select that item from the **File** menu to store your changes in the HEC-DSS file.

3.6.4 Editing Data in the Table

You can edit, add, and delete data in an irregular-interval time series table but you cannot extend the table. Regular-interval values cannot be deleted or added but you can change values to missing. To edit data within the table:

1. Select the data set from the drop down box at the top of the screen if it is not selected already.
2. Select the row in the table that you want to edit at, or select the location on the graph with the mouse after pressing the **Select** tool.
3. Type in the new values in the **Estimate/Entry** column. To move to the next row in the column, press the **Tab** key or use the up and down arrows.

4. You can copy the original value to the **Estimate/Entry** column by selecting all data sets you want to change, right-clicking and selecting **Fill - Linear**, as shown in Figure 3.26. You can then change it by pressing the **Estimate** or **Estimate All** button.
5. When complete, press the **Accept** button or the **Accept All** button to accept all edits. This will copy the data from the **Estimate/Entry** column into the **Revised** column and will be ready to save to the HEC-DSS file.
6. At the end of all of your edits, press the **Save** or **Save As** button or select that item from the **File** menu to store your changes in the HEC-DSS file.

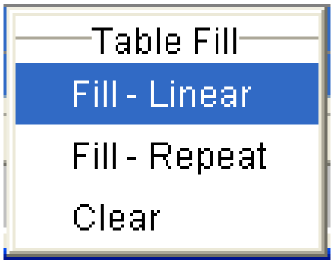


Figure 3.26 Shortcut Menu Table Fill

You can repeat values or linearly interpolate between selected values or delete values by using the shortcut shown in Figure 3.26.

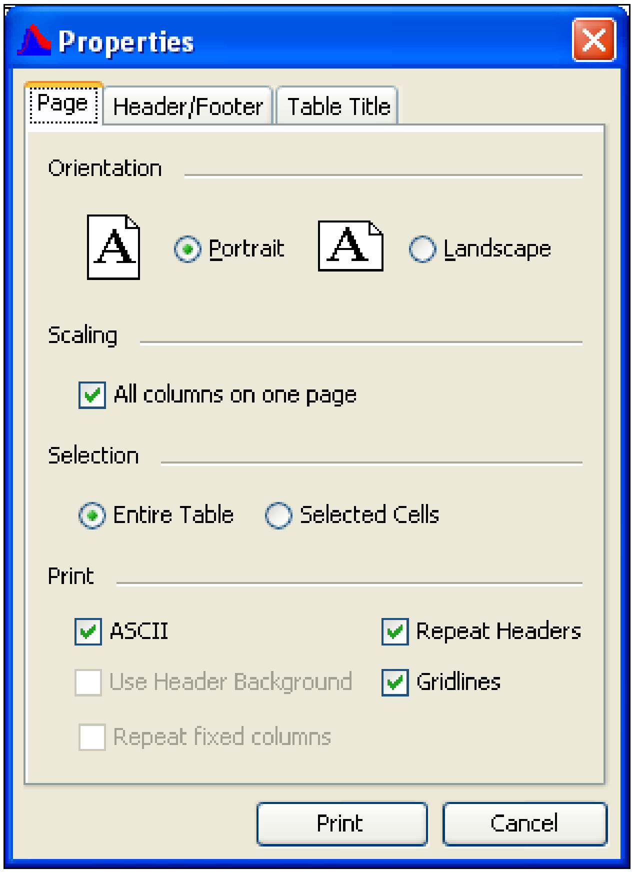
To fill multiple values in the table:

1. Select (highlight) the starting row of the data that you want to fill. For linear and repeat fill, this marks the starting value to use, which will not be changed. If the clear fill option is selected, this is the first value to be cleared.
2. Mark the ending row. You can do this by holding down the left mouse button and moving the mouse down the table, or by left-clicking the mouse on the ending row while holding down the **Shift** key.
3. Right click the mouse button to bring up the popup window in Figure 3.26 and then select either **Fill – Linear**, **Fill – Repeat**, or **Clear**. For **Linear**, the values between the first and last point will be linearly interpolated, with the first and last point remaining the same. For **Repeat**, the first value will be repeated through the last value. For **Clear**, all values within the selected rows will be set to missing.
4. When complete, press the **Accept** button or the **Accept All** button to accept all edits. This will copy the data from the **Estimate/Entry** column into the **Revised** column and will be ready to save to the HEC-DSS file.
5. At the end of all of your edits, press the **Save** or **Save As** button or select that item from the **File** menu to store your changes in the HEC-DSS file.

3.6.5 Printing the Table and Graph

You can print the table or graph by using the **Print Table** or **Print Graph** command under the **Graphical Editor**'s **File** menu.

The **Print Table** command opens the **Properties** dialog (Figure 3.27), which offers options on three tabs.



Orientation

● Portrait ○ Landscape

Scaling

☑ All columns on one page

Selection

● Entire Table ○ Selected Cells

Print

☑ ASCII ☑ Repeat Headers

□ Use Header Background ☑ Gridlines

□ Repeat fixed columns

Figure 3.27 Properties Dialog Box - Printing Tables

The **Page** tab allows you to specify the page **Orientation**, **Scaling**, and **Selection**. You can also choose to print the table as **ASCII**, **Repeat Headers** on every page, and print the **Gridlines**.

On the **Header/Footer** tab, you can type in the header and footer you want to appear on your printed pages.

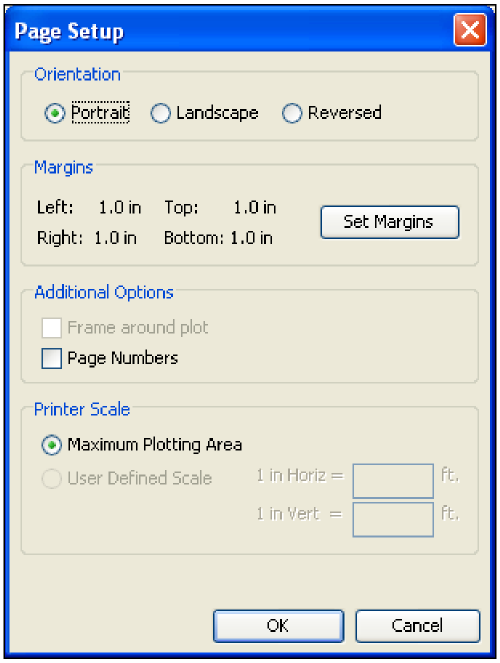
The **Table Title** tab offers a default title for the table based on the data source, which you can change.

When you click **Print**, a standard Windows-style print dialog box opens to select the printer and printer options.

The **Print Graph** command opens a standard Windows-style print dialog box to select the printer and printer options for printing the graph. The printed graph will include the shaded hash area and the current selected row line.

You can set the page settings for the printed graph. To do this, click **Page Setup** from the **File** menu.

From the **File** menu, click **Page Setup**. The **Page Setup** dialog will open (Figure 3.28). Here you can set the page **Orientation**, **Margins**, **Page Numbers**, and **Printer Scale** properties.



Orientation

● Portrait ○ Landscape ○ Reversed

Margins

Left: 1.0 in Top: 1.0 in
Right: 1.0 in Bottom: 1.0 in

Additional Options

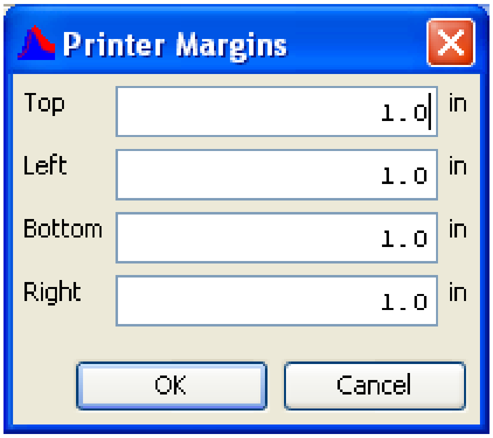
□ Frame around plot
□ Page Numbers

Printer Scale

● Maximum Plotting Area
○ User Defined Scale 1 in Horiz = ft.
1 in Vert = ft.

Figure 3.28 Page Setup Dialog Box

The **Set Margins** button opens the **Printer Margins** dialog (Figure 3.29).



Top 1.0 in
Left 1.0 in
Bottom 1.0 in
Right 1.0 in

Figure 3.29 Printer Margins Dialog Box

3.6.6 Graphical Editor View Options

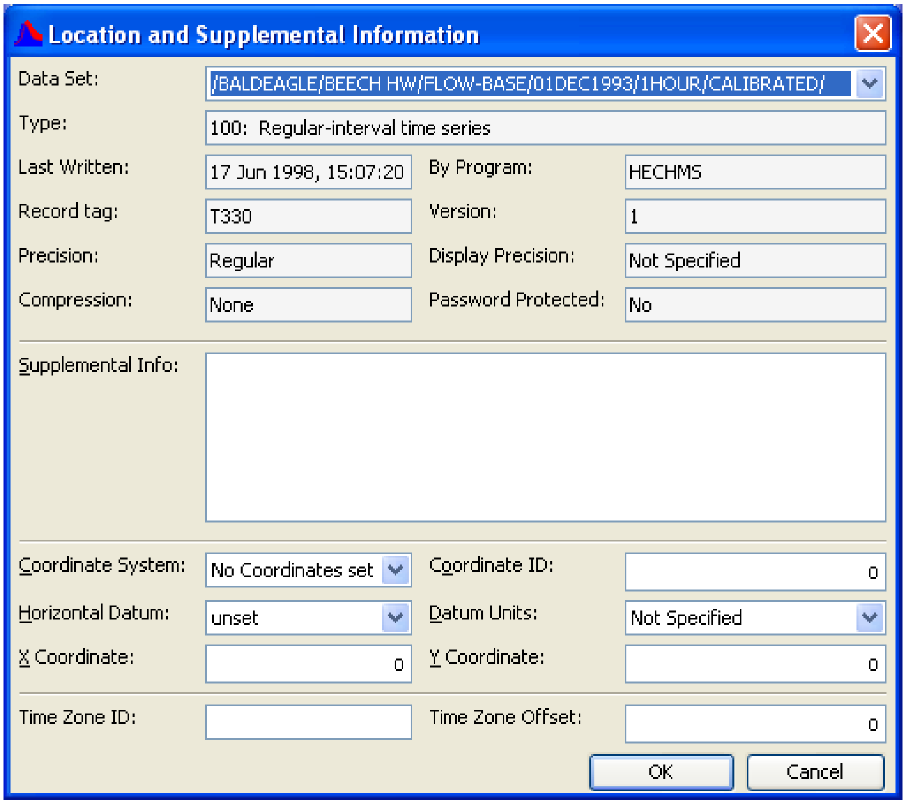
From the **View** menu, you can choose to plot the other data sets that you have selected on the same plot at the same time by clicking **Show Comparison Data**. If the data types are the same, the data sets will plot in the same viewport; otherwise they will be plotted in different viewports.

You cannot edit the comparison data sets. To edit comparison data sets, you must select a data set from the **Selected Data Set** list.

You can also display a standard table of the selected data, from the **View** menu, click **Tabulate**, or to display a standard plot from the **View** menu, click **Show Plot**. You can set the precision of decimal places for your data by pointing to **Decimal Places** and selecting the number of decimal places you wish to display from the list in the submenu.

3.7 Supplemental Information

To view supplemental information, from the DSSVue main window, from the **Display** menu, click **Supplemental Information**, the **Location and Supplemental Information** dialog box will open (Figure 3.30). Supplemental information for a pathname includes the following:



Data Set:	/BALDEAGLE/BEECH HW/FLOW-BASE/01DEC1993/1HOUR/CALIBRATED/		
Type:	100: Regular-interval time series		
Last Written:	17 Jun 1998, 15:07:20	By Program:	HECHMS
Record tag:	T330	Version:	1
Precision:	Regular	Display Precision:	Not Specified
Compression:	None	Password Protected:	No
Supplemental Info:			
Coordinate System:	No Coordinates set	Coordinate ID:	0
Horizontal Datum:	unset	Datum Units:	Not Specified
X Coordinate:	0	Y Coordinate:	0
Time Zone ID:		Time Zone Offset:	0

Figure 3.30 Location and Supplemental Information Dialog Box

Type: The type of the data in the data set, such as regular-interval time series or paired data, etc. The number is the HEC-DSS internal number stored with the data.

Last Written: The date and time the latest data was written.

By Program: Displays the name of the program that last wrote this data set.

Record tag: Tag ID used with legacy programs.

Version: The number of times this data set has been written to.

Precision: Indicates if this data set is single (4 bytes per value) or double precision (8 bytes per value).

Display Precision: Precision used to view the data in tables. This number is the number of digits to the right of the decimal place to show and is automatically set when data is entered.

Compression: Displays the type of compression used if the data record has been compressed.

Password Protected: Displays **Yes** if the data is password protected.

Supplemental Info: Meta-data set in the user header. This can be a variety of information, dependent on the program that stored it. Often, data is given in a key: parameter format.

Coordinate System: Coordinate system identifying the location. Available coordinate systems include: **Lat/long**, **State Plane FIPS**, **State Plane ADS**, **UTM**, or **Local (other)**.

Coordinate ID: The UTM zone number, or FIPS SPEC number or ADS SPCS number.

Horizontal Datum: Horizontal datum can be either: **unset**, **NAD83**, **NAD27**, **WAS84**, **WAS72**, or **Local (other)**.

Datum Units: Units of the datum, including: **Not Specified**, **English**, **SI**, **Decimal Degrees**, or **Degrees Minutes Seconds**.

X Coordinate: Longitude, Easting or decimal degrees (negative for Western Hemisphere).

Y Coordinate: Latitude, Northing or decimal degrees

Time Zone ID: Time zone identifier for this location

Time Zone Offset: Offset from UTC, in milliseconds.

CHAPTER 4

Utilities

With HEC-DSSVue, you can rename, delete, duplicate, and copy data in HEC-DSS files. You can also compare data sets and whole files to determine differences as well as search for values within files.

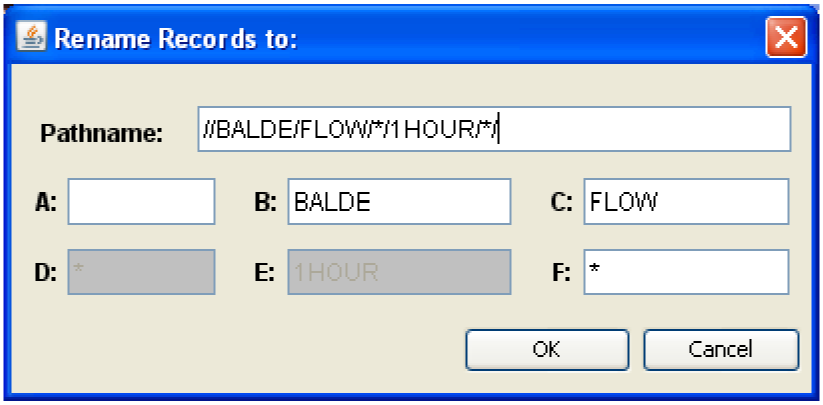
Other utilities include verifying the database integrity, "squeezing" a database to remove unused space, and rebuild tables to make access more efficient.

HEC-DSSVue supports Jython (a Python derivative for Java) scripting to aid in performing repeated tasks and accessing programming level operations.

4.1 Renaming HEC-DSS Data in HEC-DSSVue

The **Rename Records** command opens a pathname editor, which allows you to change the common pathname parts of the selected records. To rename HEC-DSS records:

1. Select the record or records to rename.
2. From the **Edit** menu, select **Rename Records**. The **Rename Records to:** dialog will open, see Figure 4.1.



Rename Records to:

Pathname: //BALDE/FLOW/*/1HOUR/*/

A:

B: BALDE

C: FLOW

D: *

E: 1HOUR

F: *

Figure 4.1 Rename Records to: Dialog Box

3. You may change one or more pathname parts or the entire record. Type the new **Pathname Parts** into the **A:**, **B:**, **C:**, **D:**, **E:**, and **F:** boxes. ***Note:*** *You cannot change the* **D** *or* **E** *parts for time series data (use Math Functions to accomplish this).*

4. For multiple records, pathname parts that are the same for all records will show up in the dialog. Where parts differ (such as having pathnames with C parts of FLOW and ELEVATION), the part will be displayed with an asterisk (*****).
5. When you are finished changing the pathnames, click **OK** to close the **Rename Records to:** dialog and save your changes.

4.2 Deleting Records

When you delete data from a HEC-DSS file, the data is marked as missing but it is not actually removed from the file until you run a **Squeeze** of the file. You can recover deleted records with the **Undelete** command described in the next section. Once a file has been squeezed, records cannot be undeleted. To delete records from a HEC-DSS file:

1. Select the record or records to delete.
2. From the **Edit** menu, select **Delete Records**. A dialog box will confirm that you want to delete those records. If the number of data sets is four or less, their pathnames will be displayed in the dialog.
3. Click **OK**. A confirmation message will appear, stating that the records have been deleted.

4.3 Undoing Deletions

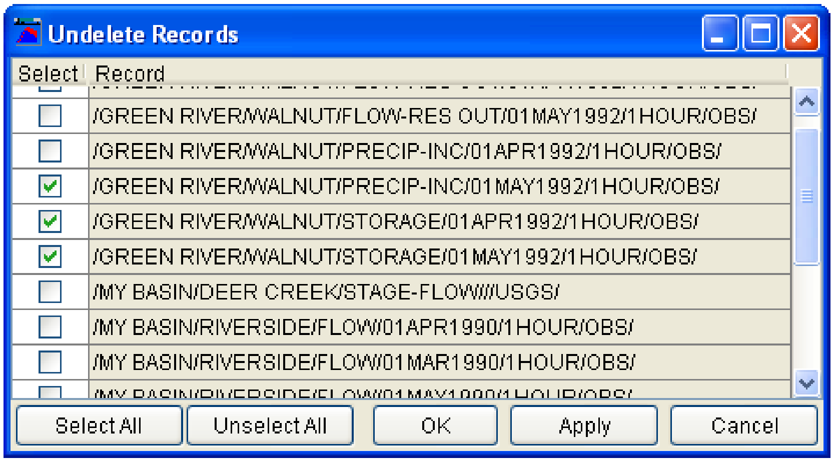
Deleted records may be recovered as long as the HEC-DSS file has not been squeezed. When a file is squeezed, all deleted records are physically removed.

You can undelete records in three ways: undelete all records in the file, select the records to undelete from a list, or undelete the records that you just deleted.

To undelete records from a HEC-DSS file:

1. From the **Edit** menu, point to **Undelete**.
2. To undelete all records click **All**. A dialog box will appear to confirm that you want to undelete all of the records in the file. Click **OK**. A confirmation message will appear that states the records have been undeleted.
3. To undelete the records that you have just deleted click**Last Deleted**. A dialog box will confirm that you want to undelete the records that you just deleted. Click **OK**. A confirmation message will appear that states the records have been undeleted.

4. If you click **Select**, a list of the previously deleted pathnames will open, as shown in Figure 4.2.



Select	Record
□	/GREEN RIVER/WALNUT/FLOW-RES OUT/01MAY1992/1HOUR/OBS/
□	/GREEN RIVER/WALNUT/PRECIP-INC/01APR1992/1HOUR/OBS/
☑	/GREEN RIVER/WALNUT/PRECIP-INC/01MAY1992/1HOUR/OBS/
☑	/GREEN RIVER/WALNUT/STORAGE/01APR1992/1HOUR/OBS/
☑	/GREEN RIVER/WALNUT/STORAGE/01MAY1992/1HOUR/OBS/
□	/MY BASIN/DEER CREEK/STAGE-FLOW///USGS/
□	/MY BASIN/RIVERSIDE/FLOW/01APR1990/1HOUR/OBS/
□	/MY BASIN/RIVERSIDE/FLOW/01MAR1990/1HOUR/OBS/
□	/MY BASIN/RIVERSIDE/FLOW/01MAY1990/1HOUR/OBS/

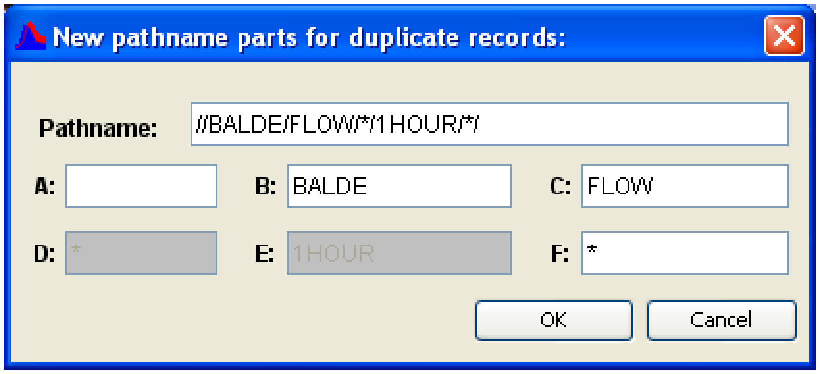
Figure 4.2 Undelete Records Dialog Box

5. Check the boxes next to the records that you want to undelete, or click **Select All** to undelete all records in the list. You can unselect the checked records by clicking **Unselect All**. Once you have made your selection, click **OK** to undelete those records and close the dialog. Alternatively, you can click **Apply** to undelete the records and leave the dialog box open. A confirmation message will appear stating that the records have been undeleted.

4.4 Duplicating Records

The **Duplicate** command duplicates records in the same HEC-DSS file, giving the new records different pathnames. To duplicate records:

1. Select the record or records to duplicate.
2. From the **Edit** menu, click Duplicate Records. The **New pathname parts for duplicate 5ecords** dialog box will open, Figure 4.3.



Pathname: //BALDE/FLOW/*/1HOUR/*/

A:

B: BALDE

C: FLOW

D: *

E: 1HOUR

F: *

Figure 4.3 New pathname parts for duplicate records: Dialog Box

3. Type the new **Pathname Parts** into the **A:**, **B:**, **C:**, **D:**, **E:**, and **F:** boxes. You cannot change the D or E parts for time series data (use Math Functions to accomplish this).
4. For multiple records, pathname parts that are the same for all records will show up in the dialog. Where parts differ (such as having pathnames with C parts of FLOW and ELEVATION), the part will be displayed with an asterisk (*).
5. When you are finished changing the pathnames, click **OK** to close the **New Pathname Parts for Duplicate Records** dialog and save your changes. A confirmation message will appear stating that the records have been duplicated.

4.5 Copying Records into Another HEC-DSS File

To copy records into another HEC-DSS file:

1. Select the record or records to copy.
2. From the **Edit** menu, click **Copy To**. The **Copy Records into HEC-DSS File** browser will open, Figure 4.4.

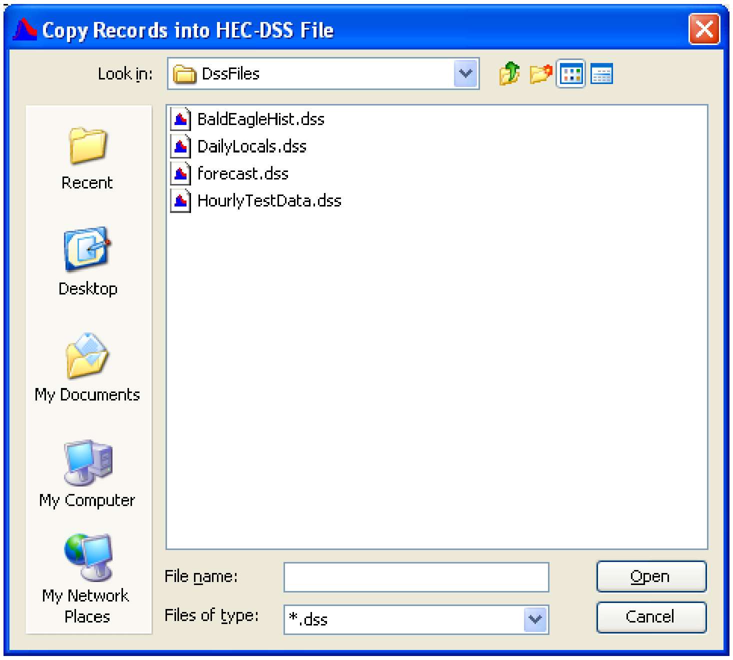
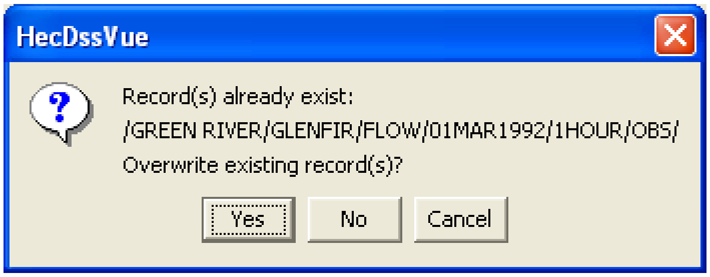


Figure 4.4 Copy Records into HEC-DSS File Browser

3. In the **File name** box, type in a new HEC-DSS filename or select an existing HEC-DSS File you want to copy the record into and click **Open**. A confirmation message will appear stating that the record has been copied to the HEC-DSS file you selected.
4. If the record(s) already exist in that file, a message box will appear asking if you want them to be overwritten, Figure 4.5 (page 4-5).



HecDssVue

Record(s) already exist:
/GREEN RIVER/GLENFIR/FLOW/01MAR1992/1HOUR/OBS/
Overwrite existing record(s)?

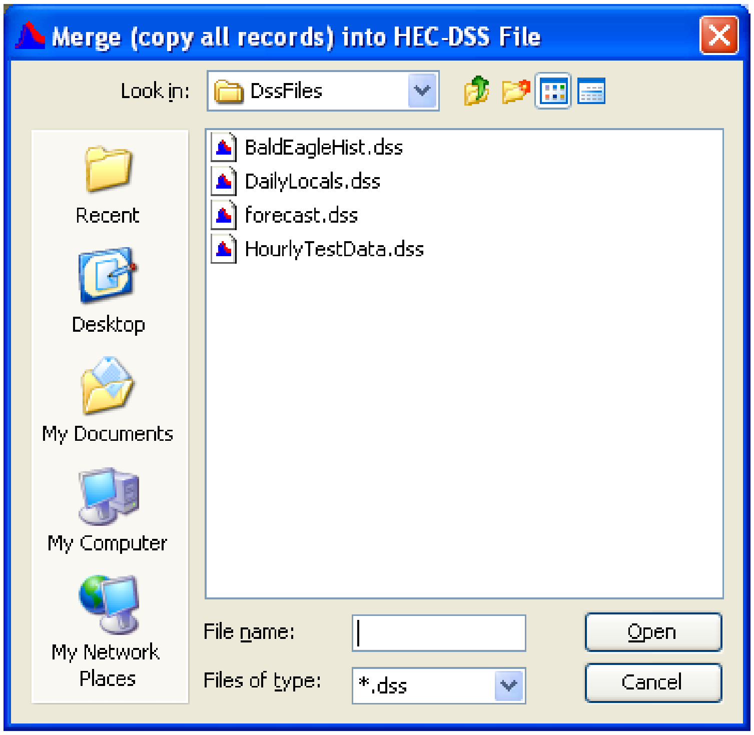
Figure 4.5 Message Box - Overwriting Existing Records

If the number of pathnames is four or less, their names will be displayed in the message box; otherwise they will be printed in the log screen.

4.6 Merging HEC-DSS Files

Merging copies all of the records in the currently opened HEC-DSS file into another. This is similar to selecting all records then using the **Copy Records** option, but is much more efficient. However, this option will overwrite any records with the same pathnames and will not splice together time series records. To merge the current HEC-DSS file into another:

1. From the **Edit** menu, click **Merge (copy)**. The **Merge (copy all records) into HEC-DSS File** browser (Figure 4.6) will open.



Merge (copy all records) into HEC-DSS File

Look in: DssFiles

BaldEagleHist.dss
DailyLocals.dss
forecast.dss
HourlyTestData.dss

File name:
Files of type: *.dss

Figure 4.6 Merge (copy all records) into HEC-DSS File Browser

2. In the **File Name** box, select an existing HEC-DSS File into which you want to copy all of the records into and click **Open**. A confirmation message will appear stating that the records have been copied into the HEC-DSS file you selected.

4.7 Comparing Data Sets

If you have similar time series data sets, you can compare them and find the differences. The Compare functions will tell you how many values are different and highlight them in a table. You can set options in the compare functions to allow a tolerance between values or compare only where there are valid values for the same times.

You can compare data sets from the **Compare data sets** menu option from the **Edit** menu (Figure 4.7) of a table dialog box. Also, from the DSSVue main window, by selecting the pathnames for those data sets and from the **Tools** menu, pointing to **Compare** and selecting on of the available options.

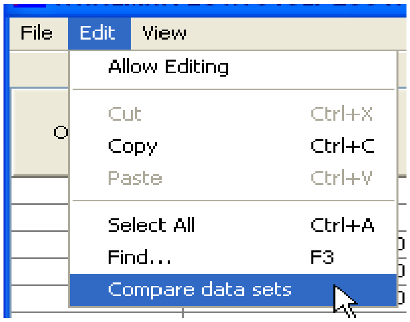
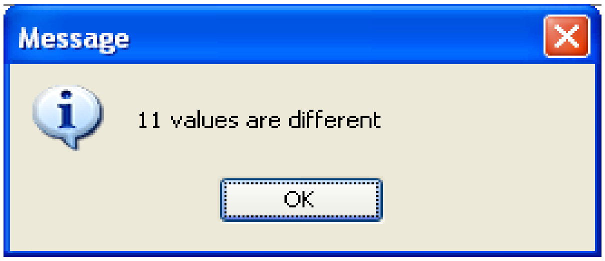


Figure 4.7 Edit Menu

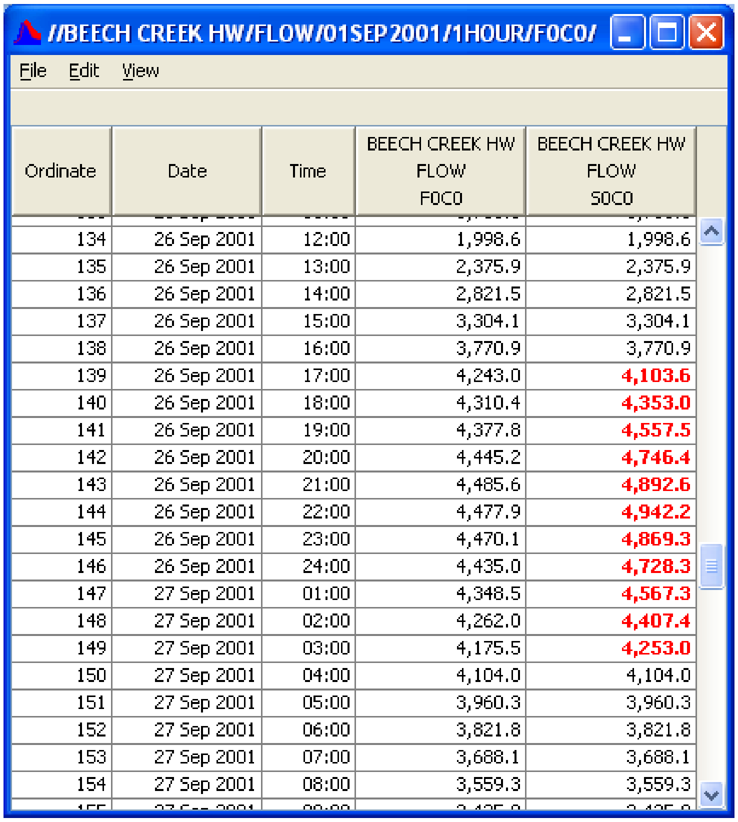
To compare data sets from a table dialog box:

1. At least two pathnames must be selected for comparing data sets to work. Select the records you would like to compare from the **Selected Pathnames List**. Open a table dialog box, from the **Edit** menu, click **Compare data sets**.
2. A message box will open displaying the number of values that differ in the selected data sets (Figure 4.8) and the differences between the two data sets will be highlighted in red, as show in Figure 4.9 (page 4-7).


Message

11 values are different

Figure 4.8 Number of Different Values

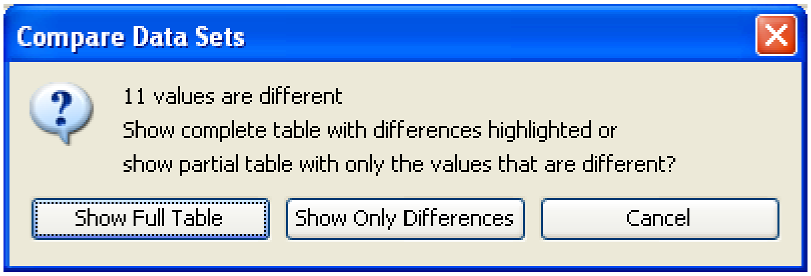


Ordinate	Date	Time	BEECH CREEK HW FLOW F0C0	BEECH CREEK HW FLOW S0C0
134	26 Sep 2001	12:00	1,998.6	1,998.6
135	26 Sep 2001	13:00	2,375.9	2,375.9
136	26 Sep 2001	14:00	2,821.5	2,821.5
137	26 Sep 2001	15:00	3,304.1	3,304.1
138	26 Sep 2001	16:00	3,770.9	3,770.9
139	26 Sep 2001	17:00	4,243.0	**4,103.6**
140	26 Sep 2001	18:00	4,310.4	**4,353.0**
141	26 Sep 2001	19:00	4,377.8	**4,557.5**
142	26 Sep 2001	20:00	4,445.2	**4,746.4**
143	26 Sep 2001	21:00	4,485.6	**4,892.6**
144	26 Sep 2001	22:00	4,477.9	**4,942.2**
145	26 Sep 2001	23:00	4,470.1	**4,869.3**
146	26 Sep 2001	24:00	4,435.0	**4,728.3**
147	27 Sep 2001	01:00	4,348.5	**4,567.3**
148	27 Sep 2001	02:00	4,262.0	**4,407.4**
149	27 Sep 2001	03:00	4,175.5	**4,253.0**
150	27 Sep 2001	04:00	4,104.0	4,104.0
151	27 Sep 2001	05:00	3,960.3	3,960.3
152	27 Sep 2001	06:00	3,821.8	3,821.8
153	27 Sep 2001	07:00	3,688.1	3,688.1
154	27 Sep 2001	08:00	3,559.3	3,559.3

Figure 4.9 Highlight of Differences Between Two Data Sets

To compare data sets from the DSSVue main window:

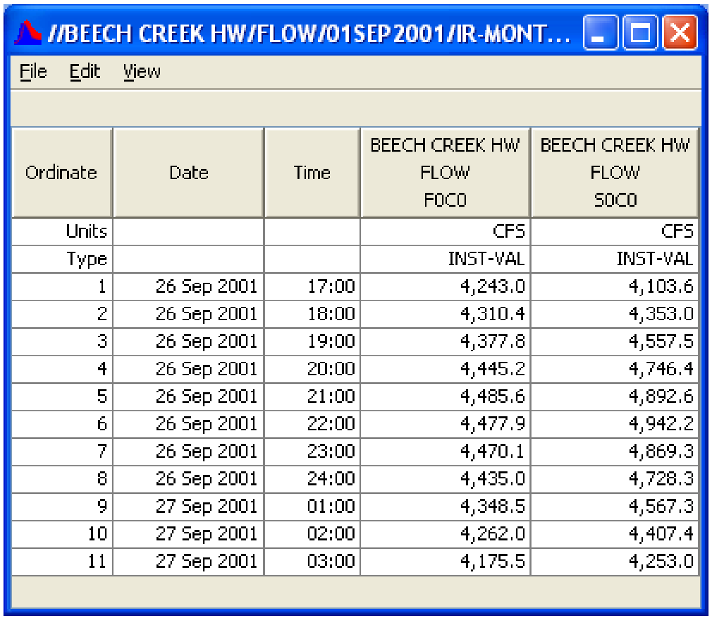
1. At least two pathnames must be selected for comparing data sets to work. Select the records you would like to compare from the **Selected Pathnames List**. From the **Tools** menu, point to **Compare**, click **Data Sets**, the **Compare Data Sets** message box will open (Figure 4.10).



11 values are different
Show complete table with differences highlighted or
show partial table with only the values that are different?

Figure 4.10 Compare Data Sets Message Box

2. To view a complete table of the data sets, click **Show Full Table.** To view a table of only the values that differ, click **Show Only Differences**, as shown in Figure 4.11 (page 4-8).

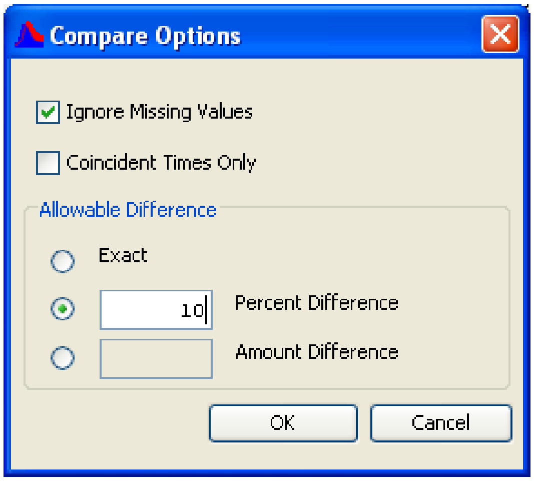


Ordinate	Date	Time	BEECH CREEK HW FLOW F0C0	BEECH CREEK HW FLOW S0C0
Units			CFS	CFS
Type			INST-VAL	INST-VAL
1	26 Sep 2001	17:00	4,243.0	4,103.6
2	26 Sep 2001	18:00	4,310.4	4,353.0
3	26 Sep 2001	19:00	4,377.8	4,557.5
4	26 Sep 2001	20:00	4,445.2	4,746.4
5	26 Sep 2001	21:00	4,485.6	4,892.6
6	26 Sep 2001	22:00	4,477.9	4,942.2
7	26 Sep 2001	23:00	4,470.1	4,869.3
8	26 Sep 2001	24:00	4,435.0	4,728.3
9	27 Sep 2001	01:00	4,348.5	4,567.3
10	27 Sep 2001	02:00	4,262.0	4,407.4
11	27 Sep 2001	03:00	4,175.5	4,253.0

Figure 4.11 Display Only the Differences in the Data Sets

Comparing data sets can be customized to include tolerance levels, ignoring of missing data, and set to compare only values with matching time steps. To use these options, you must compare data sets from the DSSVue main window:

1. At least two pathnames must be selected for comparing data sets to work. Select the records you would like to compare from the **Selected Pathnames List**. From the **Tools** menu, point to **Compare**, click **Data Sets with Options**, the **Compare Options** dialog box will open (Figure 4.12).



☑ Ignore Missing Values

□ Coincident Times Only

Allowable Difference

○ Exact

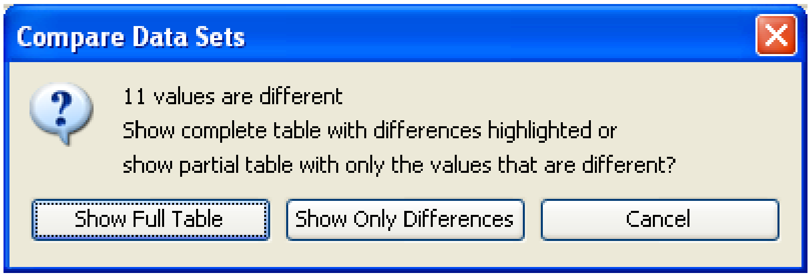
● 10 Percent Difference

○ Amount Difference

Figure 4.12 Compare Options Dialog Box

2. To ignore all missing values during the compare, check **Ignore Missing Values**. To only compare values at match time steps, check **Coincident Times Only**.

3. In the **Allowable Difference** area of the **Compare Options** dialog, you can change the allowable tolerance for the comparison. By default the comparison uses no tolerance; the **Exact** option is selected. Instead of using the exact value, you can choose to allow a percent by selecting **Percent Difference** and entering a percent value from 0-100 in the field. To set a numerical difference, select **Amount Difference** and enter a value in the field.
4. When you are finished selecting **Compare Options**, click **OK**.
5. The **Compare Data Sets** message box will appear displaying the number of values that differ in the selected data sets (Figure 4.13).



11 values are different
Show complete table with differences highlighted or
show partial table with only the values that are different?

Figure 4.13 Compare Data Sets Message Box

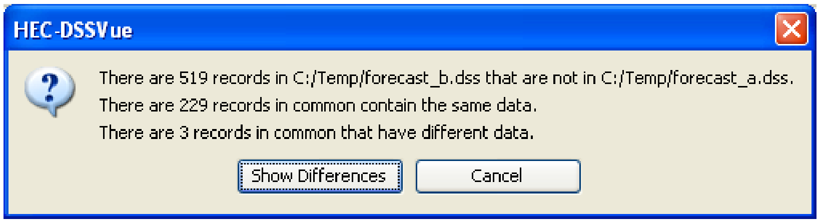
6. To view a table of only the values that differ, click **Show Only Differences**. To view a complete table of the data sets, click **Show Full Table**.

4.8 Comparing HEC-DSS Files

Two HEC-DSS files can be compared to determine the differences of records contained in each file. This is useful in determining if a new version of a program gives the same answers as an older one. Individual or groups of records within the HEC-DSS files can be compared to determine the differences between their data. Options to compare are found under the **Compare** command.

To compare two HEC-DSS Files:

1. The current selected HEC-DSS file, displayed on the selected tab and in the **File Name** field, will be compared to a selected HEC-DSS file you will choose using a file browser. From the **Tools** menu, point to **Compare**, click **Files**, the **Compare to HEC-DSS File** browser open.
2. The **Compare to HEC-DSS File** browser will appear. Use the file browser to select the comparison HEC-DSS file. Once you have selected a file, click **Open**.
3. A message box will display (see Figure 4.14, page 4-10) , summarizing the number of records contained in both of the HEC-

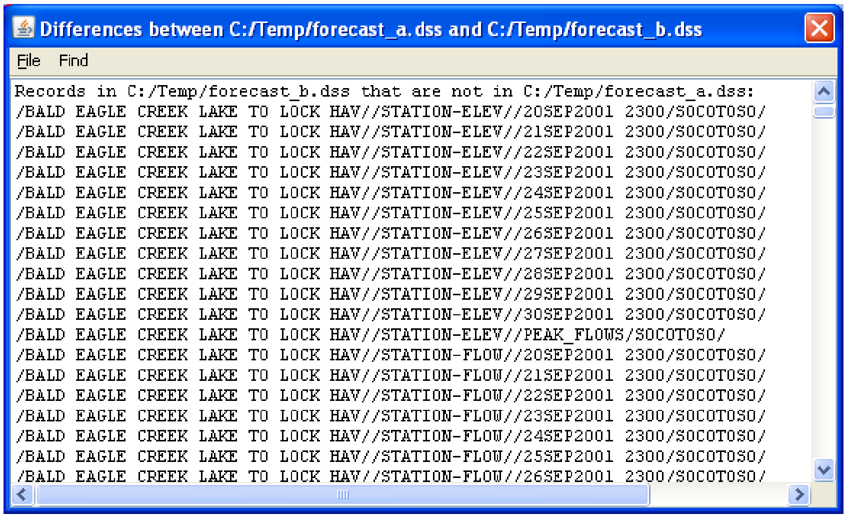


There are 519 records in C:/Temp/forecast_b.dss that are not in C:/Temp/forecast_a.dss.
There are 229 records in common contain the same data.
There are 3 records in common that have different data.

Figure 4.14 File Comparison Result Message Box

DSS files. To view a list of the different records, click **Show Differences**.

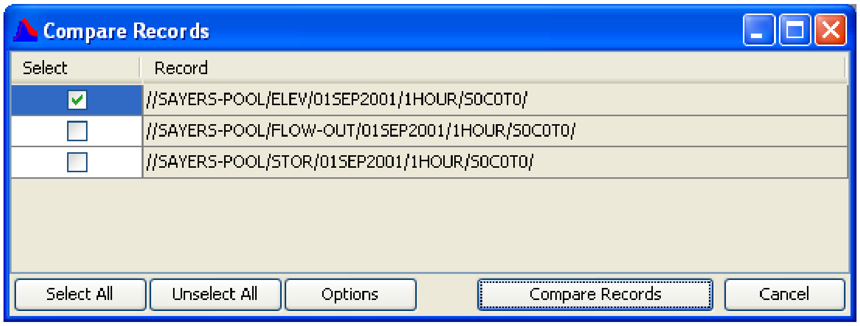
4. The **Differences** window will appear listing all the records that are different between the two files as in Figure 4.15.


Differences between C:/Temp/forecast_a.dss and C:/Temp/forecast_b.dss
File
Find

```
Records in C:/Temp/forecast_b.dss that are not in C:/Temp/forecast_a.dss:
/BALD EAGLE CREEK LAKE TO LOCK HAV//STATION-ELEV//20SEP2001 2300/S0C0T0S0/
/BALD EAGLE CREEK LAKE TO LOCK HAV//STATION-ELEV//21SEP2001 2300/S0C0T0S0/
/BALD EAGLE CREEK LAKE TO LOCK HAV//STATION-ELEV//22SEP2001 2300/S0C0T0S0/
/BALD EAGLE CREEK LAKE TO LOCK HAV//STATION-ELEV//23SEP2001 2300/S0C0T0S0/
/BALD EAGLE CREEK LAKE TO LOCK HAV//STATION-ELEV//24SEP2001 2300/S0C0T0S0/
/BALD EAGLE CREEK LAKE TO LOCK HAV//STATION-ELEV//25SEP2001 2300/S0C0T0S0/
/BALD EAGLE CREEK LAKE TO LOCK HAV//STATION-ELEV//26SEP2001 2300/S0C0T0S0/
/BALD EAGLE CREEK LAKE TO LOCK HAV//STATION-ELEV//27SEP2001 2300/S0C0T0S0/
/BALD EAGLE CREEK LAKE TO LOCK HAV//STATION-ELEV//28SEP2001 2300/S0C0T0S0/
/BALD EAGLE CREEK LAKE TO LOCK HAV//STATION-ELEV//29SEP2001 2300/S0C0T0S0/
/BALD EAGLE CREEK LAKE TO LOCK HAV//STATION-ELEV//30SEP2001 2300/S0C0T0S0/
/BALD EAGLE CREEK LAKE TO LOCK HAV//STATION-ELEV//PEAK_FLOWS/S0C0T0S0/
/BALD EAGLE CREEK LAKE TO LOCK HAV//STATION-FLOW//20SEP2001 2300/S0C0T0S0/
/BALD EAGLE CREEK LAKE TO LOCK HAV//STATION-FLOW//21SEP2001 2300/S0C0T0S0/
/BALD EAGLE CREEK LAKE TO LOCK HAV//STATION-FLOW//22SEP2001 2300/S0C0T0S0/
/BALD EAGLE CREEK LAKE TO LOCK HAV//STATION-FLOW//23SEP2001 2300/S0C0T0S0/
/BALD EAGLE CREEK LAKE TO LOCK HAV//STATION-FLOW//24SEP2001 2300/S0C0T0S0/
/BALD EAGLE CREEK LAKE TO LOCK HAV//STATION-FLOW//25SEP2001 2300/S0C0T0S0/
/BALD EAGLE CREEK LAKE TO LOCK HAV//STATION-FLOW//26SEP2001 2300/S0C0T0S0/
```

Figure 4.15 Comparing Files - List of records that are different between files

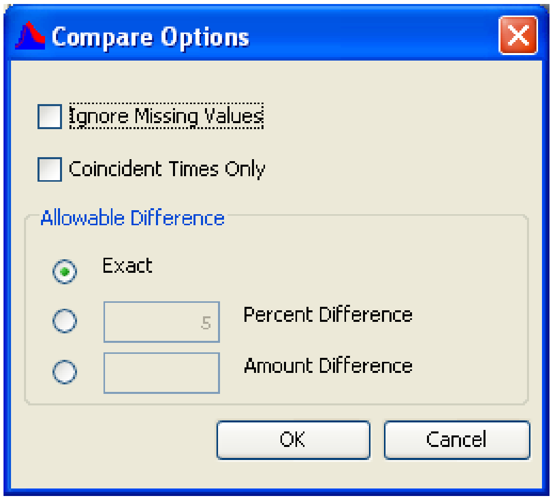
5. The **Compare Records** dialog box will open (Figure 4.16) that lists records in common between the two files that contain different values. If you want to set a tolerance to compare values with, press the **Options** button.



Select	Record
☑	//SAYERS-POOL/ELEV/01SEP2001/1HOUR/S0C0T0/
☐	//SAYERS-POOL/FLOW-OUT/01SEP2001/1HOUR/S0C0T0/
☐	//SAYERS-POOL/STOR/01SEP2001/1HOUR/S0C0T0/

Figure 4.16 Compare Record Dialog Box

6. The **Compare Options** dialog box (see Figure 4.17, page 4-11) allows you to set options to compare values by ignoring missing values (where both values are non-missing) for the same times,


Compare Options

☐ Ignore Missing Values

☐ Coincident Times Only

Allowable Difference

● Exact

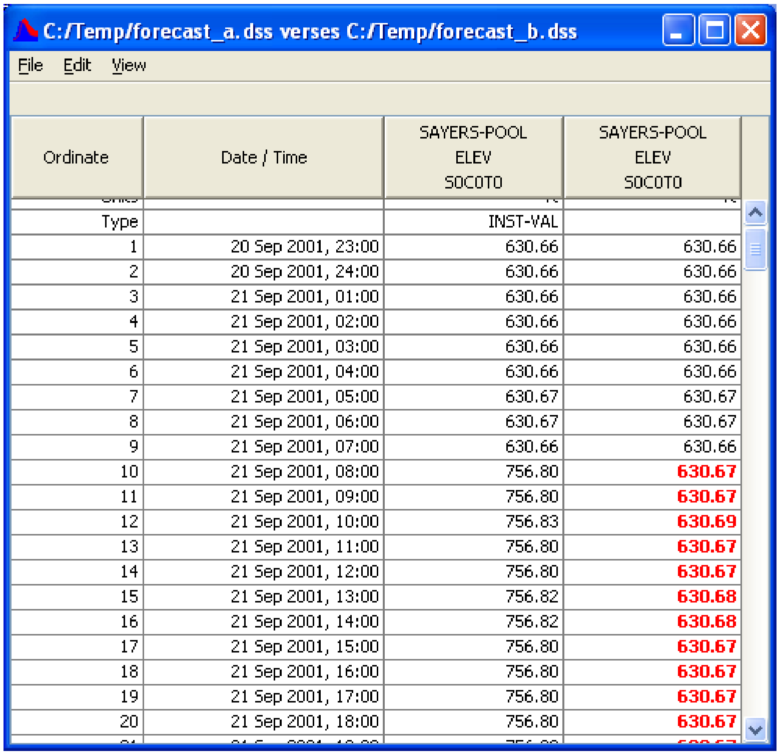
○ 5 Percent Difference

○ Amount Difference

Figure 4.17 Compare Options Dialog Box

compare values for coincident times only (where there are valid values in both data sets at the same time), and set a tolerance to compare between numbers. For the tolerance, you can set a percent amount that values can differ by or an absolute value that they can differ.

7. When you select the records that you want to compare and press the **Compare Records** button, a table showing the differences for each pair will be displayed. Values that are different are highlighted in red for the file that you are comparing to, as shown in Figure 4.18.


C:/Temp/forecast_a.dss verses C:/Temp/forecast_b.dss

Ordinate	Date / Time	SAYERS-POOL ELEV 50C0T0	SAYERS-POOL ELEV 50C0T0
Type		INST-VAL	
1	20 Sep 2001, 23:00	630.66	630.66
2	20 Sep 2001, 24:00	630.66	630.66
3	21 Sep 2001, 01:00	630.66	630.66
4	21 Sep 2001, 02:00	630.66	630.66
5	21 Sep 2001, 03:00	630.66	630.66
6	21 Sep 2001, 04:00	630.66	630.66
7	21 Sep 2001, 05:00	630.67	630.67
8	21 Sep 2001, 06:00	630.67	630.67
9	21 Sep 2001, 07:00	630.66	630.66
10	21 Sep 2001, 08:00	756.80	**630.67**
11	21 Sep 2001, 09:00	756.80	**630.67**
12	21 Sep 2001, 10:00	756.83	**630.69**
13	21 Sep 2001, 11:00	756.80	**630.67**
14	21 Sep 2001, 12:00	756.80	**630.67**
15	21 Sep 2001, 13:00	756.82	**630.68**
16	21 Sep 2001, 14:00	756.82	**630.68**
17	21 Sep 2001, 15:00	756.80	**630.67**
18	21 Sep 2001, 16:00	756.80	**630.67**
19	21 Sep 2001, 17:00	756.80	**630.67**
20	21 Sep 2001, 18:00	756.80	**630.67**

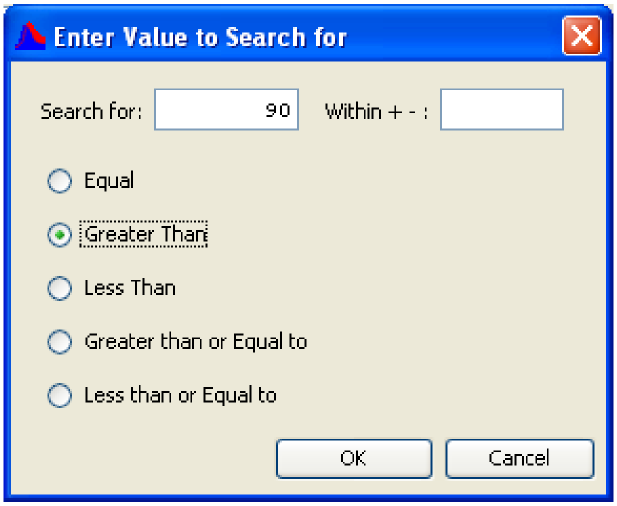
Figure 4.18 Comparison of Data Sets

4.9 Searching for a Value

You can search for a value in time series and paired data sets. You can set a tolerance to search within or you can search for values less than or greater than a specified number.

You can tabulate the records that match your specifications. Those values will be highlighted in red. To search for a value:

1. Select the data sets that you want to search. From the **Tools** menu, click **Search for Value**, the **Enter Value to Search for** dialog box will open (Figure 4.19).



Enter Value to Search for

Search for: 90 Within + - :

○ Equal

● Greater Than

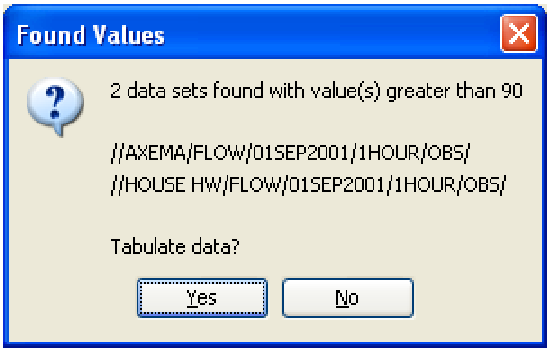
○ Less Than

○ Greater than or Equal to

○ Less than or Equal to

Figure 4.19 Enter Value to Search for Dialog Box

2. From the **Enter Value to Search for** dialog, enter the value you would like to find in the **Search for** box. If you would like to search within a tolerance for this value, enter the range in the **Within +-** box.
3. By default, the search is set to find the exact value, give or take the **Within** value. However, you can choose to search for values **Greater Than**, **Less Than**, **Greater than or Equal to**, or **Less than or Equal to** the Search for value.
4. Once you are finished selecting your search criteria, click **OK**.
5. The **Found Values** message box will appear (Figure 4.20) listing all selected data sets that meet your search criteria.



Found Values

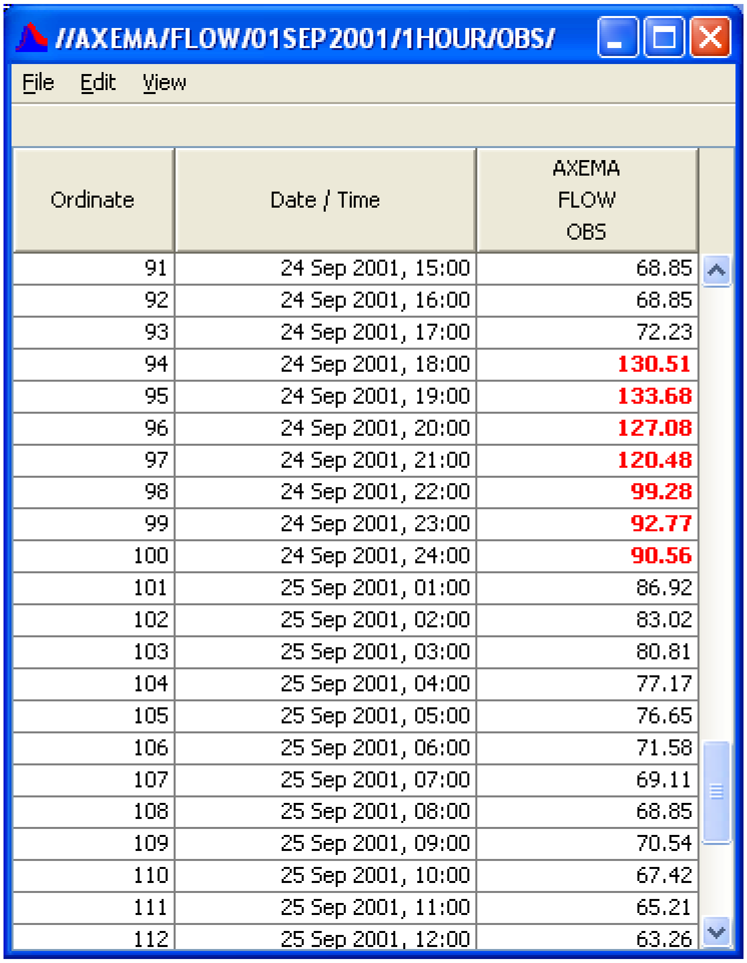
2 data sets found with value(s) greater than 90

//AXEMA/FLOW/01SEP2001/1HOUR/OBS/
//HOUSE HW/FLOW/01SEP2001/1HOUR/OBS/

Tabulate data?

Figure 4.20 Found Values Message Box

6. To tabulate the search findings, click **Yes** on the **Found Values** window. A table will display listing all values that meet the search criteria in red, see Figure 4.21 below.



Ordinate	Date / Time	AXEMA FLOW OBS
91	24 Sep 2001, 15:00	68.85
92	24 Sep 2001, 16:00	68.85
93	24 Sep 2001, 17:00	72.23
94	24 Sep 2001, 18:00	**130.51**
95	24 Sep 2001, 19:00	**133.68**
96	24 Sep 2001, 20:00	**127.08**
97	24 Sep 2001, 21:00	**120.48**
98	24 Sep 2001, 22:00	**99.28**
99	24 Sep 2001, 23:00	**92.77**
100	24 Sep 2001, 24:00	**90.56**
101	25 Sep 2001, 01:00	86.92
102	25 Sep 2001, 02:00	83.02
103	25 Sep 2001, 03:00	80.81
104	25 Sep 2001, 04:00	77.17
105	25 Sep 2001, 05:00	76.65
106	25 Sep 2001, 06:00	71.58
107	25 Sep 2001, 07:00	69.11
108	25 Sep 2001, 08:00	68.85
109	25 Sep 2001, 09:00	70.54
110	25 Sep 2001, 10:00	67.42
111	25 Sep 2001, 11:00	65.21
112	25 Sep 2001, 12:00	63.26

Figure 4.21 Found Values from Search

4.10 Checking File Integrity

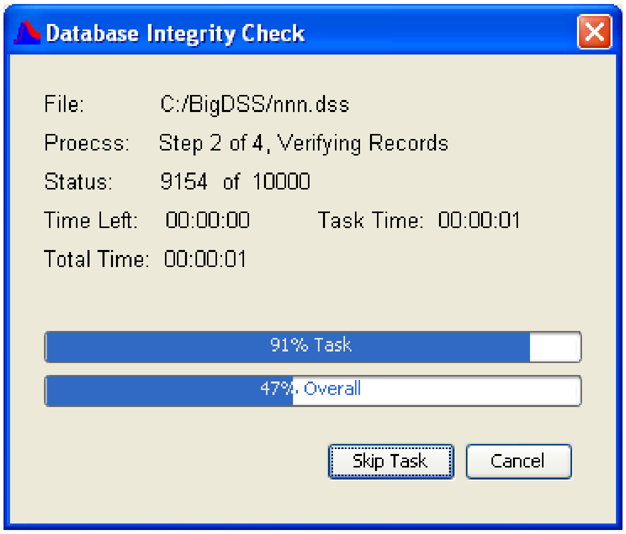
You can check to see if a database has been damaged. Generally, this is rare but occasionally a file can be damaged from disk problems, errant programs lost file links or other situations. Checking file integrity is similar to a "Check Disk" in Microsoft Windows.

In addition to checking for damage, checking file integrity will tell you if you can improve efficiency by squeezing the database. However, it is usually faster just to squeeze the database than to check for it.

Checking file integrity is a thorough and comprehensive process, so it will take time.

To check the integrity of an HEC-DSS file:

1. Open an HEC-DSS file, from the **Tools** menu, click **Check File Integrity**.
2. For larger files, the Database Integrity Check dialog box (Figure 4.22) will open providing the status of the operation. There are four major steps in the operation. You can cancel at anytime without hurting anything. Generally, the process goes faster once the first step has been completed.


Database Integrity Check

File: C:/BigDSS/nnn.dss

Proecss: Step 2 of 4, Verifying Records

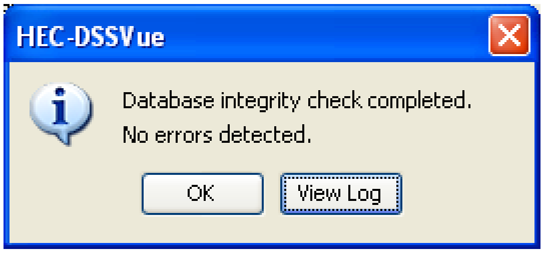
Status: 9154 of 10000

Time Left: 00:00:00 Task Time: 00:00:01

Total Time: 00:00:01

Figure 4.22 Database Integrity Check Dialog Box

3. A status report message window will open (Figure 4.23) after the integrity check has finished. To view the integrity report, click **View Log**.

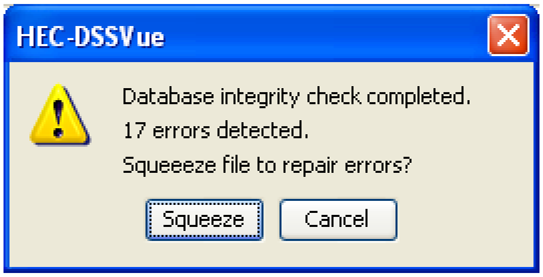

HEC-DSSVue

Database integrity check completed.
No errors detected.

Figure 4.23 Status Report Message Window

4. If any errors in the database are detected, a message window saying so will be displayed (see Figure 4.24, page 4-15). Due to the comprehensive nature of the check, an error is usually reported for several processes within the check, so the total number of errors reported can be substantially more than there really are.

   If an error is detected, it is recommended that you make a backup of your file and then squeeze the file. Squeezing rebuilds the file, using a brute force approach that should retrieve all data possible

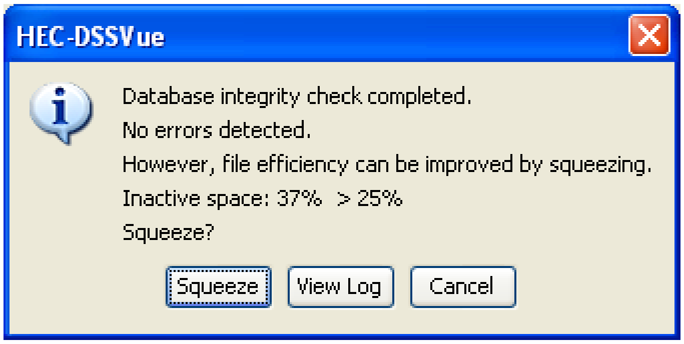


Database integrity check completed.
17 errors detected.
Squeeeze file to repair errors?

Figure 4.24 Error from Database Integrity Check

within the file. You should review any critical data after you have squeezed it, as the same event that broke links within the file may have changed data values also.

5. If you have a lot of unused space due to deletions or other actions or the internal tables are not suited well to the number of data sets, a message will be displayed saying so. (Figure 4.25) It is recommended that you squeeze the file to remove unused space and rebuild the internal tables.



Database integrity check completed.
No errors detected.
However, file efficiency can be improved by squeezing.
Inactive space: 37% > 25%
Squeeze?

Figure 4.25 Status Report Message After Data Sets are Deleted

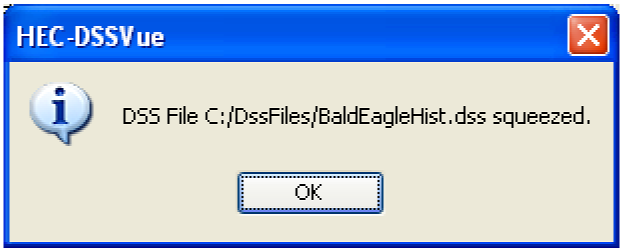
4.11 Squeezing an HEC-DSS File

When you delete or rename records, a HEC-DSS file will accumulate inactive space. The **Squeeze** command removes inactive space by copying all valid data to a new file then renaming the new file to the old filename. The **Squeeze** command will also automatically re-adjust internal HEC-DSS table sizes to optimize access to the data. This is similar to de-fragmenting your file system. Once a squeeze has been accomplished, deleted data cannot be recovered.

To squeeze an HEC-DSS file:

1. Open an HEC-DSS file, from the **Tools** menu, click **Squeeze**. A window will appear indicating the status of the squeeze process.
2. If you wish, you may cancel the squeeze prior to completion, by clicking **Cancel**.

3. When the process is complete, a confirmation will appear (Figure 4.26).

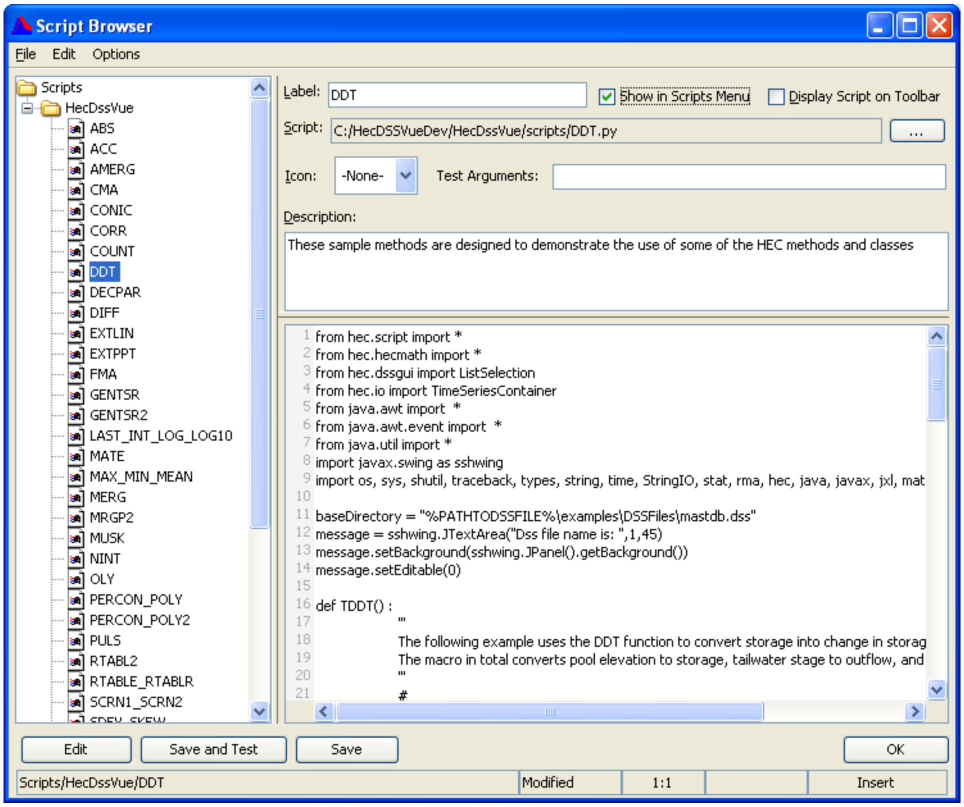


DSS File C:/DssFiles/BaldEagleHist.dss squeezed.

Figure 4.26 Squeeze Confirmation Window

4.12 Scripting

You can use Jython scripts that you or others write to perform math functions, create custom plots or tables, and automate repetitive tasks. To edit, test, and select scripts, use the **Script Editor** (Figure 4.27). To access the **Script Editor**, from the **Tools** menu, click **Script Editor**.



Label: DDT ☑ Show in Scripts Menu ☐ Display Script on Toolbar

Script: C:/HecDSSVueDev/HecDssVue/scripts/DDT.py

Icon: -None- Test Arguments:

Description:

These sample methods are designed to demonstrate the use of some of the HEC methods and classes

```
from hec.script import *
from hec.hecmath import *
from hec.dssgui import ListSelection
from hec.io import TimeSeriesContainer
from java.awt import  *
from java.awt.event import  *
from java.util import *
import javax.swing as sshwing
import os, sys, shutil, traceback, types, string, time, StringIO, stat, rma, hec, java, javax, jxl, mat

baseDirectory = "%PATHTODSSFILE%\examples\DSSFiles\mastdb.dss"
message = sshwing.JTextArea("Dss file name is: ",1,45)
message.setBackground(sshwing.JPanel().getBackground())
message.setEditable(0)

def TDDT() :
                '''
                The following example uses the DDT function to convert storage into change in storag
                The macro in total converts pool elevation to storage, tailwater stage to outflow, and
                '''
                #
```

Scripts/HecDssVue/DDT | Modified | 1:1 | Insert

Figure 4.27 Script Editor

To run scripts that have already been setup, use the **Script Selector** dialog box (Figure 4.28). To access the **Script Selector** dialog box, from the **Tools** menu, click **Script Selector**. For more information, refer to the chapter on **Scripting**.

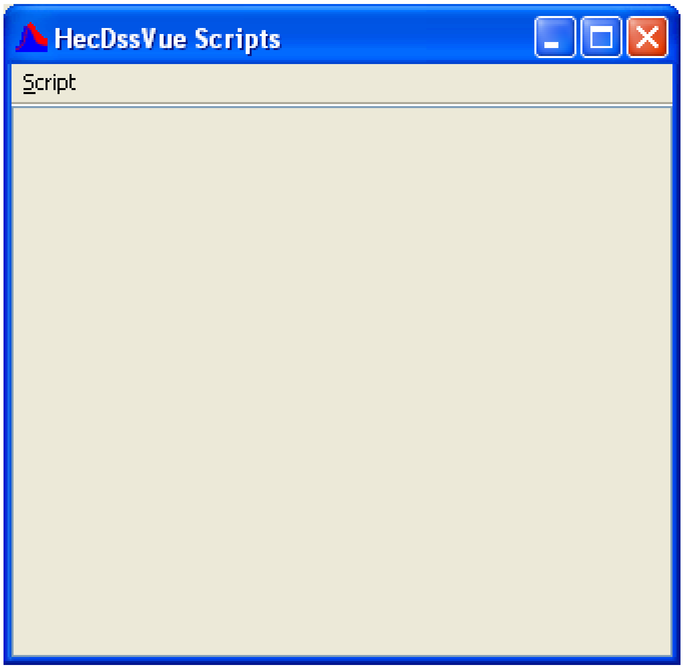


Figure 4.28 Script Selector Dialog Box

4.13 Catalogs

The list of pathnames on the main screen is a modified version of the HEC-DSS Catalog. A catalog is essentially a list of record pathnames in a DSS file, organized in one of several ways. Because a catalog is separate and not really part of the DSS file, it only reflects the state of the DSS file the last time it was updated. Whenever you add a new record to the database, the catalog becomes out of date; the catalog is not updated as it would be prohibitively slow.

The new version of HEC-DSSVue retrieves an internal listing of pathnames to show in the main window. However, since it takes time to sort and order them, a cache disk file is usually saved, and only updated when it is detected to be out of date. For larger DSS files, HEC-DSSVue will ask the user if they want to spend the time to have it updated when it becomes out of date.

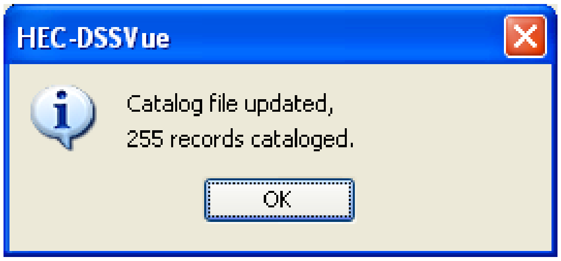
This section covers the three primary forms of catalogs that can be stored on disk and viewed or printed. A full catalog is a listing of all record pathnames in the file, along with their last written date and time, their version, the number of data they contain and the type of data it contains. An abbreviated catalog contains just the record pathnames and a condensed catalog groups dates for time series data and just shows a date span for the date part of a time series data set.

4.13.1 Condensed Disk Catalog

A condensed catalog is a list of pathnames where the dates of time series data are grouped together and show a date span instead of a date D part for each record.

To create a condensed catalog:

1. Open an HEC-DSS file, from the **Advanced** menu, point to **Condensed Disk Catalog**, click **New**.
2. A message box will appear stating the catalog has been updated and the number of records it contained (Figure 4.29).

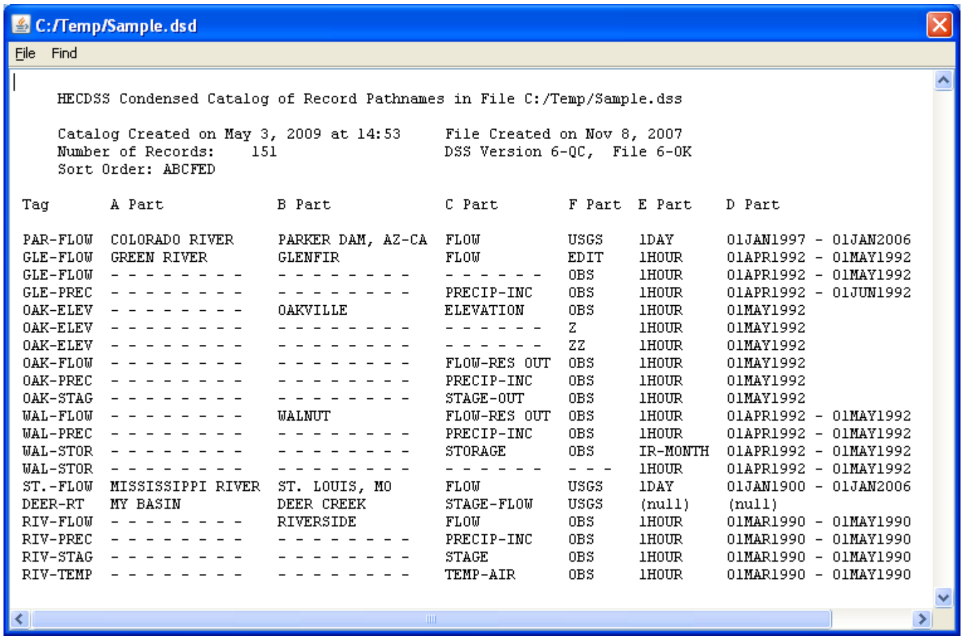


Catalog file updated,
255 records cataloged.

Figure 4.29 Catalog File Update Message Box

To view a condensed disk catalog:

1. Open an HEC-DSS file, from the **Advanced** menu, point to **Condensed Disk Catalog**, click **View**.
2. A text dialog box will open (Figure 4.30) displaying the condensed disk catalog.



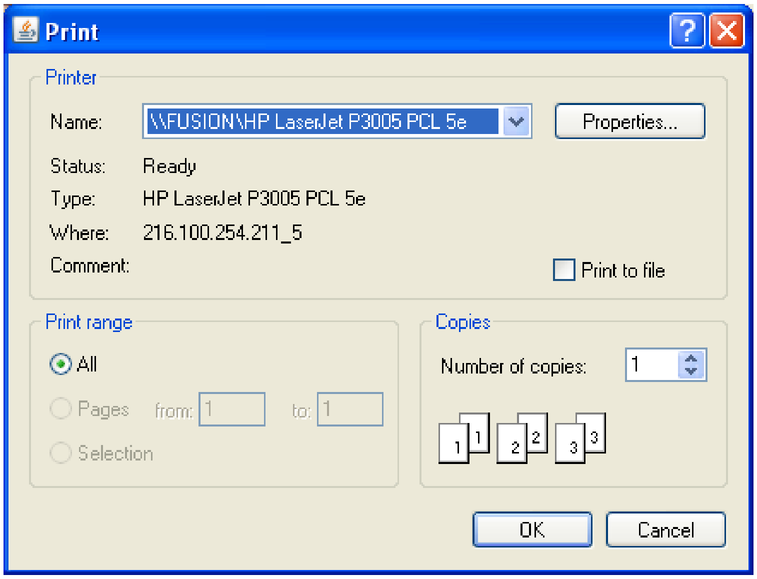
```
     HECDSS Condensed Catalog of Record Pathnames in File C:/Temp/Sample.dss

     Catalog Created on May 3, 2009 at 14:53    File Created on Nov 8, 2007
     Number of Records:    151                  DSS Version 6-QC,  File 6-OK
     Sort Order: ABCFED

 Tag       A Part             B Part             C Part        F Part  E Part    D Part

 PAR-FLOW  COLORADO RIVER     PARKER DAM, AZ-CA  FLOW          USGS    1DAY      01JAN1997 - 01JAN2006
 GLE-FLOW  GREEN RIVER        GLENFIR            FLOW          EDIT    1HOUR     01APR1992 - 01MAY1992
 GLE-FLOW  - - - - - - - -    - - - - - - - -    - - - - - -   OBS     1HOUR     01APR1992 - 01MAY1992
 GLE-PREC  - - - - - - - -    - - - - - - - -    PRECIP-INC    OBS     1HOUR     01APR1992 - 01JUN1992
 OAK-ELEV  - - - - - - - -    OAKVILLE           ELEVATION     OBS     1HOUR     01MAY1992
 OAK-ELEV  - - - - - - - -    - - - - - - - -    - - - - - -   Z       1HOUR     01MAY1992
 OAK-ELEV  - - - - - - - -    - - - - - - - -    - - - - - -   ZZ      1HOUR     01MAY1992
 OAK-FLOW  - - - - - - - -    - - - - - - - -    FLOW-RES OUT  OBS     1HOUR     01MAY1992
 OAK-PREC  - - - - - - - -    - - - - - - - -    PRECIP-INC    OBS     1HOUR     01MAY1992
 OAK-STAG  - - - - - - - -    - - - - - - - -    STAGE-OUT     OBS     1HOUR     01MAY1992
 WAL-FLOW  - - - - - - - -    WALNUT             FLOW-RES OUT  OBS     1HOUR     01APR1992 - 01MAY1992
 WAL-PREC  - - - - - - - -    - - - - - - - -    PRECIP-INC    OBS     1HOUR     01APR1992 - 01MAY1992
 WAL-STOR  - - - - - - - -    - - - - - - - -    STORAGE       OBS     IR-MONTH  01APR1992 - 01MAY1992
 WAL-STOR  - - - - - - - -    - - - - - - - -    - - - - - -   - - -   1HOUR     01APR1992 - 01MAY1992
 ST.-FLOW  MISSISSIPPI RIVER  ST. LOUIS, MO      FLOW          USGS    1DAY      01JAN1900 - 01JAN2006
 DEER-RT   MY BASIN           DEER CREEK         STAGE-FLOW    USGS    (null)    (null)
 RIV-FLOW  - - - - - - - -    RIVERSIDE          FLOW          OBS     1HOUR     01MAR1990 - 01MAY1990
 RIV-PREC  - - - - - - - -    - - - - - - - -    PRECIP-INC    OBS     1HOUR     01MAR1990 - 01MAY1990
 RIV-STAG  - - - - - - - -    - - - - - - - -    STAGE         OBS     1HOUR     01MAR1990 - 01MAY1990
 RIV-TEMP  - - - - - - - -    - - - - - - - -    TEMP-AIR      OBS     1HOUR     01MAR1990 - 01MAY1990
```

Figure 4.30 Condensed Catalog Text Output

3. You can print the catalog from the text dialog, from **File** menu, click **Print**.
4. From the text dialog, you can perform several file management functions: **Insert .txt**, **Save As .txt**, change the **Font**, and **Print Setup**. It also contains a **Find** menu that can facilitate a search of the catalog displayed.

To print a condensed disk catalog:

1. Open an HEC-DSS file, from the **Advanced** menu, point to **Condensed Disk Catalog**, click **Print**.
2. A **Print** dialog box will open (Figure 4.31); click **OK** to send the catalog to the printer.



Printer

Name: \\FUSION\HP LaserJet P3005 PCL 5e

Status: Ready

Type: HP LaserJet P3005 PCL 5e

Where: 216.100.254.211_5

Comment:

□ Print to file

Print range

● All

○ Pages from: 1 to: 1

○ Selection

Copies

Number of copies: 1

Figure 4.31 Print Dialog Box

4.13.2 Abbreviated Disk Catalog

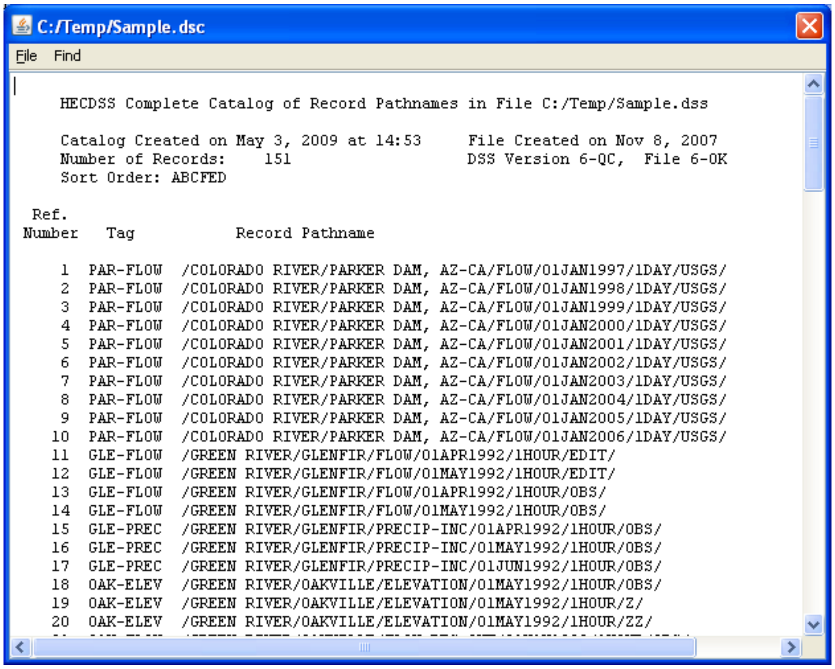
An abbreviated catalog is a sorted list of record pathnames in a HEC-DSS file.

To create an abbreviated catalog:

1. Open an HEC-DSS file, from the **Advanced** menu, point to **Abbreviated Disk Catalog**, click **New**.
2. A message box will appear stating the catalog has been updated and the number of records it contained (see Figure 4.29, page 4-18).

To view an abbreviated catalog:

1. Open an HEC-DSS file, from the **Advanced** menu, point to **Abbreviated Disk Catalog**, click **View**.
2. A text dialog box will open (Figure 4.32) displaying the abbreviated disk catalog.



```
     HECDSS Complete Catalog of Record Pathnames in File C:/Temp/Sample.dss

     Catalog Created on May 3, 2009 at 14:53     File Created on Nov 8, 2007
     Number of Records:    151                   DSS Version 6-QC,  File 6-OK
     Sort Order: ABCFED

 Ref.
Number   Tag           Record Pathname

     1  PAR-FLOW  /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN1997/1DAY/USGS/
     2  PAR-FLOW  /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN1998/1DAY/USGS/
     3  PAR-FLOW  /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN1999/1DAY/USGS/
     4  PAR-FLOW  /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN2000/1DAY/USGS/
     5  PAR-FLOW  /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN2001/1DAY/USGS/
     6  PAR-FLOW  /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN2002/1DAY/USGS/
     7  PAR-FLOW  /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN2003/1DAY/USGS/
     8  PAR-FLOW  /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN2004/1DAY/USGS/
     9  PAR-FLOW  /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN2005/1DAY/USGS/
    10  PAR-FLOW  /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN2006/1DAY/USGS/
    11  GLE-FLOW  /GREEN RIVER/GLENFIR/FLOW/01APR1992/1HOUR/EDIT/
    12  GLE-FLOW  /GREEN RIVER/GLENFIR/FLOW/01MAY1992/1HOUR/EDIT/
    13  GLE-FLOW  /GREEN RIVER/GLENFIR/FLOW/01APR1992/1HOUR/OBS/
    14  GLE-FLOW  /GREEN RIVER/GLENFIR/FLOW/01MAY1992/1HOUR/OBS/
    15  GLE-PREC  /GREEN RIVER/GLENFIR/PRECIP-INC/01APR1992/1HOUR/OBS/
    16  GLE-PREC  /GREEN RIVER/GLENFIR/PRECIP-INC/01MAY1992/1HOUR/OBS/
    17  GLE-PREC  /GREEN RIVER/GLENFIR/PRECIP-INC/01JUN1992/1HOUR/OBS/
    18  OAK-ELEV  /GREEN RIVER/OAKVILLE/ELEVATION/01MAY1992/1HOUR/OBS/
    19  OAK-ELEV  /GREEN RIVER/OAKVILLE/ELEVATION/01MAY1992/1HOUR/Z/
    20  OAK-ELEV  /GREEN RIVER/OAKVILLE/ELEVATION/01MAY1992/1HOUR/ZZ/
```

Figure 4.32 Abbreviated Catalog Text Output

3. You can print the catalog from the text dialog, from **File** menu, click **Print**.
4. From the text dialog, you can perform several file management functions: **Insert .txt**, **Save As .txt**, change the **Font**, and **Print Setup**. It also contains a **Find** menu that can facilitate a search of the catalog displayed.

To print an abbreviated catalog:

1. Open an HEC-DSS file, from the **Advanced** menu, point to **Abbreviated Disk Catalog**, click **Print**.
2. A **Print** dialog box will open (see Figure 4.31, page 4-19); click **OK** to send the catalog to the printer.

4.13.3 Full Disk Catalog

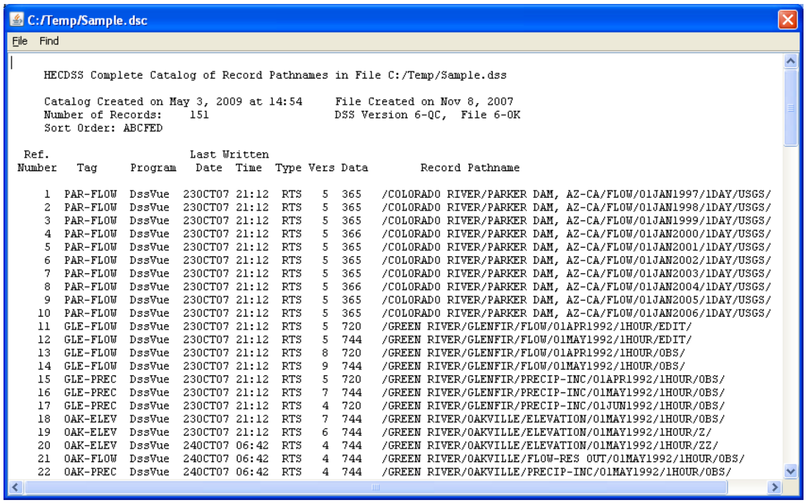
A Full Catalog is a listing of all record pathnames in the file, along with their last written date and time, their version, the number of data they contain and the type of data it contains.

To create a full catalog:

1. Open an HEC-DSS file, from the **Advanced** menu, point to **Full Disk Catalog**, click **New**.
2. A message box will appear stating the catalog has been updated and the number of records it contained (see Figure 4.29, page 4-18).

To view a full catalog:

1. Open an HEC-DSS file, from the **Advanced** menu, point to **Full Disk Catalog**, click **View**.
2. A text dialog box will open (Figure 4.33) displaying the full disk catalog.



```
    HECDSS Complete Catalog of Record Pathnames in File C:/Temp/Sample.dss

    Catalog Created on May 3, 2009 at 14:54     File Created on Nov 8, 2007
    Number of Records:    151                   DSS Version 6-QC,  File 6-OK
    Sort Order: ABCFED

 Ref.                     Last Written
Number   Tag     Program   Date  Time  Type Vers Data       Record Pathname

    1  PAR-FLOW  DssVue  23OCT07 21:12  RTS   5  365   /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN1997/1DAY/USGS/
    2  PAR-FLOW  DssVue  23OCT07 21:12  RTS   5  365   /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN1998/1DAY/USGS/
    3  PAR-FLOW  DssVue  23OCT07 21:12  RTS   5  365   /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN1999/1DAY/USGS/
    4  PAR-FLOW  DssVue  23OCT07 21:12  RTS   5  366   /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN2000/1DAY/USGS/
    5  PAR-FLOW  DssVue  23OCT07 21:12  RTS   5  365   /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN2001/1DAY/USGS/
    6  PAR-FLOW  DssVue  23OCT07 21:12  RTS   5  365   /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN2002/1DAY/USGS/
    7  PAR-FLOW  DssVue  23OCT07 21:12  RTS   5  365   /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN2003/1DAY/USGS/
    8  PAR-FLOW  DssVue  23OCT07 21:12  RTS   5  366   /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN2004/1DAY/USGS/
    9  PAR-FLOW  DssVue  23OCT07 21:12  RTS   5  365   /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN2005/1DAY/USGS/
   10  PAR-FLOW  DssVue  23OCT07 21:12  RTS   5  365   /COLORADO RIVER/PARKER DAM, AZ-CA/FLOW/01JAN2006/1DAY/USGS/
   11  GLE-FLOW  DssVue  23OCT07 21:12  RTS   5  720   /GREEN RIVER/GLENFIR/FLOW/01APR1992/1HOUR/EDIT/
   12  GLE-FLOW  DssVue  23OCT07 21:12  RTS   5  744   /GREEN RIVER/GLENFIR/FLOW/01MAY1992/1HOUR/EDIT/
   13  GLE-FLOW  DssVue  23OCT07 21:12  RTS   8  720   /GREEN RIVER/GLENFIR/FLOW/01APR1992/1HOUR/OBS/
   14  GLE-FLOW  DssVue  23OCT07 21:12  RTS   9  744   /GREEN RIVER/GLENFIR/FLOW/01MAY1992/1HOUR/OBS/
   15  GLE-PREC  DssVue  23OCT07 21:12  RTS   5  720   /GREEN RIVER/GLENFIR/PRECIP-INC/01APR1992/1HOUR/OBS/
   16  GLE-PREC  DssVue  23OCT07 21:12  RTS   7  744   /GREEN RIVER/GLENFIR/PRECIP-INC/01MAY1992/1HOUR/OBS/
   17  GLE-PREC  DssVue  23OCT07 21:12  RTS   4  720   /GREEN RIVER/GLENFIR/PRECIP-INC/01JUN1992/1HOUR/OBS/
   18  OAK-ELEV  DssVue  23OCT07 21:12  RTS   7  744   /GREEN RIVER/OAKVILLE/ELEVATION/01MAY1992/1HOUR/OBS/
   19  OAK-ELEV  DssVue  23OCT07 21:12  RTS   6  744   /GREEN RIVER/OAKVILLE/ELEVATION/01MAY1992/1HOUR/Z/
   20  OAK-ELEV  DssVue  24OCT07 06:42  RTS   4  744   /GREEN RIVER/OAKVILLE/ELEVATION/01MAY1992/1HOUR/ZZ/
   21  OAK-FLOW  DssVue  24OCT07 06:42  RTS   4  744   /GREEN RIVER/OAKVILLE/FLOW-RES OUT/01MAY1992/1HOUR/OBS/
   22  OAK-PREC  DssVue  24OCT07 06:42  RTS   4  744   /GREEN RIVER/OAKVILLE/PRECIP-INC/01MAY1992/1HOUR/OBS/
```

Figure 4.33 Full Disk Catalog Text Output

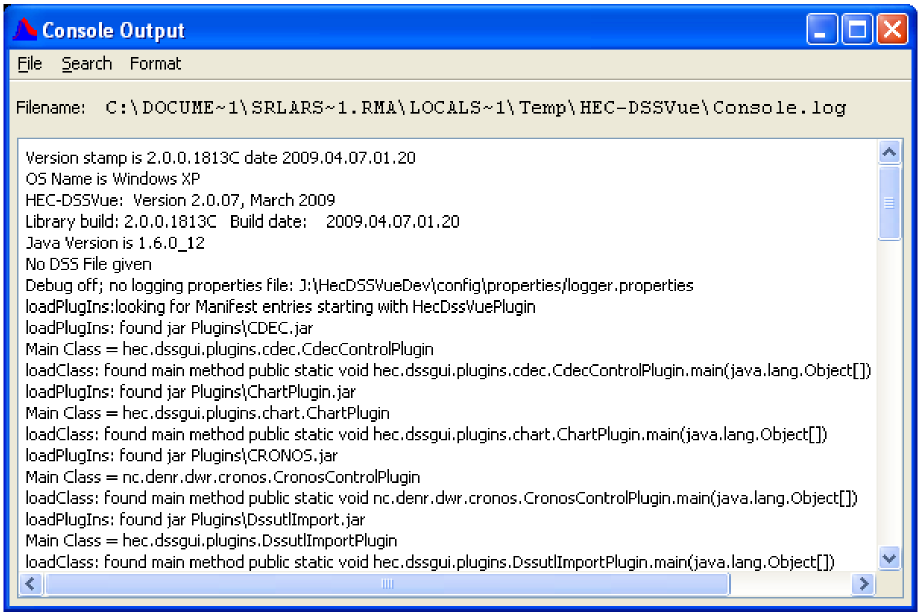
3. You can print the catalog from the text dialog, from **File** menu, click **Print**.
4. From the text dialog, you can perform several file management functions: **Insert .txt**, **Save As .txt**, change the **Font**, and **Print Setup**. It also contains a **Find** menu that can facilitate a search of the catalog displayed.

To print an abbreviated catalog:

1. Open an HEC-DSS file, from the **Advanced** menu, point to **Full Disk Catalog**, click **Print**.
2. A **Print** dialog box will open (see Figure 4.31, page 4-19); click **OK** to send the catalog to the printer.

4.14 Viewing the Console Output

As the HEC-DSSVue application is running, it generates application runtime and DSS file output. To view the output for the current instance of the HEC-DSSVue application, from the **Advanced** menu, click **Console Output**. The **Console Output** dialog box will open (Figure 4.34).



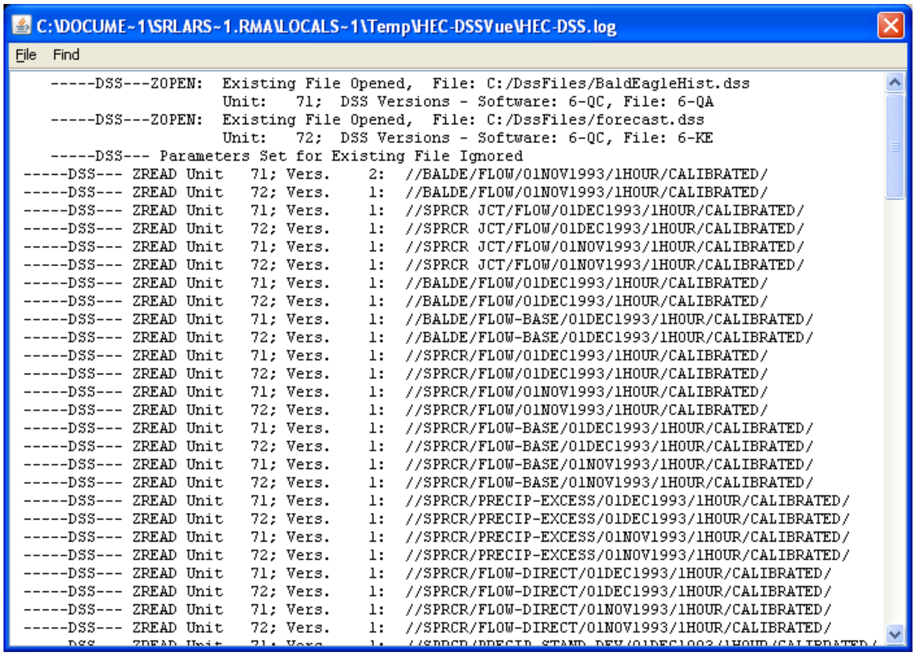
Filename: C:\DOCUME~1\SRLARS~1.RMA\LOCALS~1\Temp\HEC-DSSVue\Console.log

```
Version stamp is 2.0.0.1813C date 2009.04.07.01.20
OS Name is Windows XP
HEC-DSSVue:  Version 2.0.07, March 2009
Library build: 2.0.0.1813C   Build date:    2009.04.07.01.20
Java Version is 1.6.0_12
No DSS File given
Debug off; no logging properties file: J:\HecDSSVueDev\config\properties/logger.properties
loadPlugIns:looking for Manifest entries starting with HecDssVuePlugin
loadPlugIns: found jar Plugins\CDEC.jar
Main Class = hec.dssgui.plugins.cdec.CdecControlPlugin
loadClass: found main method public static void hec.dssgui.plugins.cdec.CdecControlPlugin.main(java.lang.Object[])
loadPlugIns: found jar Plugins\ChartPlugin.jar
Main Class = hec.dssgui.plugins.chart.ChartPlugin
loadClass: found main method public static void hec.dssgui.plugins.chart.ChartPlugin.main(java.lang.Object[])
loadPlugIns: found jar Plugins\CRONOS.jar
Main Class = nc.denr.dwr.cronos.CronosControlPlugin
loadClass: found main method public static void nc.denr.dwr.cronos.CronosControlPlugin.main(java.lang.Object[])
loadPlugIns: found jar Plugins\DssutlImport.jar
Main Class = hec.dssgui.plugins.DssutlImportPlugin
loadClass: found main method public static void hec.dssgui.plugins.DssutlImportPlugin.main(java.lang.Object[])
```

Figure 4.34 Console Output Dialog Box

4.15 Viewing HEC-DSS File Output

To view the output generated by HEC-DSS file interaction, from the **Advanced** menu, click **DSS Output**. A basic text dialog box will open (Figure 4.35) displaying application process information on the DSS file.


C:\DOCUME~1\SRLARS~1.RMA\LOCALS~1\Temp\HEC-DSSVue\HEC-DSS.log
File
Find

```
    -----DSS---ZOPEN:  Existing File Opened,  File: C:/DssFiles/BaldEagleHist.dss
                       Unit:    71;  DSS Versions - Software: 6-QC, File: 6-QA
    -----DSS---ZOPEN:  Existing File Opened,  File: C:/DssFiles/forecast.dss
                       Unit:    72;  DSS Versions - Software: 6-QC, File: 6-KE
    -----DSS--- Parameters Set for Existing File Ignored
 -----DSS--- ZREAD Unit   71; Vers.    2:  //BALDE/FLOW/01NOV1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   72; Vers.    1:  //BALDE/FLOW/01NOV1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   71; Vers.    1:  //SPRCR JCT/FLOW/01DEC1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   72; Vers.    1:  //SPRCR JCT/FLOW/01DEC1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   71; Vers.    1:  //SPRCR JCT/FLOW/01NOV1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   72; Vers.    1:  //SPRCR JCT/FLOW/01NOV1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   71; Vers.    1:  //BALDE/FLOW/01DEC1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   72; Vers.    1:  //BALDE/FLOW/01DEC1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   71; Vers.    1:  //BALDE/FLOW-BASE/01DEC1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   72; Vers.    1:  //BALDE/FLOW-BASE/01DEC1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   71; Vers.    1:  //SPRCR/FLOW/01DEC1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   72; Vers.    1:  //SPRCR/FLOW/01DEC1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   71; Vers.    1:  //SPRCR/FLOW/01NOV1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   72; Vers.    1:  //SPRCR/FLOW/01NOV1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   71; Vers.    1:  //SPRCR/FLOW-BASE/01DEC1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   72; Vers.    1:  //SPRCR/FLOW-BASE/01DEC1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   71; Vers.    1:  //SPRCR/FLOW-BASE/01NOV1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   72; Vers.    1:  //SPRCR/FLOW-BASE/01NOV1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   71; Vers.    1:  //SPRCR/PRECIP-EXCESS/01DEC1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   72; Vers.    1:  //SPRCR/PRECIP-EXCESS/01DEC1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   71; Vers.    1:  //SPRCR/PRECIP-EXCESS/01NOV1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   72; Vers.    1:  //SPRCR/PRECIP-EXCESS/01NOV1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   71; Vers.    1:  //SPRCR/FLOW-DIRECT/01DEC1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   72; Vers.    1:  //SPRCR/FLOW-DIRECT/01DEC1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   71; Vers.    1:  //SPRCR/FLOW-DIRECT/01NOV1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   72; Vers.    1:  //SPRCR/FLOW-DIRECT/01NOV1993/1HOUR/CALIBRATED/
 -----DSS--- ZREAD Unit   71; Vers.    1:  //SPRCR/PRECIP-STAND-DEV/01DEC1993/1HOUR/CALIBRATED/
```

Figure 4.35 DSS Output Dialog Box

4.15.1 Program Options

During the process of using HEC-DSSVue, properties are saved to maintain a memory of HEC-DSS file history, UI settings, and user preferences. **Program Options** allow you to re-set any such properties back to the default settings that were set when HEC-DSSVue was first installed. Other **Program Options** include required files to import/export SHEF data. For more information on the SHEF tab, see Chapter 5.

To reset all default properties for HEC-DSSVue:

1. Open an HEC-DSS file, from the **Advanced** menu, click **Program Options**, the **Options** dialog box will open (Figure 4.36).

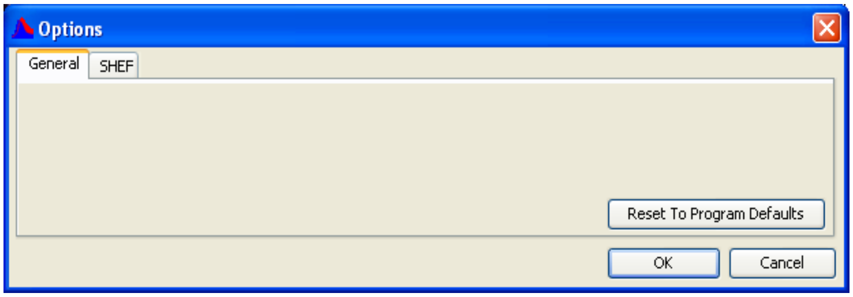
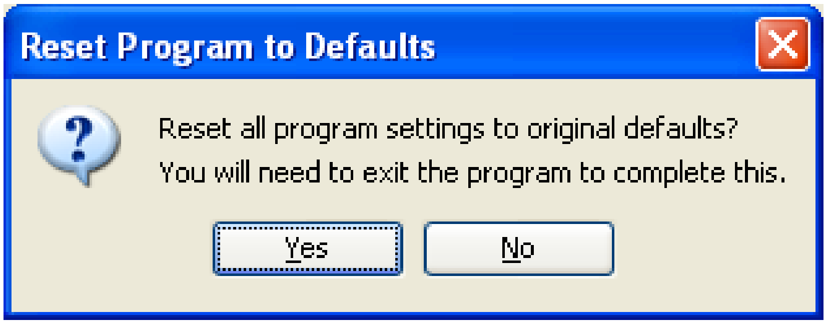


Figure 4.36 Options Dialog Box - General Tab

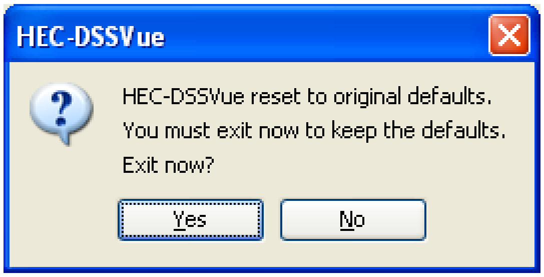
2. On the **General** tab, click the **Reset To Program Defaults** button.
3. A **Reset all Program to Default** message box will appear (Figure 4.37), verifying you would like to reset the defaults. Click **Yes**.



Reset all program settings to original defaults?
You will need to exit the program to complete this.

Figure 4.37 Reset Program to Defaults Message Box

4. Once the program defaults have been set, a message will open asking you to exit HEC-DSSVue to complete the reset process (Figure 4.38). Clicking **Yes** will close HEC-DSSVue, or you can choose to exit HEC-DSSVue later by clicking **No**. If you select **No**, some program settings may remain.



HEC-DSSVue

HEC-DSSVue reset to original defaults.
You must exit now to keep the defaults.
Exit now?

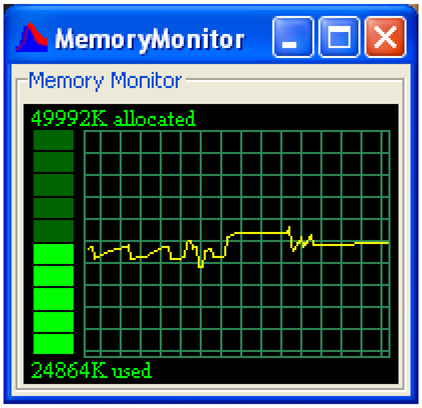
Figure 4.38 Reset Program to Default Confirmation Window

4.16 Advanced Functions

The following sections provide internal information, generally only for internal use. They involve examining and modifying internal file structure and settings, changing the trace and output levels, and setting internal flags in HEC-DSS. They are not intended for general use.

4.16.1 Memory Monitor

A set amount of memory is allocated for use during the HEC-DSSVue application runtime. The **Memory Monitor** (Figure 4.39) displays the real-time memory used by the HEC-DSSVue application in relation to the memory the program has allocated. To access it, from the **Advanced** menu, point to **Status**, click **Memory Monitor**.



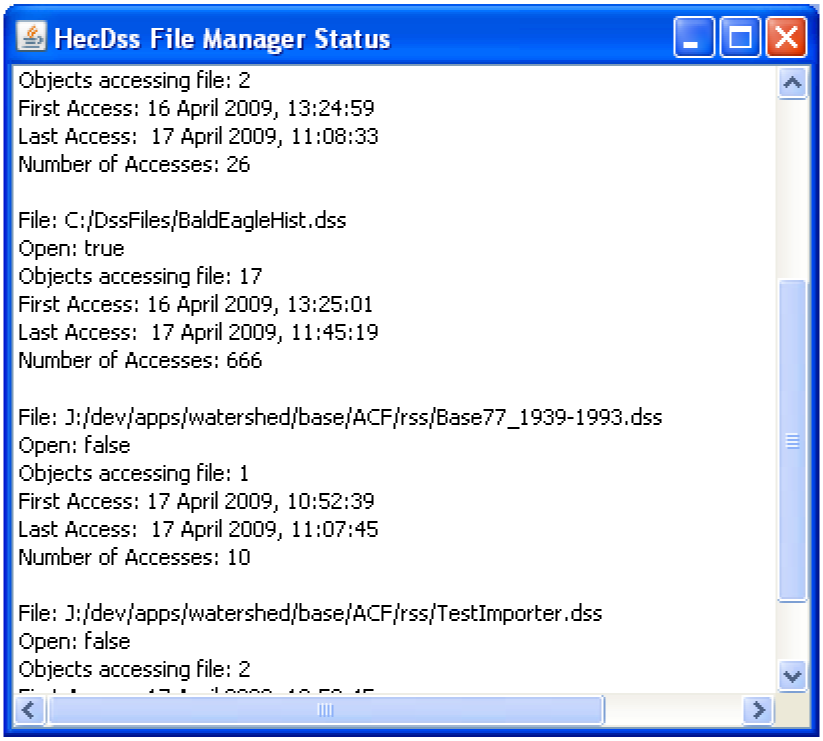
49992K allocated

24864K used

Figure 4.39 Memory Montior

4.16.2 Open List of HEC-DSS Files

A report listing the current open HEC-DSS file in HEC-DSSVue is available from the **Advanced** menu, point to **Status**, click **DSS Files Opened**. The report, Figure 4.40 (page 4-25), lists the DSS Trace date and time and how many files were accessed. It also contains access



```
Objects accessing file: 2
First Access: 16 April 2009, 13:24:59
Last Access:  17 April 2009, 11:08:33
Number of Accesses: 26

File: C:/DssFiles/BaldEagleHist.dss
Open: true
Objects accessing file: 17
First Access: 16 April 2009, 13:25:01
Last Access:  17 April 2009, 11:45:19
Number of Accesses: 666

File: J:/dev/apps/watershed/base/ACF/rss/Base77_1939-1993.dss
Open: false
Objects accessing file: 1
First Access: 17 April 2009, 10:52:39
Last Access:  17 April 2009, 11:07:45
Number of Accesses: 10

File: J:/dev/apps/watershed/base/ACF/rss/TestImporter.dss
Open: false
Objects accessing file: 2
```

Figure 4.40 HecDSS File Manager Status

information for each file, the number of records accessed, how many of these records are currently selected, the file path to the HEC-DSS file that contains the file access information, and the number of records accessed for the current session of HEC-DSSVue.

4.16.3 Viewing the DSS Header List

The DSS Header List displays all the database variables and their values used for the current HEC-DSS file.

To view the header list:

1. Open an HEC-DSS file, from the **Advanced** menu, point to **Status**, click **DSS File Header**.
2. A dialog box will open (see Figure 4.41, page 4-25) listing all the headers set for the current HEC-DSS file.

4.16.4 HEC-DSS File Internals

HEC-DSSVue offers tools to aid in viewing information on the current HEC-DSS file and debugging problems that might occur during runtime. These tools include increasing the **Message Level, DSS ZSET, DSS ZINQIR**, and **Debug/Examine File**. The **Debug** options are intended to be used internally only. For more information see the separate HEC-DSS Programmer’s Manual.

Message Levels

Message Levels control the detail of the messages written out to the HEC-DSSVue Console Output. The higher the Message Level, the more messages and detail will be written. A normal setting is four; above that provide debugging output. This setting is intended to be used by programmers only. For more information on the **Message Levels,** see the separate **HEC-DSS Programmer’s Manual**.

DSS ZSET

DSS ZSET is a tool for programmers to set file parameters. For more information on the **DSS ZSET,** see the separate **HEC-DSS Programmer's Manual**.

DSS ZINQIR

DSS ZINQIR is a tool for programmers to query file parameters. For more information on the **DSS ZINQIR,** see the separate **HEC-DSSVue Programmer's Manual**.

Debug/Examine File

The Debug/Examine File displays the internal contents of a HEC-DSS file for internal use. Modifying values in this dialog can damage files.

CHAPTER 5

Data Entry, Import and Export

HEC-DSSVue has several components to enter data into HEC-DSS files. You can enter data manually, paste data from the clipboard into a data entry from, or import data from a variety of formats. The formats include Microsoft Excel spreadsheet, the National Center for Climatic Data (NCDC), or directly from web pages, such as from the USGS NWIS database or the North Carolina CRONOS database.

Most of the import functions are provided as "plug-ins". Plug-ins are separate components that communicate with HEC-DSSVue and can be added, removed, or updated by the user. Plug-ins are added or updated by simply copying the appropriate plug-in .jar file to the *Plugins* directory. Generally, a plug-in is accessible from a menu option and is not apparent to the user that a function is being provided by a plug-in instead of HEC-DSSVue. Plug-ins can be updated without having to update the HEC-DSSVue program itself. Refer to the HEC-DSSVue plug-ins web page for updates at: http://www.hec.usace.army.mil/software/hec-dss/plug-ins.htm

HEC-DSSVue can also import and store virtually any other kind of file in HEC-DSS files, including *.jpg, .pdf, .doc,* and *.mp3* files. When you select the pathname for a generic file and "plot" or "tabulate" the file, HEC-DSSVue will launch the native application associated with the file extension. For example, if you select a *.doc* file and then press the plot or tabulate button, Microsoft Word will be launched to read the file.

You can also import files with the appropriate extension by simply "dragging and dropping" them from Windows Explorer onto HEC-DSSVue with an open HEC-DSS file. For example if you have SHEF data in a file with an extension name of "*.shef*", you can simply grab that file in Windows Explorer and drag it on top of HEC-DSSVue and it will be automatically imported.

HEC-DSSVue can also export data to a variety of formats, depending on which plug-ins are loaded. For example, if you have time-series data, you can select your pathnames. From the HEC-DSSVue main window, from the **Data Entry** menu, point to **Export**, click **SHEF**. From the **Save** browser choose the file you want to export to, and your data will be written in SHEF format to that file.

Data Entry, Import and **Export** capabilities are accessible from the **Data Entry** menu.

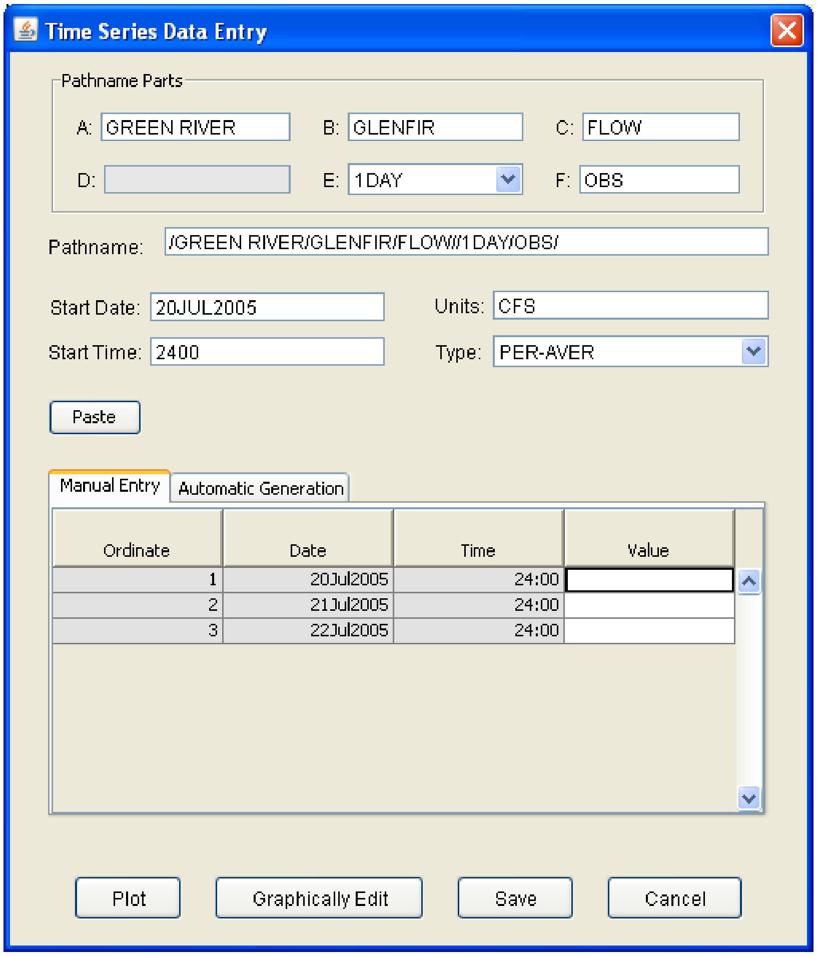
5.1 Entering Data Manually

Time series data, paired data and text data can be manually entered using the Manual Data Entry Editors available from the **Data Entry** menu. The following sections discuss all options in detail.

5.1.1 Entering Time Series Data Manually

To enter time series data manually:

1. From the **Data Entry** menu, click **Manual Time Series**. The **Time Series Data Entry** dialog box will open (Figure 5.1).



A: GREEN RIVER B: GLENFIR C: FLOW
D: E: 1DAY F: OBS

Pathname: /GREEN RIVER/GLENFIR/FLOW//1DAY/OBS/

Start Date: 20JUL2005 Units: CFS
Start Time: 2400 Type: PER-AVER

Ordinate	Date	Time	Value
1	20Jul2005	24:00	
2	21Jul2005	24:00	
3	22Jul2005	24:00	

Figure 5.1 Time Series Data Entry Dialog Box

2. Type the **Pathname Parts** into the A, B, C, and F boxes, then select the appropriate time interval for the E box. The complete pathname will automatically appear in the **Pathname** box. Alternatively, you can enter the pathname into the **Pathname** box

and the parts will appear in the **Pathname Parts** boxes. You cannot enter the "D" (date) part, as this is set according to your **Start Date**.

3. Enter the **Start Date** (e.g., *25Mar2002*) and **Start Time** (e.g., *1400*).
4. Enter the **Units** (e.g., *CFS*).
5. From the **Type** list, select a data type. Options are **PER-AVER**, **PER-CUM**, **INST-VAL**, and **INST-CUM**.
6. For regular interval time series data, the **Date**/**Time** cells in the table will fill automatically, starting with the start date and time you entered. For irregular interval data, you will need to enter a date and time for each data value.
7. Type the data values into the **Value** column.
8. To view the data in plot form, click **Plot**.
9. To graphically edit the data you have entered, click **Graphically Edit**.
10. To save the new time series record, click **Save**.

You can paste data on the clipboard from another application by clicking **Paste**. This allows you to copy data from various applications like Microsoft Excel into HEC-DSS. When you paste data into HEC-DSS, the tab characters and carriage return characters in the program that you are copying from must match the data entry table. Tab characters move to the next column to the right, while carriage return characters move to the next row down. These are the default characters used in Excel.

For regular interval time series data, the data must be in a columnar format (e.g., a column from Microsoft Excel) and should contain data values only. For irregular interval data, the date and time must precede each data value, with tab characters separating them. For paired data, all values for each ordinate must be separated by a tab character. To enter data using the paste capability:

1. Complete steps 1 through 5 above.
2. Select your data set from the other applications and **Copy** it to the clipboard.
3. Click **Paste**, you should see the data from the clipboard appear in the table.

You can also automatically generate regular-interval time series data. This will fill in a single number for a time window. To generate this data:

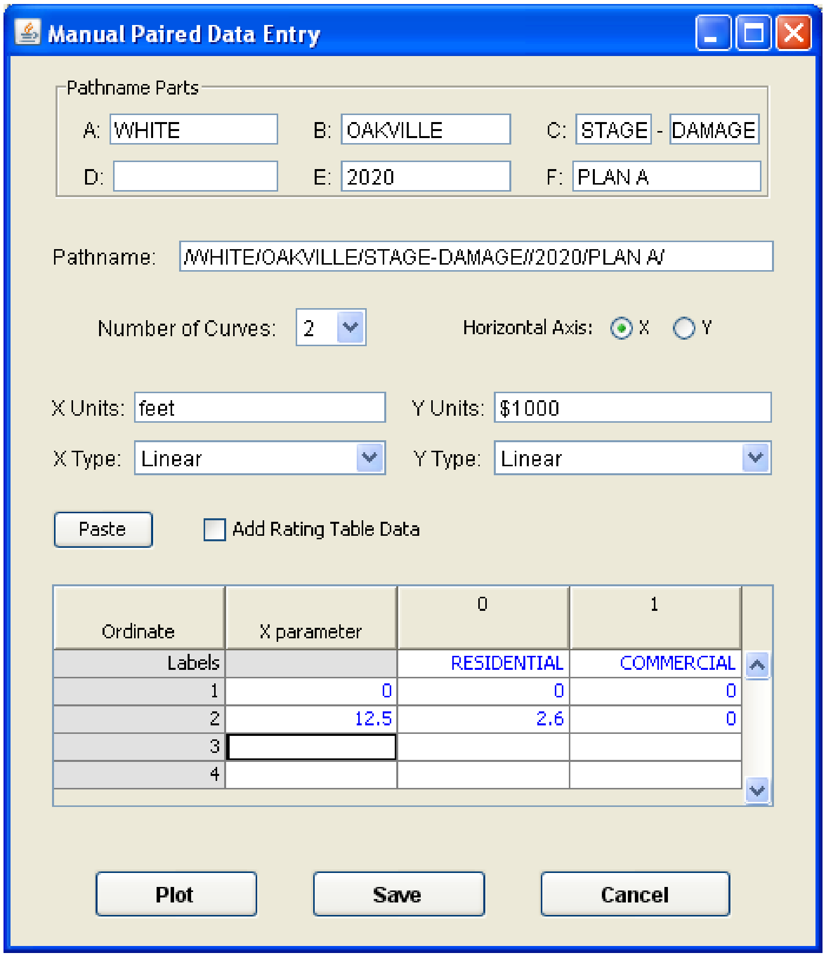
1. Complete steps 1 through 5 above.
2. Select the **Automatic Generation** tab.
3. Enter the **End Date** and **End Time** for the data.
4. Enter the **Fill Value**, which is the single value for the specified time window.

5. Click **Generate**, this will return you to the **Manual Entry** tab, where you can plot, save, or further edit the data.

5.1.2 Entering Paired Data Manually

To enter paired data manually:

1. From the **Data Entry** menu, click **Manual Paired Data**. The **Manual Paired Data Entry** dialog box dialog box (Figure 5.2) will open.



Manual Paired Data Entry

Pathname Parts

A: WHITE B: OAKVILLE C: STAGE - DAMAGE

D: E: 2020 F: PLAN A

Pathname: /WHITE/OAKVILLE/STAGE-DAMAGE//2020/PLAN A/

Number of Curves: 2 Horizontal Axis: ● X ○ Y

X Units: feet Y Units: $1000

X Type: Linear Y Type: Linear

□ Add Rating Table Data

Ordinate	X parameter	0	1
Labels		RESIDENTIAL	COMMERCIAL
1	0	0	0
2	12.5	2.6	0
3			
4			

Figure 5.2 Manual Paired Data Entry Dialog Box

2. Type the **Pathname Parts** into the A, B, C, D, E, and F boxes. The complete pathname will automatically appear in the **Pathname** box. You can also enter the pathname into the **Pathname** box; the parts will appear in the **Pathname Parts** boxes. Be sure the "C" part contains both an X parameter and a Y parameter, separated by a hyphen (e.g., *STAGE-FLOW* or *ELEV-DAMAGE*).

3. From the **Number of Curves** list select the number of curves to be entered. Also, from **Horizontal Axis**, select **X** or **Y**.
4. In the **X Units** box, enter the units for *X* (belonging to the first parameter) and in the **Y Units** box, enter the units for *Y* (belonging to the second parameter).
5. From the **X Type** and **Y Type** lists, select the type for the respective parameters. Available options are **Linear**, **Log**, and **Probability**.
6. In the table, the *Y* parameter columns will split into individual columns according to the number of curves you specified.
7. Enter the data values into the **X parameter** and **Y parameter** columns.
8. To view the data in plot form, click **Plot**.
9. To save the new time series record, click **Save**.

You can paste data on the clipboard from another application by clicking **Paste**. This allows you to copy data from various applications like Microsoft Excel into HEC-DSS. When you paste data into HEC-DSS, the tab characters and carriage return characters in the program that you are copying from must match the data entry table. Tab characters move to the next column to the right, while carriage return characters move to the next row down. These are the default characters used in Excel. For paired data, all values for each ordinate must be separated by a tab character with a carriage return character at the end of each row. To enter data using the paste capability:

1. Complete steps 1 through 6 above.
2. Select your data set from the other applications and **Copy** it to the clipboard.
3. Click **Paste**, you should see the data from the clipboard appear in the table.

5.1.3 Manual Text Entry

HEC-DSS has a text convention format. You can enter data by manual text data entry and type or paste text into the text editor, or you can import a text file. You cannot import text files into the HEC-DSS text format by dragging and dropping a .txt file; this will save a .txt file in the generic file format.

To enter text data manually:

1. From the **Data Entry** menu, click **Manual Text**. A dialog box will open (see Figure 5.3, page 5-6).
2. Type or paste the text data into the text dialog box. You can change the font, but the font will not be stored with the text.

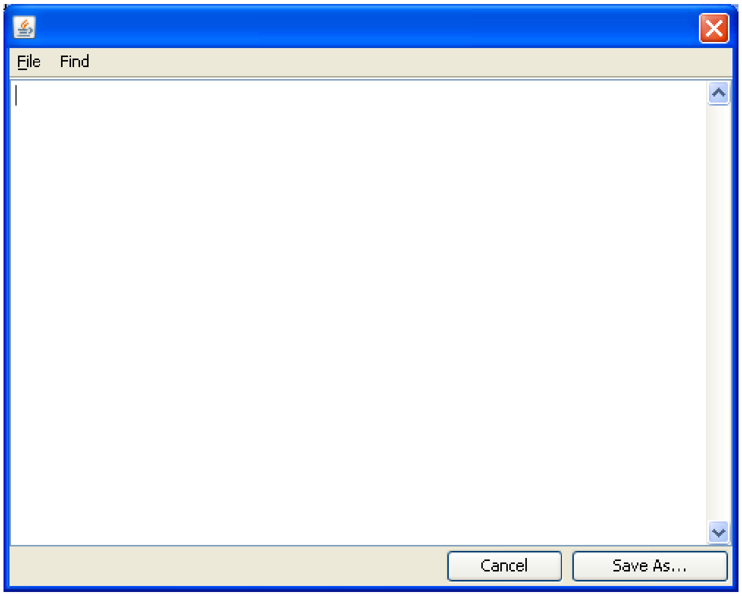


Figure 5.3 Dialog Box used to Enter Text

3. From the **File** menu you can insert a text file (*.txt*). You can also save your text as a *.txt* file, save as an HEC-DSS file, or print your file (Figure 5.4). You can also change the font of the text, but that will not be retained. (**Note:** You can also import text data directly from a *.txt* file from the HEC-DSSVue main window. From the **Date Entry** menu, point to **Import**, click **Text from File**.)
4. After the text has been entered, click **Save As** to save the data to a DSS file. Also, from the **File** menu you can click **Save** or **Save As** to save the data as a DSS file. Either method opens the **Save As** dialog box (Figure 5.5).

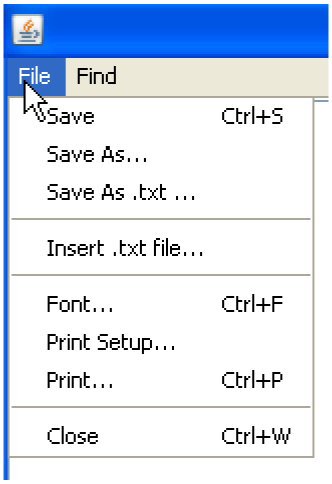
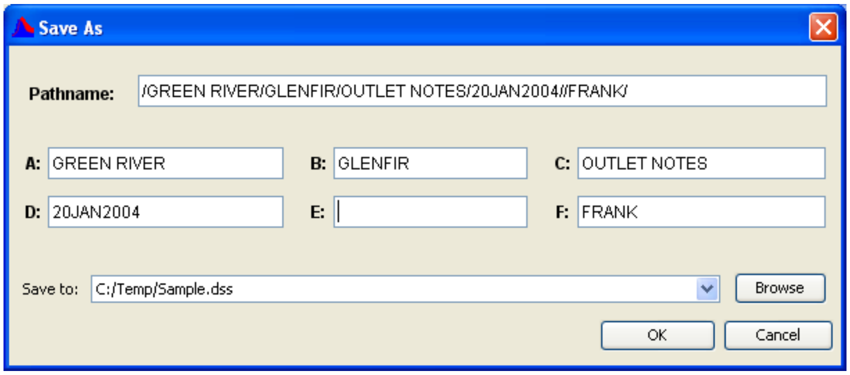


Figure 5.4 File Menu


Save As

Pathname: /GREEN RIVER/GLENFIR/OUTLET NOTES/20JAN2004//FRANK/

A: GREEN RIVER B: GLENFIR C: OUTLET NOTES

D: 20JAN2004 E: F: FRANK

Save to: C:/Temp/Sample.dss

Figure 5.5 Save As Dialog Box

5. Enter the **Pathname Parts** into the A, B, C, D, E, and F boxes. The complete pathname will automatically appear in the **Pathname** box. You can also enter the pathname into the **Pathname** box; the parts will appear in the **Pathname Parts** boxes (see Figure 5.5, page 5-6).
6. To save the record, click **OK**.

5.2 Import/Export Files

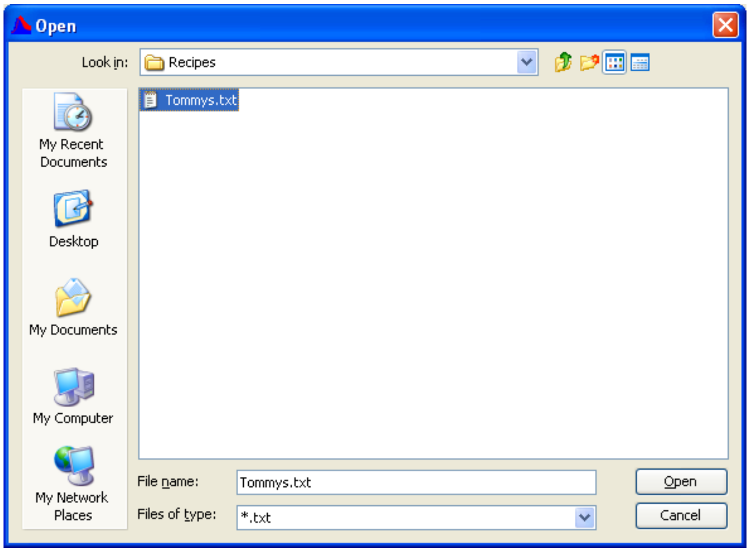
You can import and export text files, image files and other types of files into HEC-DSS files from the **Import** or **Export** menu items on the **Data Entry** menu.

5.2.1 Import Text Files

HEC-DSS has a text convention format. You can enter data by manual text data entry and type or paste text into the text editor, or you can import a text file. You cannot import text files into the HEC-DSS text format by dragging and dropping a .txt file; this will save a .txt file in the generic file format.

To import a text file:

1. From the **Data Entry** menu, point to **Import**, click **Text from File.**
2. An **Open** browser will be open (Figure 5.6). Navigate to the directory and file you wish to import. Select the file and click **Open**, you can only import non-formatted regular text.



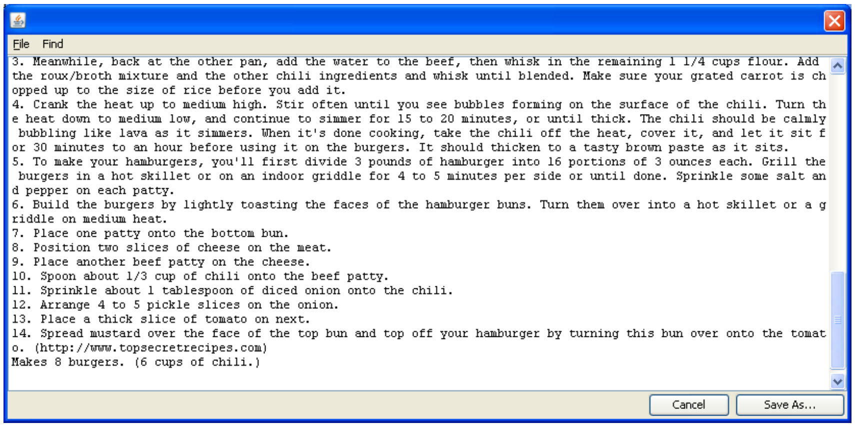
Look in: Recipes

File name: Tommys.txt

Files of type: *.txt

Figure 5.6 Open Browser

3. A dialog box will open (Figure 5.7) displaying the contents for the selected file. You can make any changes you want to the text.



```
3. Meanwhile, back at the other pan, add the water to the beef, then whisk in the remaining 1 1/4 cups flour. Add
the roux/broth mixture and the other chili ingredients and whisk until blended. Make sure your grated carrot is ch
opped up to the size of rice before you add it.
4. Crank the heat up to medium high. Stir often until you see bubbles forming on the surface of the chili. Turn th
e heat down to medium low, and continue to simmer for 15 to 20 minutes, or until thick. The chili should be calmly
 bubbling like lava as it simmers. When it's done cooking, take the chili off the heat, cover it, and let it sit f
or 30 minutes to an hour before using it on the burgers. It should thicken to a tasty brown paste as it sits.
5. To make your hamburgers, you'll first divide 3 pounds of hamburger into 16 portions of 3 ounces each. Grill the
 burgers in a hot skillet or on an indoor griddle for 4 to 5 minutes per side or until done. Sprinkle some salt an
d pepper on each patty.
6. Build the burgers by lightly toasting the faces of the hamburger buns. Turn them over into a hot skillet or a g
riddle on medium heat.
7. Place one patty onto the bottom bun.
8. Position two slices of cheese on the meat.
9. Place another beef patty on the cheese.
10. Spoon about 1/3 cup of chili onto the beef patty.
11. Sprinkle about 1 tablespoon of diced onion onto the chili.
12. Arrange 4 to 5 pickle slices on the onion.
13. Place a thick slice of tomato on next.
14. Spread mustard over the face of the top bun and top off your hamburger by turning this bun over onto the tomat
o. (http://www.topsecretrecipes.com)
Makes 8 burgers. (6 cups of chili.)
```

Figure 5.7 Dialog Box for Importing Text

4. After the text has been entered, click **Save As** to save the data to a DSS file. Also, from the **File** menu you can click **Save** or **Save As** to save the data as a DSS file. Either method opens the **Save As** dialog box (see Figure 5.5, page 5-6).
5. Enter the **Pathname Parts** into the A, B, C, D, E, and F boxes. The complete pathname will automatically appear in the **Pathname** box. You can also enter the pathname into the **Pathname** box; the parts will appear in the **Pathname Parts** boxes (see Figure 5.5, page 5-6).
6. To save the record, click **OK**.

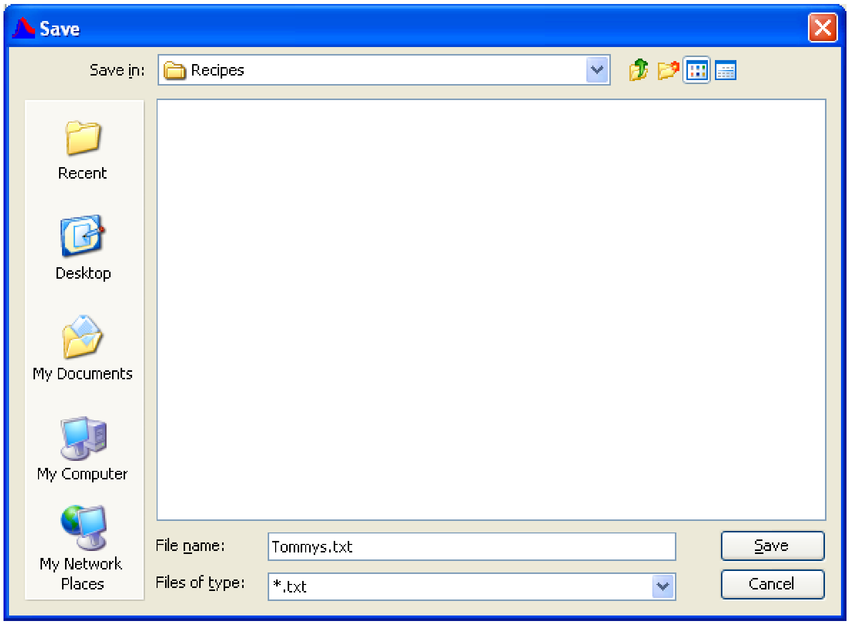
5.2.2 Exporting Text Files

To export to a text file:

1. You can export data to a text file by selecting the pathname for the data set that you want to export, from the **Data Entry** menu, point to **Export,** click **Export to File(s).** You can only export one text data set at a time for this convention. (For image and generic files you can export several at one time.) This menu option will only appear for text, image and generic files.
2. A **Save** browser will open (see Figure 5.8, page 5-9). Navigate to the directory where you want the file saved, in the **File name** box enter the name of the file, and click **Save**.

5.2.3 Import Image and Generic Files

HEC-DSS has a convention for storing image (photos, etc.) and other files in a generic file format. For this format, the pathname will have a C part



Save in: Recipes

File name: Tommys.txt

Files of type: *.txt

Figure 5.8 Save Browser

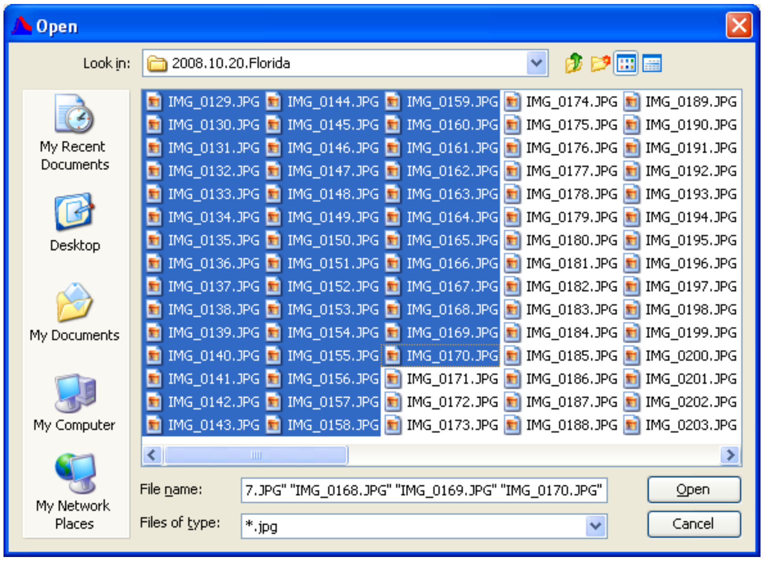
that is the name of the file (without directory), a D part that is either "*IMAGE*" or "*FILE*", and an E part that is the extension of the file (e.g., "*JPG*" or "*PDF*"). For more information, refer to Chapter 1, Section 1.2.3.

When you select images or files (generic type) in HEC-DSSVue and select either “plot" or "tabulate", HEC-DSSVue will launch a photo viewer for image data, or launch the application associated for that file type. For example, if the file type is "*.pdf*", HEC-DSSVue will launch Adobe Acrobat with a copy of that file loaded into it.

The easiest way to import both image and generic files is to drag them from Windows Explorer and drop them into the main screen of HEC-DSSVue with an opened HEC-DSS file. A **Save As** dialog box will open (see Figure 5.5, page 5-6) where you can enter the pathname parts. You can also import images and generic files from the **Data Entry** menu.

To import image or generic files into HEC-DSS:

1. From the **Data Entry** menu, point to **Import,** either click **Images** or **Files (Generic type).**
2. An **Open** browser will open (see Figure 5.9, page 5-10). Navigate to the directory and select the files you wish to import. Select the files and click **Open**. You can import multiple files at one time.
3. The **Save As** dialog box will open (see Figure 5.5, page 5-6). Enter the **Pathname Parts** into the A, B, and F boxes. The complete pathname will automatically appear in the **Pathname** box. You can also enter the pathname into the **Pathname** box; the



Look in: 2008.10.20.Florida

IMG_0129.JPG	IMG_0144.JPG	IMG_0159.JPG	IMG_0174.JPG	IMG_0189.JPG
IMG_0130.JPG	IMG_0145.JPG	IMG_0160.JPG	IMG_0175.JPG	IMG_0190.JPG
IMG_0131.JPG	IMG_0146.JPG	IMG_0161.JPG	IMG_0176.JPG	IMG_0191.JPG
IMG_0132.JPG	IMG_0147.JPG	IMG_0162.JPG	IMG_0177.JPG	IMG_0192.JPG
IMG_0133.JPG	IMG_0148.JPG	IMG_0163.JPG	IMG_0178.JPG	IMG_0193.JPG
IMG_0134.JPG	IMG_0149.JPG	IMG_0164.JPG	IMG_0179.JPG	IMG_0194.JPG
IMG_0135.JPG	IMG_0150.JPG	IMG_0165.JPG	IMG_0180.JPG	IMG_0195.JPG
IMG_0136.JPG	IMG_0151.JPG	IMG_0166.JPG	IMG_0181.JPG	IMG_0196.JPG
IMG_0137.JPG	IMG_0152.JPG	IMG_0167.JPG	IMG_0182.JPG	IMG_0197.JPG
IMG_0138.JPG	IMG_0153.JPG	IMG_0168.JPG	IMG_0183.JPG	IMG_0198.JPG
IMG_0139.JPG	IMG_0154.JPG	IMG_0169.JPG	IMG_0184.JPG	IMG_0199.JPG
IMG_0140.JPG	IMG_0155.JPG	IMG_0170.JPG	IMG_0185.JPG	IMG_0200.JPG
IMG_0141.JPG	IMG_0156.JPG	IMG_0171.JPG	IMG_0186.JPG	IMG_0201.JPG
IMG_0142.JPG	IMG_0157.JPG	IMG_0172.JPG	IMG_0187.JPG	IMG_0202.JPG
IMG_0143.JPG	IMG_0158.JPG	IMG_0173.JPG	IMG_0188.JPG	IMG_0203.JPG

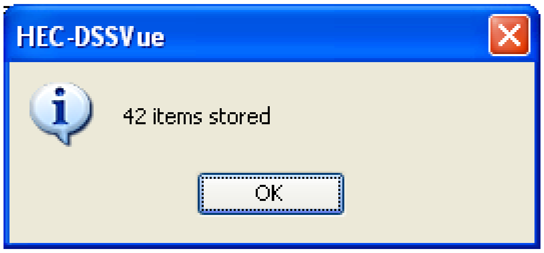
File name: 7.JPG" "IMG_0168.JPG" "IMG_0169.JPG" "IMG_0170.JPG"

Files of type: *.jpg

Figure 5.9 Open Browser – Importing Images and Generic Files

parts will appear in the **Pathname Parts** boxes (see Figure 5.5, page 5-6).

4. To save the records, click **OK**. A confirmation window will appear, see Figure 5.10.

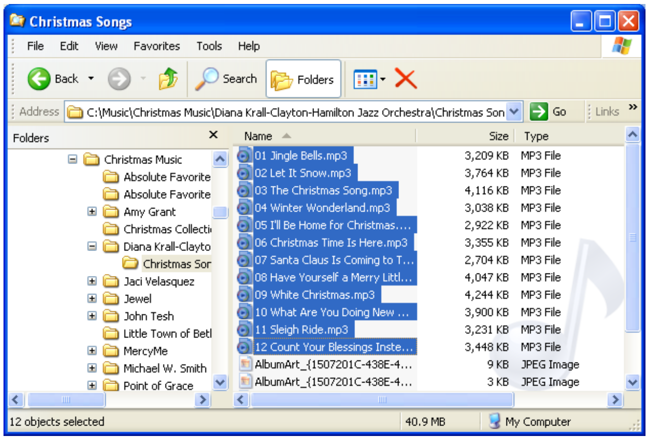


42 items stored

Figure 5.10 Save Files Confirmed

Or, alternatively you can import image and generic files by:

5. From Windows Explorer (Figure 5.11), select the files that you wish to import.



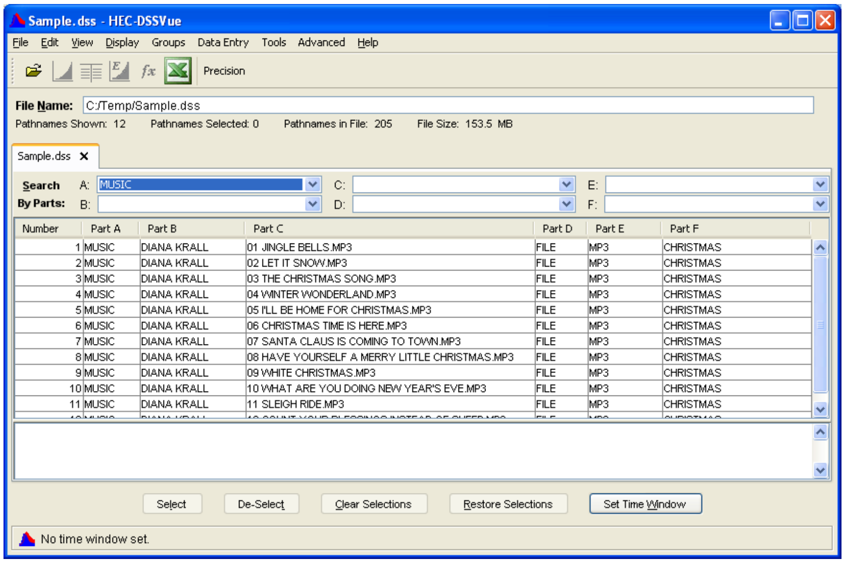
Address C:\Music\Christmas Music\Diana Krall-Clayton-Hamilton Jazz Orchestra\Christmas Son

Name	Size	Type
01 Jingle Bells.mp3	3,209 KB	MP3 File
02 Let It Snow.mp3	3,764 KB	MP3 File
03 The Christmas Song.mp3	4,116 KB	MP3 File
04 Winter Wonderland.mp3	3,038 KB	MP3 File
05 I'll Be Home for Christmas....	2,922 KB	MP3 File
06 Christmas Time Is Here.mp3	3,355 KB	MP3 File
07 Santa Claus Is Coming to T...	2,704 KB	MP3 File
08 Have Yourself a Merry Littl...	4,047 KB	MP3 File
09 White Christmas.mp3	4,244 KB	MP3 File
10 What Are You Doing New ...	3,900 KB	MP3 File
11 Sleigh Ride.mp3	3,231 KB	MP3 File
12 Count Your Blessings Inste...	3,448 KB	MP3 File
AlbumArt_{1507201C-438E-4...	9 KB	JPEG Image
AlbumArt_{1507201C-438E-4...	3 KB	JPEG Image

12 objects selected 40.9 MB

Figure 5.11 Example of Data Selection for Drag and Drop

6. "Drag" the files onto the main HEC-DSSVue screen (with a HEC-DSS file open).
7. The **Save As** dialog box will open (see Figure 5.5, page 5-6). Enter the **Pathname Parts** into the A, B, and F boxes. The complete pathname will automatically appear in the **Pathname** box. You can also enter the pathname into the **Pathname** box; the parts will appear in the **Pathname Parts** boxes.
8. To save the records, click **OK**, a message window will open (see Figure 5.10, page 5-10).
9. The actual file names will be displayed in the C part, as shown in Figure 5.12.



File Name: C:/Temp/Sample.dss

Pathnames Shown: 12 Pathnames Selected: 0 Pathnames in File: 205 File Size: 153.5 MB

Search By Parts: A: MUSIC B: C: D: E: F:

Number	Part A	Part B	Part C	Part D	Part E	Part F
1	MUSIC	DIANA KRALL	01 JINGLE BELLS.MP3	FILE	MP3	CHRISTMAS
2	MUSIC	DIANA KRALL	02 LET IT SNOW.MP3	FILE	MP3	CHRISTMAS
3	MUSIC	DIANA KRALL	03 THE CHRISTMAS SONG.MP3	FILE	MP3	CHRISTMAS
4	MUSIC	DIANA KRALL	04 WINTER WONDERLAND.MP3	FILE	MP3	CHRISTMAS
5	MUSIC	DIANA KRALL	05 I'LL BE HOME FOR CHRISTMAS.MP3	FILE	MP3	CHRISTMAS
6	MUSIC	DIANA KRALL	06 CHRISTMAS TIME IS HERE.MP3	FILE	MP3	CHRISTMAS
7	MUSIC	DIANA KRALL	07 SANTA CLAUS IS COMING TO TOWN.MP3	FILE	MP3	CHRISTMAS
8	MUSIC	DIANA KRALL	08 HAVE YOURSELF A MERRY LITTLE CHRISTMAS.MP3	FILE	MP3	CHRISTMAS
9	MUSIC	DIANA KRALL	09 WHITE CHRISTMAS.MP3	FILE	MP3	CHRISTMAS
10	MUSIC	DIANA KRALL	10 WHAT ARE YOU DOING NEW YEAR'S EVE.MP3	FILE	MP3	CHRISTMAS
11	MUSIC	DIANA KRALL	11 SLEIGH RIDE.MP3	FILE	MP3	CHRISTMAS

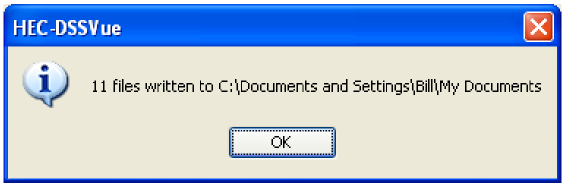
No time window set.

Figure 5.12 Imported Filenames Located in the C Part

5.2.4 Exporting Image and Generic Files

You can export image and generic files by:

1. Select the pathnames for the data sets that you want to export, from the **Data Entry** menu, point to **Export,** click **Export to File(s).**
2. A **Save** browser will open (see Figure 5.8, page 5-9); navigate to the directory you want to save the file, click **Save**. The images or files will be saved with their original names. A confirmation window will open (Figure 5.13).



11 files written to C:\Documents and Settings\Bill\My Documents

Figure 5.13 Saving Files Confirmation

5.3 Importing and Exporting SHEF Data

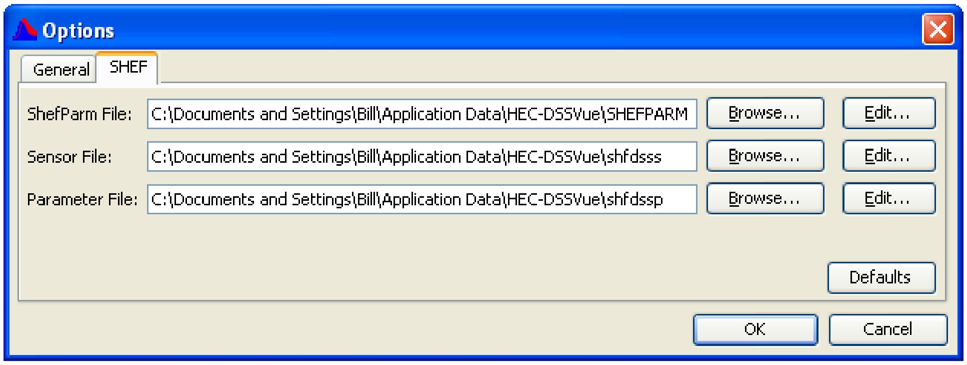
The Standard Hydrologic Exchange Format (SHEF) is a text based format developed by the National Weather Service (NWS) for exchange of real time operational data with other agencies. SHEF was adopted by mutual agreement for exchange of data between NWS and the Corps of Engineers.

The import/export SHEF features are complied into HEC-DSSVue; they are not plug-ins and cannot be updated or removed from the program.

5.3.1 SHEF Table Files

The implementation of the import and export of SHEF data in HEC-DSSVue is based on the prior existing programs SHFDSS and DSSSHF. These programs, and their implementation in HEC-DSSVue, use three reference tables: one for defining SHEF parameter codes (SHEFPARAM), one for relating SHEF physical element (PE) codes to DSS parameters (*shfdssp*), and one for relating station identifiers to observation frequencies and pathname parts (*shfdsss*). These files are required to access the import/export capability; sample files are distributed with HEC-DSSVue.

The SHEF sample table files are distributed with HEC-DSSVue and stored in the users application directory under HEC-DSSVue (the location of the application directory is dependent on the operating system). You can browse to these files and edit the files, from the **Advanced** menu, click **Program Options,** the **Options** dialog box will open (Figure 5.14). Click the **SHEF** tab, information from the dialog box will show the location of the SHEF sample table files. If you have existing files from SHFDSS, you can use those in HEC-DSSVue.



ShefParm File: C:\Documents and Settings\Bill\Application Data\HEC-DSSVue\SHEFPARM

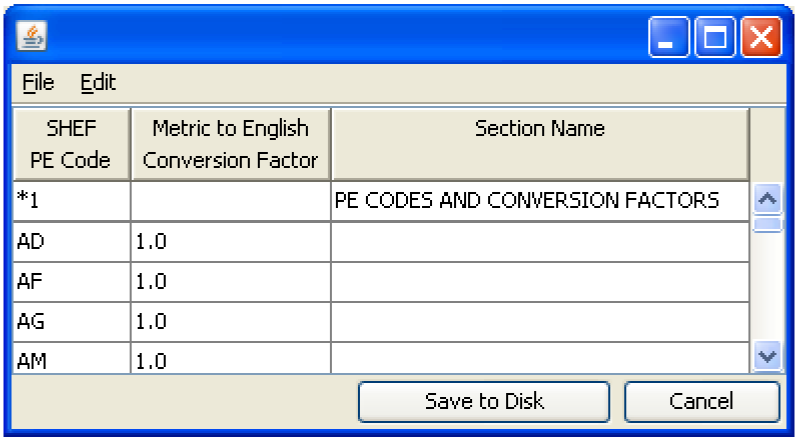
Sensor File: C:\Documents and Settings\Bill\Application Data\HEC-DSSVue\shfdsss

Parameter File: C:\Documents and Settings\Bill\Application Data\HEC-DSSVue\shfdssp

Figure 5.14 Options Dialog Box – SHEF Tab

SHEFPARM

The SHEFPARM file contains SHEF PE (Physical Element) codes and the metric - English conversion factor. Generally, the supplied file should be sufficient unless newer PE codes are being used. To edit the SHEFPARM file, from the **Options** dialog box, with the SHEF tab clicked (see Figure 5.14, page 5-12), click **Edit** that is next to the **ShefParm File** box. A dialog box will open (Figure 5.15), which will allow the user to edit the SHEFPARM file.   In this table, the first column contains the PE code and the second column contains the Metric to English conversion factor, if there is no factor, set this cell to 1.0.

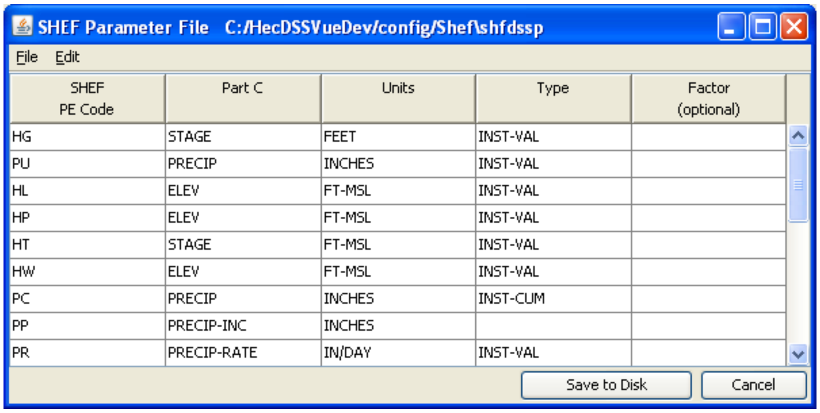


SHEF PE Code	Metric to English Conversion Factor	Section Name
*1		PE CODES AND CONVERSION FACTORS
AD	1.0	
AF	1.0	
AG	1.0	
AM	1.0	

Figure 5.15 SHEFPARM File Dialog Box

Parameter File – shfdssp

The parameter file links the HEC-DSS "C" part (the parameter), the data units, and type to the SHEF parameter (PE). To edit the parameter file, from the **Options** dialog box, with the SHEF tab clicked (see Figure 5.14, page 5-12), click **Edit** that is next to the **Parameter File** box. The **SHEF Parameter File** dialog box will open (Figure 5.16), which will allow the user to edit the parameter file. From the **SHEF PE Code** column of the



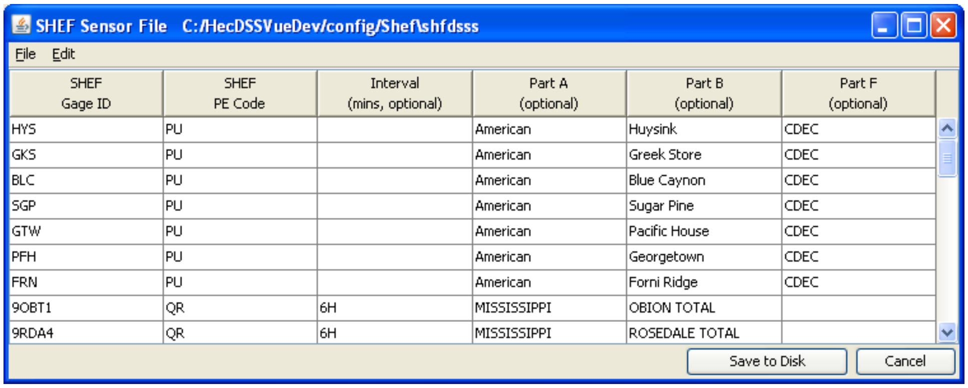
SHEF PE Code	Part C	Units	Type	Factor (optional)
HG	STAGE	FEET	INST-VAL	
PU	PRECIP	INCHES	INST-VAL	
HL	ELEV	FT-MSL	INST-VAL	
HP	ELEV	FT-MSL	INST-VAL	
HT	STAGE	FT-MSL	INST-VAL	
HW	ELEV	FT-MSL	INST-VAL	
PC	PRECIP	INCHES	INST-CUM	
PP	PRECIP-INC	INCHES		
PR	PRECIP-RATE	IN/DAY	INST-VAL	

Figure 5.16 SHEF Parameter File Dialog Box

table edit the PE code; from the **Part C** column (see Figure 5.15, page 5-13) edit the C Part of the DSS pathname, the units for the PE code can be edited in the **Units** column, the type of for the PE code can be edited in the **Type** column, and from the **Factor** column, an optional factor to multiply the data by (e.g., kilo-cfs might have a factor of 1000.0 and store the data in HEC-DSS as cfs) can be entered. The sample parameter file that is distributed with HEC-DSSVue should be relatively complete.

Sensor File – shfdsss

The SHEF sensor file links the SHEF gage ID to the HEC-DSS A, B, and F parts. To edit the parameter file, from the **Options** dialog box, with the SHEF tab clicked (see Figure 5.14, page 5-12), click **Edit** that is next to the **Sensor File** box. The **SHEF Sensor File** dialog box will open (Figure 5.17), which will allow the user to edit the sensor file. For each Gage ID and PE code, a row should exist with the Gage ID in column 1, the PE code in column 2, an optional time interval in column 3, the A part (basin) in column 4, the B part (location) in column 5, and the F part (version) in column 6. If no time interval is given, the data interval is assumed to be irregular. If a time interval is provided, is should end with "M" for minutes (e.g. 15M), or "H" for hours (e.g., 6H). If an entry does not exist for the data type being imported, default values are used.



SHEF Gage ID	SHEF PE Code	Interval (mins, optional)	Part A (optional)	Part B (optional)	Part F (optional)
HYS	PU		American	Huysink	CDEC
GKS	PU		American	Greek Store	CDEC
BLC	PU		American	Blue Caynon	CDEC
SGP	PU		American	Sugar Pine	CDEC
GTW	PU		American	Pacific House	CDEC
PFH	PU		American	Georgetown	CDEC
FRN	PU		American	Forni Ridge	CDEC
9OBT1	QR	6H	MISSISSIPPI	OBION TOTAL	
9RDA4	QR	6H	MISSISSIPPI	ROSEDALE TOTAL	

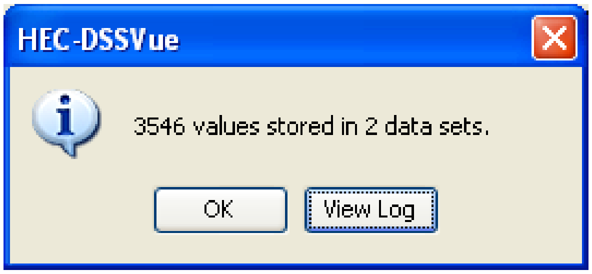
Figure 5.17 SHEF Sensor File Dialog Box

5.3.2 Importing SHEF Data

Once the SHEF tables are setup, you can import SHEF data by either dragging SHEF files with an extension of ".*shef*" or “".*shf*" from Windows Explorer and dropping them into the main screen of HEC-DSSVue with an open HEC-DSS file or from the **Data Entry** menu.

To import SHEF data files into HEC-DSS:

1. From the **Data Entry** menu, point to **Import,** click **SHEF.**
2. An **Open** browser will open (see Figure 5.6, page 5-7), navigate to a directory and select the files you wish to import. Select the files to import, click **Open**. You can import multiple files at one time. The file extensions do not matter for this option (the files do not need to end with "*.shef*").
3. A message window will open (Figure 5.18); this window will display how many values were stored from the SHEF data sets. You can view the SHEF process log by clicking **View Log**.

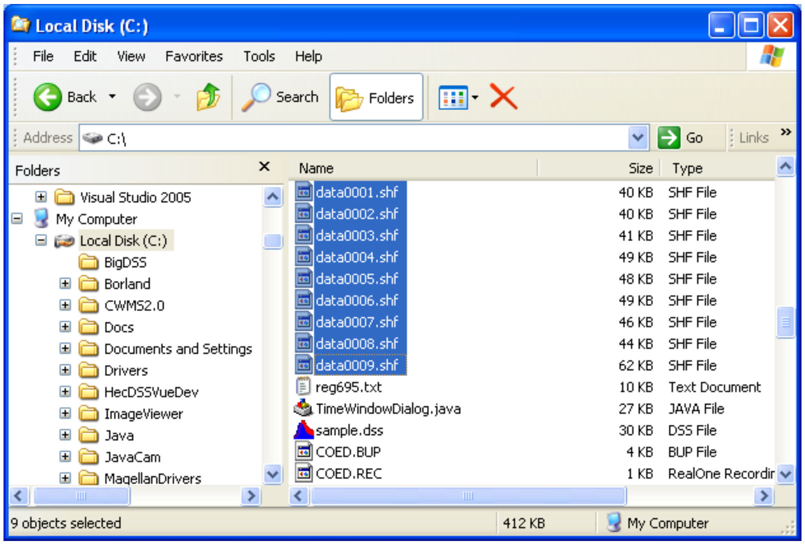


3546 values stored in 2 data sets.

Figure 5.18 SHEF Import Status Message Window

Or, alternatively you can import SHEF files if they have the extension "*.shef*" or "*.shf*" by:

1. From Windows Explorer, select the "*.shf*" or "*.shef*" files that you wish to import (Figure 5.19).



Name	Size	Type
data0001.shf	40 KB	SHF File
data0002.shf	40 KB	SHF File
data0003.shf	41 KB	SHF File
data0004.shf	49 KB	SHF File
data0005.shf	48 KB	SHF File
data0006.shf	49 KB	SHF File
data0007.shf	46 KB	SHF File
data0008.shf	44 KB	SHF File
data0009.shf	62 KB	SHF File
reg695.txt	10 KB	Text Document
TimeWindowDialog.java	27 KB	JAVA File
sample.dss	30 KB	DSS File
COED.BUP	4 KB	BUP File
COED.REC	1 KB	RealOne Recordir

9 objects selected | 412 KB | My Computer

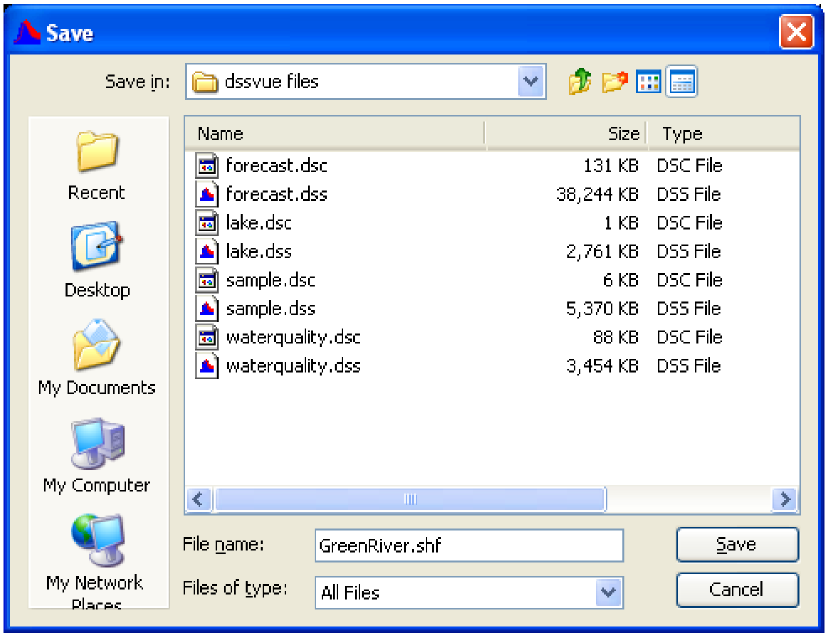
Figure 5.19 Selection of .shf or .shef Files to Drag and Drop into DSSVue

2. "Drag" the files onto the main HEC-DSSVue screen (with your HEC-DSS file that you want to import to opened.)
3. The data is imported and a message window will open (Figure 5.18), this window will display how many values were stored from the SHEF data sets. You can view the SHEF process log by clicking **View Log**.

5.3.3 Exporting SHEF

You can export SHEF data by:

1. Select the pathnames for the data sets that you want to export, then from the **Data Entry** menu, point to **Export,** click **SHEF.**
2. A **Save** browser will be open (Figure 5.20), navigate to the directory where you want the exported SHEF data to be, type the name of file in the **File name** box, click **Save**. All data will be saved in one file.



Save in: dssvue files

Name	Size	Type
forecast.dsc	131 KB	DSC File
forecast.dss	38,244 KB	DSS File
lake.dsc	1 KB	DSC File
lake.dss	2,761 KB	DSS File
sample.dsc	6 KB	DSC File
sample.dss	5,370 KB	DSS File
waterquality.dsc	88 KB	DSC File
waterquality.dss	3,454 KB	DSS File

File name: GreenRiver.shf

Files of type: All Files

Figure 5.20 Save Browser - SHEF Data

5.4 Importing and Exporting Microsoft Excel and "*.csv*" Data

Importing, exporting and editing data with Microsoft Excel is accomplished in HEC-DSSVue by the "plug-in *Excel.jar*", stored in the HEC-DSSVue **Plugins** directory. Plug-ins can be updated without having to update the HEC-DSSVue program itself. Refer to the HEC-DSSVue plug-ins web page for updates at: http://www.hec.usace.army.mil/software/hec-dss/plug-ins.htm.

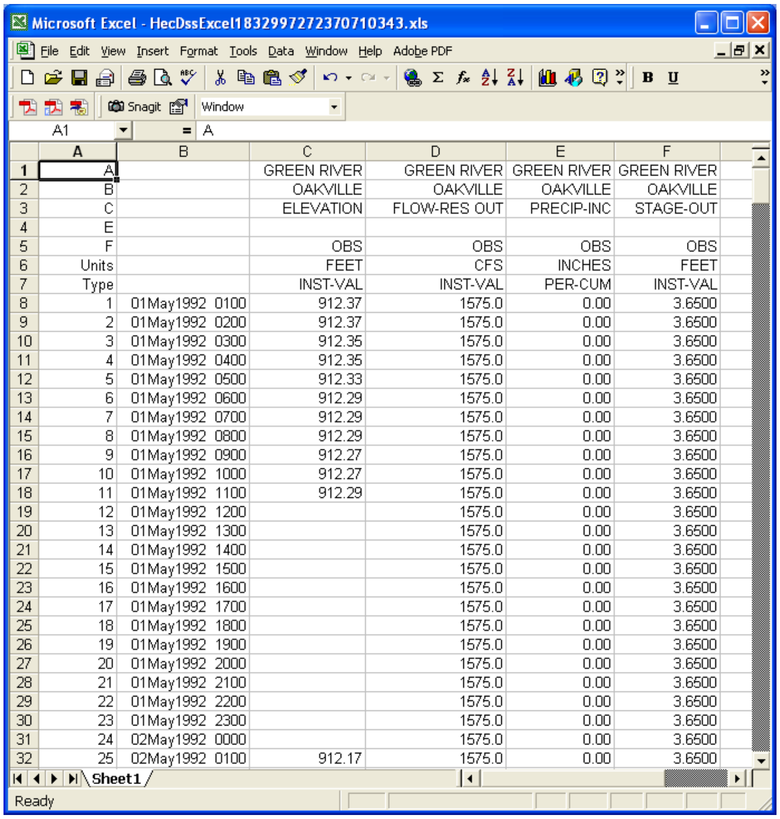
At this time, only time-series data can be imported into HEC-DSS. Check the web site above for updates. Both time series and paired data can be exported to Excel but only time-series data can be edited in Excel.

You can also import time-series data in a comma separated format (*.csv*) or data separated by blank spaces, using the same procedure with the "Time Series Wizard". This plug-in is in a development stage; check the plug-ins web page for updates.

Note: Excel uses beginning of period dates, whereas HEC-DSS uses end of period. Thus, the average data for a day will show at 0000 hours in Excel and 2400 hours in HEC-DSS. This can cause a time shift by a day if the data is not imported or exported carefully; Check your data after the operation. You can shift data to the correct times by using the **Time Functions** tab in the **Math Functions** screen.

5.4.1 Exporting Data to Microsoft Excel

To export data to Excel, simply select the data sets you want to export and then press the Excel Icon on the tool bar, or from the **Display** menu, click **Tabulate in Excel**. An example of data tabulated in Excel is shown in Figure 5.21. You can then save the Excel file to the name that you desire. Currently you cannot format the data in a different manner.



Microsoft Excel - HecDssExcel1832997272370710343.xls

	A	B	C	D	E	F
1	A		GREEN RIVER	GREEN RIVER	GREEN RIVER	GREEN RIVER
2	B		OAKVILLE	OAKVILLE	OAKVILLE	OAKVILLE
3	C		ELEVATION	FLOW-RES OUT	PRECIP-INC	STAGE-OUT
4	E					
5	F		OBS	OBS	OBS	OBS
6	Units		FEET	CFS	INCHES	FEET
7	Type		INST-VAL	INST-VAL	PER-CUM	INST-VAL
8	1	01May1992 0100	912.37	1575.0	0.00	3.6500
9	2	01May1992 0200	912.37	1575.0	0.00	3.6500
10	3	01May1992 0300	912.35	1575.0	0.00	3.6500
11	4	01May1992 0400	912.35	1575.0	0.00	3.6500
12	5	01May1992 0500	912.33	1575.0	0.00	3.6500
13	6	01May1992 0600	912.29	1575.0	0.00	3.6500
14	7	01May1992 0700	912.29	1575.0	0.00	3.6500
15	8	01May1992 0800	912.29	1575.0	0.00	3.6500
16	9	01May1992 0900	912.27	1575.0	0.00	3.6500
17	10	01May1992 1000	912.27	1575.0	0.00	3.6500
18	11	01May1992 1100	912.29	1575.0	0.00	3.6500
19	12	01May1992 1200		1575.0	0.00	3.6500
20	13	01May1992 1300		1575.0	0.00	3.6500
21	14	01May1992 1400		1575.0	0.00	3.6500
22	15	01May1992 1500		1575.0	0.00	3.6500
23	16	01May1992 1600		1575.0	0.00	3.6500
24	17	01May1992 1700		1575.0	0.00	3.6500
25	18	01May1992 1800		1575.0	0.00	3.6500
26	19	01May1992 1900		1575.0	0.00	3.6500
27	20	01May1992 2000		1575.0	0.00	3.6500
28	21	01May1992 2100		1575.0	0.00	3.6500
29	22	01May1992 2200		1575.0	0.00	3.6500
30	23	01May1992 2300		1575.0	0.00	3.6500
31	24	02May1992 0000		1575.0	0.00	3.6500
32	25	02May1992 0100	912.17	1575.0	0.00	3.6500

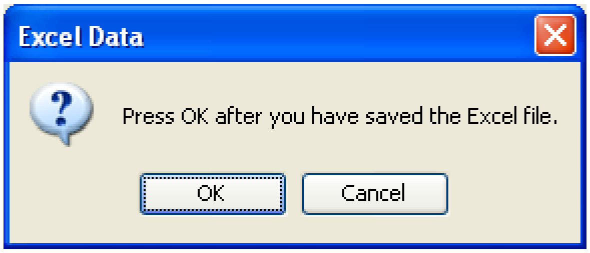
Figure 5.21 Tabulating Data in Excel

If the Excel icon is not displayed, then you need to download the *Excel.jar* file from the web page listed above and save it in the *HEC-DSSVue*

Plugins directory. This capability assumes that you have Microsoft Excel loaded on your computer or another program that is associated with ".*xls*" files.

5.4.2 Editing Data in Microsoft Excel

You can edit data in Microsoft Excel by selecting the pathnames that need to be edited. From the **Edit** menu, click **Edit in Excel**, an **Excel Data** dialog box will open (Figure 5.22), as well as Microsoft Excel. The data associated with the selected pathnames will be display in the spreadsheet.


Excel Data

Press OK after you have saved the Excel file.

Figure 5.22 Excel Data Dialog Box

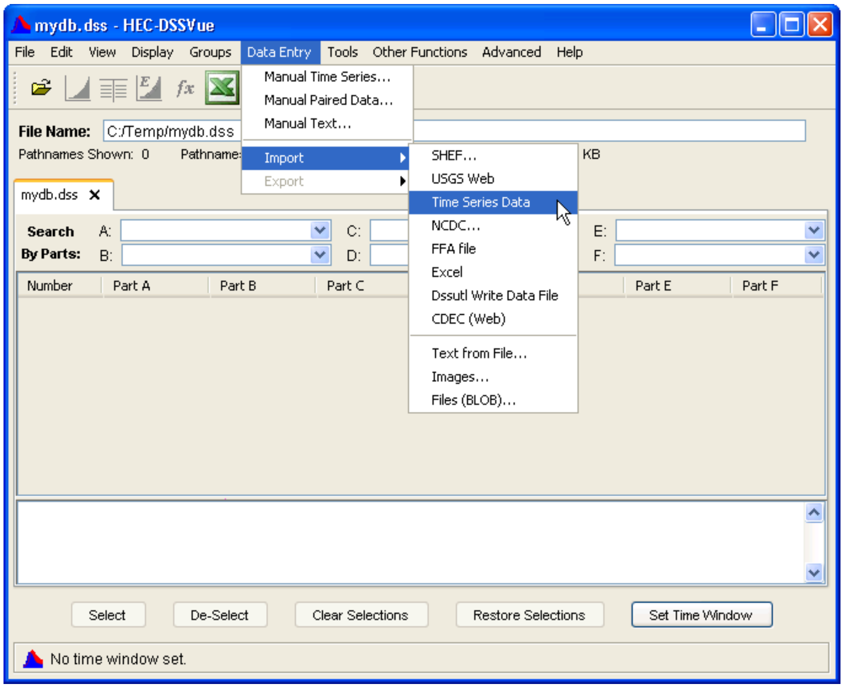
After you have made your changes, save the Excel file, and close Microsoft Excel. From the **Excel Data** dialog box click **OK,** the dialog box will close and the edits made to the data for those selected pathnames will be stored in the DSS file. If you do not want to save your changes back to HEC-DSS, click **Cancel.**

Note: You can only change data in Excel when you are editing - you cannot add or delete data. The data must remain in the same format with the same number of rows and columns to be stored back to HEC-DSS.

5.4.3 Importing Data from Microsoft Excel or ".*csv*" files using Data Entry

You can import time-series data from Excel, a ".*csv*" file, or a blank separated file using the **Data Entry** dialog box. At this time, you can only import time-series data (both regular interval and irregular interval data). Check the HEC-DSSVue plug-ins web page for updates.

1. From the HEC-DSSVue main window open a DSS file, from the **Data Entry** menu, point to **Import**, click **Excel** (see Figure 5.23, page 5-18).
2. If your data is in a ".*csv*" or blank separated file, select **Time Series Data** instead of **Excel**. An example of the browsing window is shown in Figure 5.24 (page 5-18).

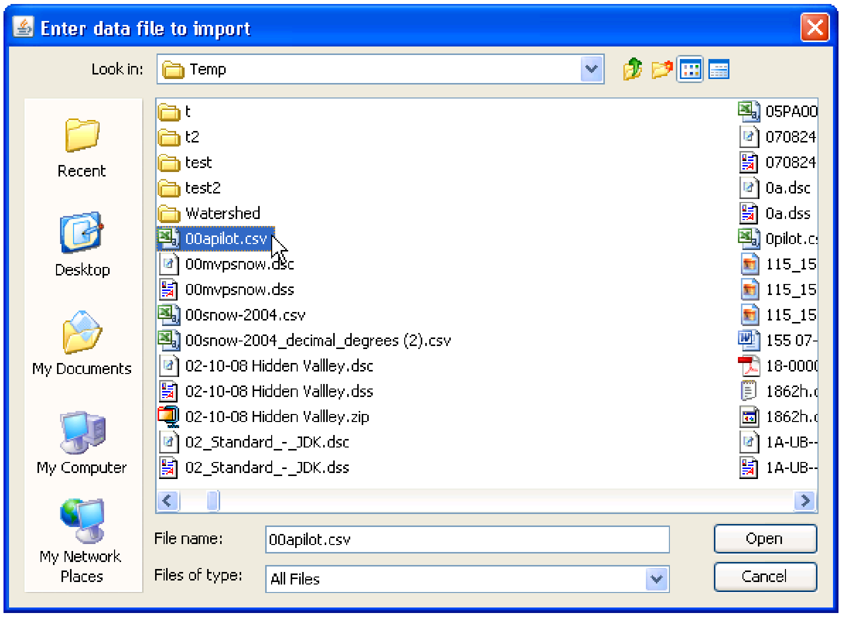


File Name: C:/Temp/mydb.dss

Pathnames Shown: 0

No time window set.

Figure 5.23 Importing Excel or .csv Files



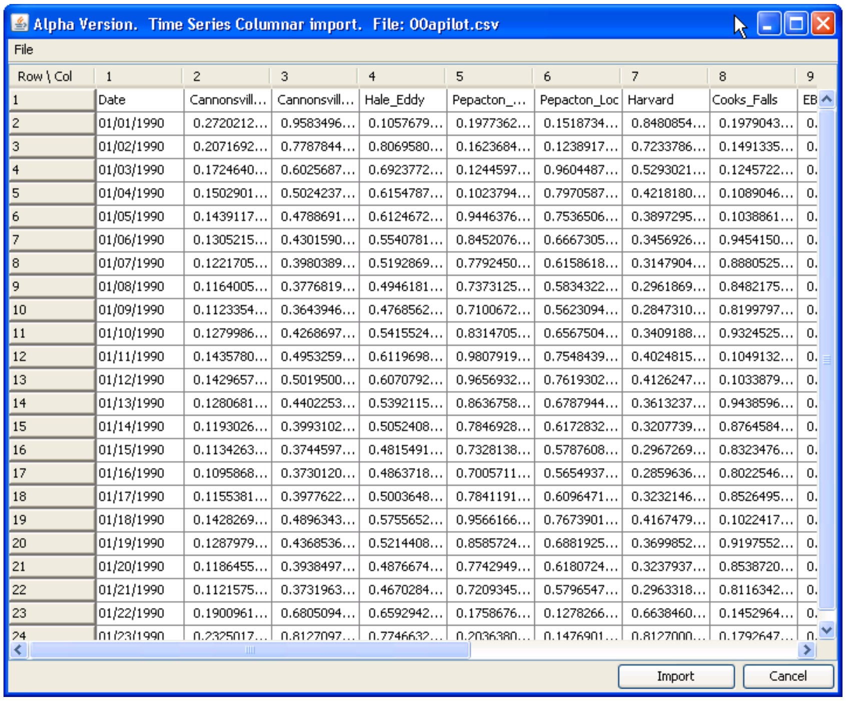
Look in: Temp

File name: 00apilot.csv

Files of type: All Files

Figure 5.24 Selection of an Excel or *.csv* File to import into HEC-DSSVue

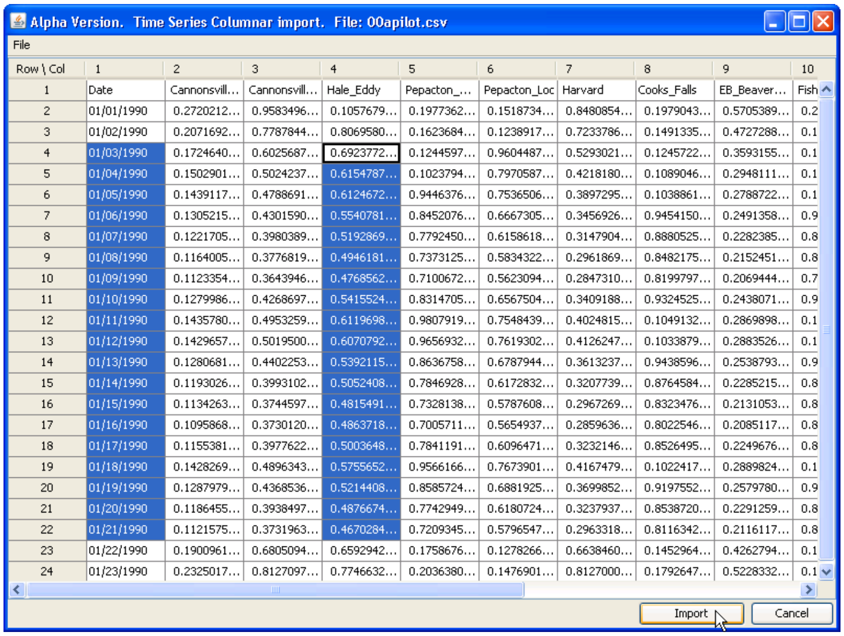
3. Select your file to import. Data should be in columns, separated by comma or spaces. Ideally, a row should contain a location or parameter, and a column should contain date/time.
4. The file will be read into an editable table as shown in Figure 5.25 (page 5-20).
5. For regular-interval time series data, select the column cells that you wish to import. For irregular-interval data, you need to select date/time column cells also.


Alpha Version. Time Series Columnar import. File: 00apilot.csv
File
Import
Cancel

Row \ Col	1	2	3	4	5	6	7	8	9
1	Date	Cannonsvill...	Cannonsvill...	Hale_Eddy	Pepacton_...	Pepacton_Loc	Harvard	Cooks_Falls	EB
2	01/01/1990	0.2720212...	0.9583496...	0.1057679...	0.1977362...	0.1518734...	0.8480854...	0.1979043...	0.
3	01/02/1990	0.2071692...	0.7787844...	0.8069580...	0.1623684...	0.1238917...	0.7233786...	0.1491335...	0.
4	01/03/1990	0.1724640...	0.6025687...	0.6923772...	0.1244597...	0.9604487...	0.5293021...	0.1245722...	0.
5	01/04/1990	0.1502901...	0.5024237...	0.6154787...	0.1023794...	0.7970587...	0.4218180...	0.1089046...	0.
6	01/05/1990	0.1439117...	0.4788691...	0.6124672...	0.9446376...	0.7536506...	0.3897295...	0.1038861...	0.
7	01/06/1990	0.1305215...	0.4301590...	0.5540781...	0.8452076...	0.6667305...	0.3456926...	0.9454150...	0.
8	01/07/1990	0.1221705...	0.3980389...	0.5192869...	0.7792450...	0.6158618...	0.3147904...	0.8880525...	0.
9	01/08/1990	0.1164005...	0.3776819...	0.4946181...	0.7373125...	0.5834322...	0.2961869...	0.8482175...	0.
10	01/09/1990	0.1123354...	0.3643946...	0.4768562...	0.7100672...	0.5623094...	0.2847310...	0.8199797...	0.
11	01/10/1990	0.1279986...	0.4268697...	0.5415524...	0.8314705...	0.6567504...	0.3409188...	0.9324525...	0.
12	01/11/1990	0.1435780...	0.4953259...	0.6119698...	0.9807919...	0.7548439...	0.4024815...	0.1049132...	0.
13	01/12/1990	0.1429657...	0.5019500...	0.6070792...	0.9656932...	0.7619302...	0.4126247...	0.1033879...	0.
14	01/13/1990	0.1280681...	0.4402253...	0.5392115...	0.8636758...	0.6787944...	0.3613237...	0.9438596...	0.
15	01/14/1990	0.1193026...	0.3993102...	0.5052408...	0.7846928...	0.6172832...	0.3207739...	0.8764584...	0.
16	01/15/1990	0.1134263...	0.3744597...	0.4815491...	0.7328138...	0.5787608...	0.2967269...	0.8323476...	0.
17	01/16/1990	0.1095868...	0.3730120...	0.4863718...	0.7005711...	0.5654937...	0.2859636...	0.8022546...	0.
18	01/17/1990	0.1155381...	0.3977622...	0.5003648...	0.7841191...	0.6096471...	0.3232146...	0.8526495...	0.
19	01/18/1990	0.1428269...	0.4896343...	0.5755652...	0.9566166...	0.7673901...	0.4167479...	0.1022417...	0.
20	01/19/1990	0.1287979...	0.4368536...	0.5214408...	0.8585724...	0.6881925...	0.3699852...	0.9197552...	0.
21	01/20/1990	0.1186455...	0.3938497...	0.4876674...	0.7742949...	0.6180724...	0.3237937...	0.8538720...	0.
22	01/21/1990	0.1121575...	0.3731963...	0.4670284...	0.7209345...	0.5796547...	0.2963318...	0.8116342...	0.
23	01/22/1990	0.1900961...	0.6805094...	0.6592942...	0.1758676...	0.1278266...	0.6638460...	0.1452964...	0.
24	01/23/1990	0.2325017...	0.8127097...	0.7746632...	0.2036380...	0.1476901...	0.8127000...	0.1792647...	0.

Figure 5.25 Editable Import Table of an Excel or *.csv* File

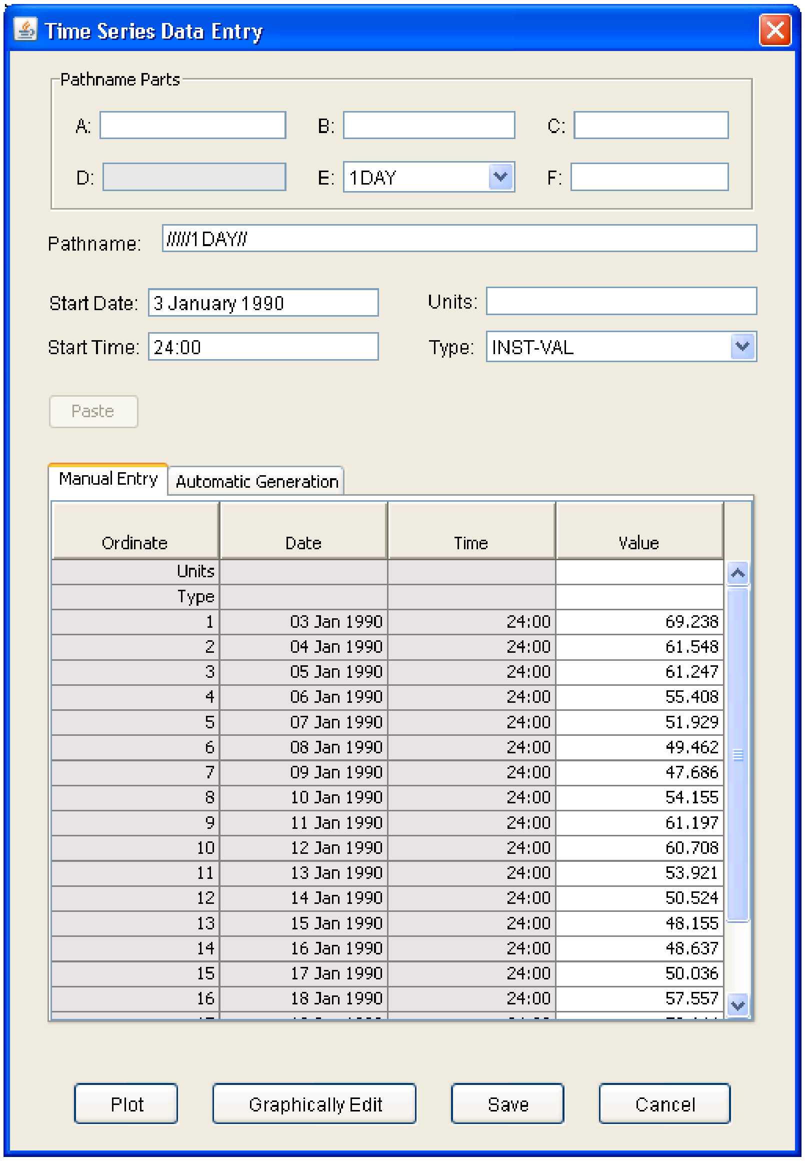
6. Press the **Import** button and your selected cells will be entered into the time series data entry dialog, as shown in Figure 5.26.


Alpha Version. Time Series Columnar import. File: 00apilot.csv
File
Import
Cancel

Row \ Col	1	2	3	4	5	6	7	8	9	10
1	Date	Cannonsvill...	Cannonsvill...	Hale_Eddy	Pepacton_...	Pepacton_Loc	Harvard	Cooks_Falls	EB_Beaver...	Fish
2	01/01/1990	0.2720212...	0.9583496...	0.1057679...	0.1977362...	0.1518734...	0.8480854...	0.1979043...	0.5705389...	0.2
3	01/02/1990	0.2071692...	0.7787844...	0.8069580...	0.1623684...	0.1238917...	0.7233786...	0.1491335...	0.4727288...	0.1
4	01/03/1990	0.1724640...	0.6025687...	0.6923772...	0.1244597...	0.9604487...	0.5293021...	0.1245722...	0.3593155...	0.1
5	01/04/1990	0.1502901...	0.5024237...	0.6154787...	0.1023794...	0.7970587...	0.4218180...	0.1089046...	0.2948111...	0.1
6	01/05/1990	0.1439117...	0.4788691...	0.6124672...	0.9446376...	0.7536506...	0.3897295...	0.1038861...	0.2788722...	0.1
7	01/06/1990	0.1305215...	0.4301590...	0.5540781...	0.8452076...	0.6667305...	0.3456926...	0.9454150...	0.2491358...	0.9
8	01/07/1990	0.1221705...	0.3980389...	0.5192869...	0.7792450...	0.6158618...	0.3147904...	0.8880525...	0.2282385...	0.8
9	01/08/1990	0.1164005...	0.3776819...	0.4946181...	0.7373125...	0.5834322...	0.2961869...	0.8482175...	0.2152451...	0.8
10	01/09/1990	0.1123354...	0.3643946...	0.4768562...	0.7100672...	0.5623094...	0.2847310...	0.8199797...	0.2069444...	0.7
11	01/10/1990	0.1279986...	0.4268697...	0.5415524...	0.8314705...	0.6567504...	0.3409188...	0.9324525...	0.2438071...	0.9
12	01/11/1990	0.1435780...	0.4953259...	0.6119698...	0.9807919...	0.7548439...	0.4024815...	0.1049132...	0.2869898...	0.1
13	01/12/1990	0.1429657...	0.5019500...	0.6070792...	0.9656932...	0.7619302...	0.4126247...	0.1033879...	0.2883526...	0.1
14	01/13/1990	0.1280681...	0.4402253...	0.5392115...	0.8636758...	0.6787944...	0.3613237...	0.9438596...	0.2538793...	0.9
15	01/14/1990	0.1193026...	0.3993102...	0.5052408...	0.7846928...	0.6172832...	0.3207739...	0.8764584...	0.2285215...	0.8
16	01/15/1990	0.1134263...	0.3744597...	0.4815491...	0.7328138...	0.5787608...	0.2967269...	0.8323476...	0.2131053...	0.8
17	01/16/1990	0.1095868...	0.3730120...	0.4863718...	0.7005711...	0.5654937...	0.2859636...	0.8022546...	0.2085117...	0.8
18	01/17/1990	0.1155381...	0.3977622...	0.5003648...	0.7841191...	0.6096471...	0.3232146...	0.8526495...	0.2249676...	0.8
19	01/18/1990	0.1428269...	0.4896343...	0.5755652...	0.9566166...	0.7673901...	0.4167479...	0.1022417...	0.2889824...	0.1
20	01/19/1990	0.1287979...	0.4368536...	0.5214408...	0.8585724...	0.6881925...	0.3699852...	0.9197552...	0.2579780...	0.9
21	01/20/1990	0.1186455...	0.3938497...	0.4876674...	0.7742949...	0.6180724...	0.3237937...	0.8538720...	0.2291259...	0.8
22	01/21/1990	0.1121575...	0.3731963...	0.4670284...	0.7209345...	0.5796547...	0.2963318...	0.8116342...	0.2116117...	0.8
23	01/22/1990	0.1900961...	0.6805094...	0.6592942...	0.1758676...	0.1278266...	0.6638460...	0.1452964...	0.4262794...	0.1
24	01/23/1990	0.2325017...	0.8127097...	0.7746632...	0.2036380...	0.1476901...	0.8127000...	0.1792647...	0.5228332...	0.1

Figure 5.26 Selecting and Importing Data

7. Enter the pathname parts, units and type, and then press **Save**. An example of the **Time Series Data Entry** window that appears after data have been imported is shown in Figure 5.27 (page 5-21).



Time Series Data Entry

Pathname Parts

A:

B:

C:

D:

E: 1DAY

F:

Pathname: /////1DAY//

Start Date: 3 January 1990

Units:

Start Time: 24:00

Type: INST-VAL

Ordinate	Date	Time	Value
Units			
Type			
1	03 Jan 1990	24:00	69.238
2	04 Jan 1990	24:00	61.548
3	05 Jan 1990	24:00	61.247
4	06 Jan 1990	24:00	55.408
5	07 Jan 1990	24:00	51.929
6	08 Jan 1990	24:00	49.462
7	09 Jan 1990	24:00	47.686
8	10 Jan 1990	24:00	54.155
9	11 Jan 1990	24:00	61.197
10	12 Jan 1990	24:00	60.708
11	13 Jan 1990	24:00	53.921
12	14 Jan 1990	24:00	50.524
13	15 Jan 1990	24:00	48.155
14	16 Jan 1990	24:00	48.637
15	17 Jan 1990	24:00	50.036
16	18 Jan 1990	24:00	57.557

Figure 5.27 Time Series Data Entry Dialog Box

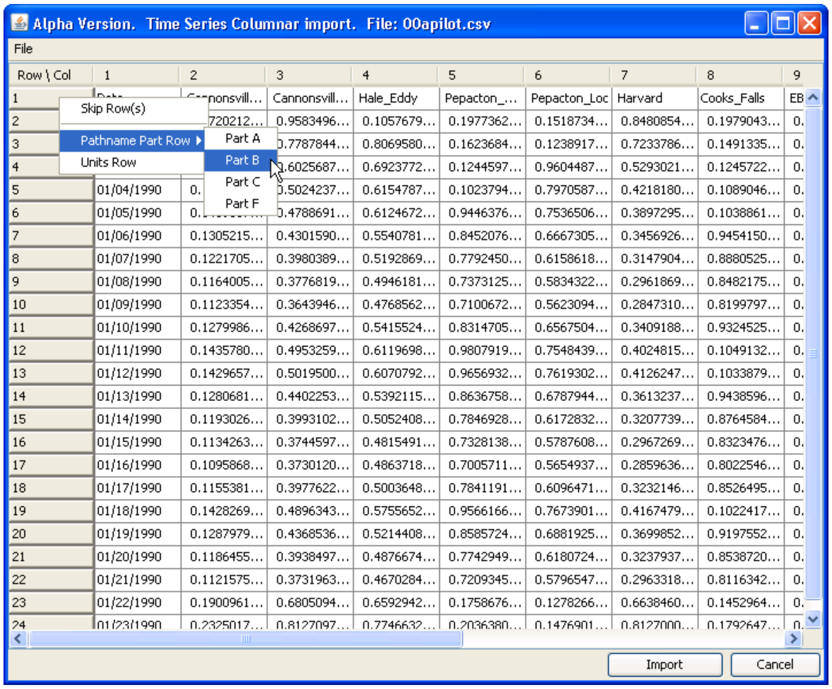
Note: Excel uses beginning of period dates, whereas HEC-DSS uses end of period. Thus, the average data for a day will show at 0000 hours in Excel and 2400 hours in HEC-DSS. This can cause a time shift by a day if the data is not imported or exported carefully; Check your data after the operation. You can shift data to the correct times by using the **Time Functions** tab in the **Math Functions** screen.

5.4.4 Importing Data from Microsoft Excel or "*.csv*" files using Time Series Wizard

You can import multiple columns of time-series data from Excel, a "*.csv*" file, or a blank separated file using the **Time Series Wizard**. The **Time**

Series Wizard is a plug-in that is distributed in a development stage; check the HEC-DSSVue plug-ins web page for updates.

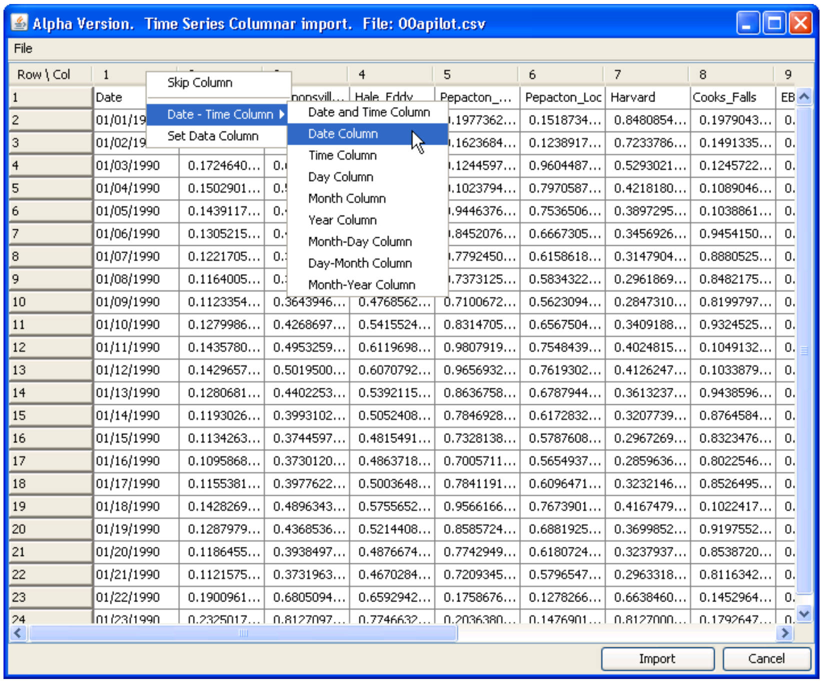
1. Open your DSS file from the **Data Entry** men, point to Import, click and click **Excel** (see Figure 5.23, page 5-19).
2. If your data is in a "*.csv*" or blank separated file, click **Time Series Data** instead of **Excel**.
3. Select your file to import (see Figure 5.24, page 5-19). Data should be in columns, separated by comma or spaces. Ideally, a row should contain a location or parameter, and a column should contain date/time.
4. The file will be read into an editable table as shown in Figure 5.25 (page 5-20).
5. Select any location, parameter, units, or version, etc. rows by right clicking on the header row in the first column and selecting the item, as shown in Figure 5.28. You may also skip any rows that are not to be imported or used.



Row \ Col	1	2	3	4	5	6	7	8	9
1	Date	Cannonsvill...	Cannonsvill...	Hale_Eddy	Pepacton_...	Pepacton_Loc	Harvard	Cooks_Falls	EB
2		720212...	0.9583496...	0.1057679...	0.1977362...	0.1518734...	0.8480854...	0.1979043...	0.
3			.7787844...	0.8069580...	0.1623684...	0.1238917...	0.7233786...	0.1491335...	0.
4			.6025687...	0.6923772...	0.1244597...	0.9604487...	0.5293021...	0.1245722...	0.
5	01/04/1990	0.	.5024237...	0.6154787...	0.1023794...	0.7970587...	0.4218180...	0.1089046...	0.
6	01/05/1990	0.	.4788691...	0.6124672...	0.9446376...	0.7536506...	0.3897295...	0.1038861...	0.
7	01/06/1990	0.1305215...	0.4301590...	0.5540781...	0.8452076...	0.6667305...	0.3456926...	0.9454150...	0.
8	01/07/1990	0.1221705...	0.3980389...	0.5192869...	0.7792450...	0.6158618...	0.3147904...	0.8880525...	0.
9	01/08/1990	0.1164005...	0.3776819...	0.4946181...	0.7373125...	0.5834322...	0.2961869...	0.8482175...	0.
10	01/09/1990	0.1123354...	0.3643946...	0.4768562...	0.7100672...	0.5623094...	0.2847310...	0.8199797...	0.
11	01/10/1990	0.1279986...	0.4268697...	0.5415524...	0.8314705...	0.6567504...	0.3409188...	0.9324525...	0.
12	01/11/1990	0.1435780...	0.4953259...	0.6119698...	0.9807919...	0.7548439...	0.4024815...	0.1049132...	0.
13	01/12/1990	0.1429657...	0.5019500...	0.6070792...	0.9656932...	0.7619302...	0.4126247...	0.1033879...	0.
14	01/13/1990	0.1280681...	0.4402253...	0.5392115...	0.8636758...	0.6787944...	0.3613237...	0.9438596...	0.
15	01/14/1990	0.1193026...	0.3993102...	0.5052408...	0.7846928...	0.6172832...	0.3207739...	0.8764584...	0.
16	01/15/1990	0.1134263...	0.3744597...	0.4815491...	0.7328138...	0.5787608...	0.2967269...	0.8323476...	0.
17	01/16/1990	0.1095868...	0.3730120...	0.4863718...	0.7005711...	0.5654937...	0.2859636...	0.8022546...	0.
18	01/17/1990	0.1155381...	0.3977622...	0.5003648...	0.7841191...	0.6096471...	0.3232146...	0.8526495...	0.
19	01/18/1990	0.1428269...	0.4896343...	0.5755652...	0.9566166...	0.7673901...	0.4167479...	0.1022417...	0.
20	01/19/1990	0.1287979...	0.4368536...	0.5214408...	0.8585724...	0.6881925...	0.3699852...	0.9197552...	0.
21	01/20/1990	0.1186455...	0.3938497...	0.4876674...	0.7742949...	0.6180724...	0.3237937...	0.8538720...	0.
22	01/21/1990	0.1121575...	0.3731963...	0.4670284...	0.7209345...	0.5796547...	0.2963318...	0.8116342...	0.
23	01/22/1990	0.1900961...	0.6805094...	0.6592942...	0.1758676...	0.1278266...	0.6638460...	0.1452964...	0.
24	01/23/1990	0.2325017...	0.8127097...	0.7746632...	0.2036380...	0.1476901...	0.8127000...	0.1792647...	0.

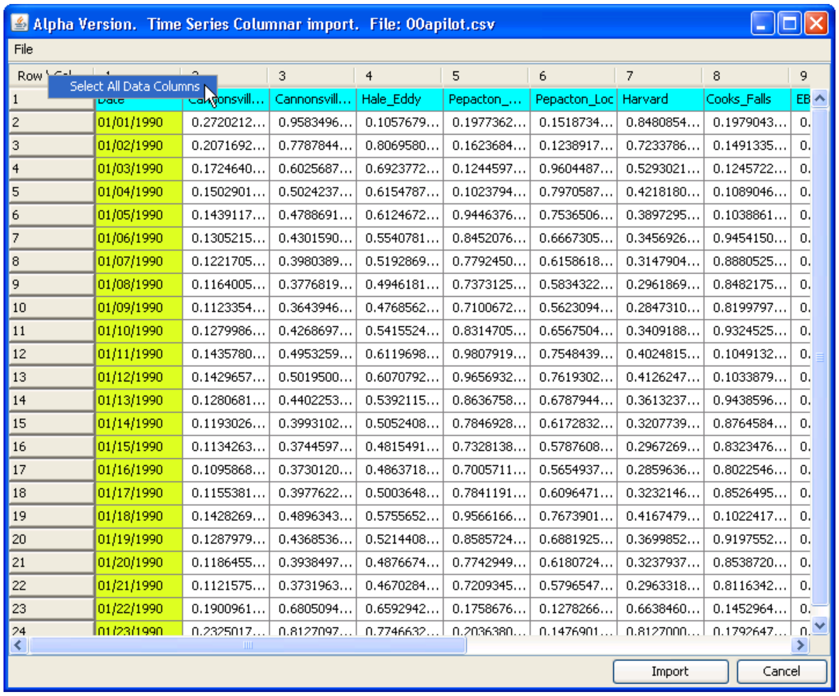
Figure 5.28 Assigning Locations, Parameters, Units, Versions, etc. to Rows

6. From the top header row, select any columns that contain dates or times, as shown in Figure 5.29 (page 5-23).
7. You can also select individual data columns from the header popup.
8. Alternatively, you can enter all data into DSS at one time by right-clicking the upper left cell and selecting **Set All Data Columns**, as shown in Figure 5.30 (page 5-23). The resulting window will look like the example in Figure 5.31 (page 5-24).



Row \ Col	1			4	5	6	7	8	9
1	Date		nonsvill…	Hale_Eddy	Pepacton_…	Pepacton_Loc	Harvard	Cooks_Falls	EB
2	01/01/19				.1977362…	0.1518734…	0.8480854…	0.1979043…	0.
3	01/02/19				.1623684…	0.1238917…	0.7233786…	0.1491335…	0.
4	01/03/1990	0.1724640…	0.		.1244597…	0.9604487…	0.5293021…	0.1245722…	0.
5	01/04/1990	0.1502901…	0.5		.1023794…	0.7970587…	0.4218180…	0.1089046…	0.
6	01/05/1990	0.1439117…	0.		.9446376…	0.7536506…	0.3897295…	0.1038861…	0.
7	01/06/1990	0.1305215…	0.		.8452076…	0.6667305…	0.3456926…	0.9454150…	0.
8	01/07/1990	0.1221705…	0.		.7792450…	0.6158618…	0.3147904…	0.8880525…	0.
9	01/08/1990	0.1164005…	0.		.7373125…	0.5834322…	0.2961869…	0.8482175…	0.
10	01/09/1990	0.1123354…	0.3643946…	0.4768562…	0.7100672…	0.5623094…	0.2847310…	0.8199797…	0.
11	01/10/1990	0.1279986…	0.4268697…	0.5415524…	0.8314705…	0.6567504…	0.3409188…	0.9324525…	0.
12	01/11/1990	0.1435780…	0.4953259…	0.6119698…	0.9807919…	0.7548439…	0.4024815…	0.1049132…	0.
13	01/12/1990	0.1429657…	0.5019500…	0.6070792…	0.9656932…	0.7619302…	0.4126247…	0.1033879…	0.
14	01/13/1990	0.1280681…	0.4402253…	0.5392115…	0.8636758…	0.6787944…	0.3613237…	0.9438596…	0.
15	01/14/1990	0.1193026…	0.3993102…	0.5052408…	0.7846928…	0.6172832…	0.3207739…	0.8764584…	0.
16	01/15/1990	0.1134263…	0.3744597…	0.4815491…	0.7328138…	0.5787608…	0.2967269…	0.8323476…	0.
17	01/16/1990	0.1095868…	0.3730120…	0.4863718…	0.7005711…	0.5654937…	0.2859636…	0.8022546…	0.
18	01/17/1990	0.1155381…	0.3977622…	0.5003648…	0.7841191…	0.6096471…	0.3232146…	0.8526495…	0.
19	01/18/1990	0.1428269…	0.4896343…	0.5755652…	0.9566166…	0.7673901…	0.4167479…	0.1022417…	0.
20	01/19/1990	0.1287979…	0.4368536…	0.5214408…	0.8585724…	0.6881925…	0.3699852…	0.9197552…	0.
21	01/20/1990	0.1186455…	0.3938497…	0.4876674…	0.7742949…	0.6180724…	0.3237937…	0.8538720…	0.
22	01/21/1990	0.1121575…	0.3731963…	0.4670284…	0.7209345…	0.5796547…	0.2963318…	0.8116342…	0.
23	01/22/1990	0.1900961…	0.6805094…	0.6592942…	0.1758676…	0.1278266…	0.6638460…	0.1452964…	0.
24	01/23/1990	0.2325017…	0.8127097…	0.7746632…	0.2036380…	0.1476901…	0.8127000…	0.1792647…	0.

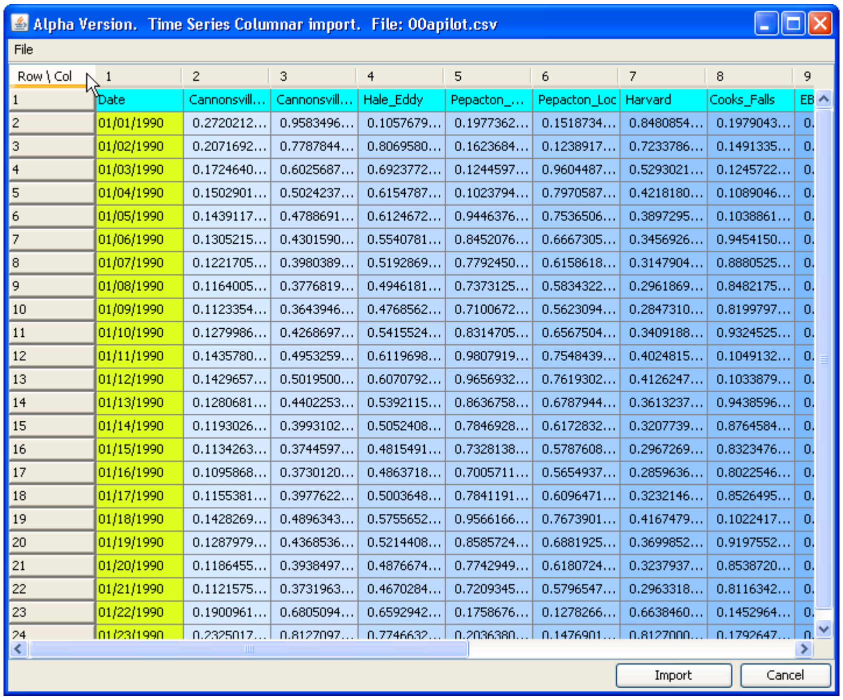
Figure 5.29 Assigning Data and Time to Columns


Alpha Version. Time Series Columnar import. File: 00apilot.csv
File
Select All Data Columns
Import
Cancel

Row \ Col	1	2	3	4	5	6	7	8	9
1	Date	Cannonsvill…	Cannonsvill…	Hale_Eddy	Pepacton_…	Pepacton_Loc	Harvard	Cooks_Falls	EB
2	01/01/1990	0.2720212…	0.9583496…	0.1057679…	0.1977362…	0.1518734…	0.8480854…	0.1979043…	0.
3	01/02/1990	0.2071692…	0.7787844…	0.8069580…	0.1623684…	0.1238917…	0.7233786…	0.1491335…	0.
4	01/03/1990	0.1724640…	0.6025687…	0.6923772…	0.1244597…	0.9604487…	0.5293021…	0.1245722…	0.
5	01/04/1990	0.1502901…	0.5024237…	0.6154787…	0.1023794…	0.7970587…	0.4218180…	0.1089046…	0.
6	01/05/1990	0.1439117…	0.4788691…	0.6124672…	0.9446376…	0.7536506…	0.3897295…	0.1038861…	0.
7	01/06/1990	0.1305215…	0.4301590…	0.5540781…	0.8452076…	0.6667305…	0.3456926…	0.9454150…	0.
8	01/07/1990	0.1221705…	0.3980389…	0.5192869…	0.7792450…	0.6158618…	0.3147904…	0.8880525…	0.
9	01/08/1990	0.1164005…	0.3776819…	0.4946181…	0.7373125…	0.5834322…	0.2961869…	0.8482175…	0.
10	01/09/1990	0.1123354…	0.3643946…	0.4768562…	0.7100672…	0.5623094…	0.2847310…	0.8199797…	0.
11	01/10/1990	0.1279986…	0.4268697…	0.5415524…	0.8314705…	0.6567504…	0.3409188…	0.9324525…	0.
12	01/11/1990	0.1435780…	0.4953259…	0.6119698…	0.9807919…	0.7548439…	0.4024815…	0.1049132…	0.
13	01/12/1990	0.1429657…	0.5019500…	0.6070792…	0.9656932…	0.7619302…	0.4126247…	0.1033879…	0.
14	01/13/1990	0.1280681…	0.4402253…	0.5392115…	0.8636758…	0.6787944…	0.3613237…	0.9438596…	0.
15	01/14/1990	0.1193026…	0.3993102…	0.5052408…	0.7846928…	0.6172832…	0.3207739…	0.8764584…	0.
16	01/15/1990	0.1134263…	0.3744597…	0.4815491…	0.7328138…	0.5787608…	0.2967269…	0.8323476…	0.
17	01/16/1990	0.1095868…	0.3730120…	0.4863718…	0.7005711…	0.5654937…	0.2859636…	0.8022546…	0.
18	01/17/1990	0.1155381…	0.3977622…	0.5003648…	0.7841191…	0.6096471…	0.3232146…	0.8526495…	0.
19	01/18/1990	0.1428269…	0.4896343…	0.5755652…	0.9566166…	0.7673901…	0.4167479…	0.1022417…	0.
20	01/19/1990	0.1287979…	0.4368536…	0.5214408…	0.8585724…	0.6881925…	0.3699852…	0.9197552…	0.
21	01/20/1990	0.1186455…	0.3938497…	0.4876674…	0.7742949…	0.6180724…	0.3237937…	0.8538720…	0.
22	01/21/1990	0.1121575…	0.3731963…	0.4670284…	0.7209345…	0.5796547…	0.2963318…	0.8116342…	0.
23	01/22/1990	0.1900961…	0.6805094…	0.6592942…	0.1758676…	0.1278266…	0.6638460…	0.1452964…	0.
24	01/23/1990	0.2325017…	0.8127097…	0.7746632…	0.2036380…	0.1476901…	0.8127000…	0.1792647…	0.

Figure 5.30 Select all Data and Columns to Import

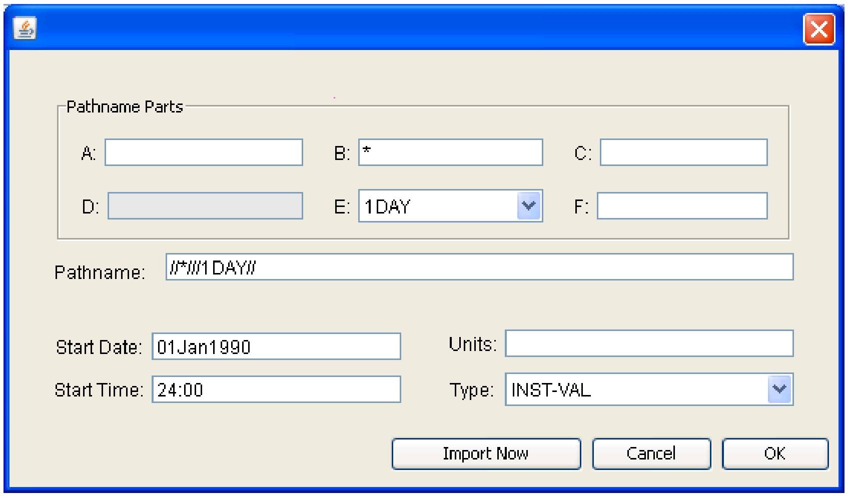
9. This will bring up a dialog to set additional pathname parts or units, as shown in Figure 5.32 (page 5-24).
10. Enter additional information desired. To import all the data sets, click **Import Now**. A confirmation window will appear, see Figure 5.33 (page 5-24).
11. Alternatively, if all columns are not to be imported, individual data set columns can be imported by right clicking on the header row of



Alpha Version. Time Series Columnar import. File: 00apilot.csv

Row \ Col	1	2	3	4	5	6	7	8	9
1	Date	Cannonsvill…	Cannonsvill…	Hale_Eddy	Pepacton_…	Pepacton_Loc	Harvard	Cooks_Falls	EB
2	01/01/1990	0.2720212…	0.9583496…	0.1057679…	0.1977362…	0.1518734…	0.8480854…	0.1979043…	0.
3	01/02/1990	0.2071692…	0.7787844…	0.8069580…	0.1623684…	0.1238917…	0.7233786…	0.1491335…	0.
4	01/03/1990	0.1724640…	0.6025687…	0.6923772…	0.1244597…	0.9604487…	0.5293021…	0.1245722…	0.
5	01/04/1990	0.1502901…	0.5024237…	0.6154787…	0.1023794…	0.7970587…	0.4218180…	0.1089046…	0.
6	01/05/1990	0.1439117…	0.4788691…	0.6124672…	0.9446376…	0.7536506…	0.3897295…	0.1038861…	0.
7	01/06/1990	0.1305215…	0.4301590…	0.5540781…	0.8452076…	0.6667305…	0.3456926…	0.9454150…	0.
8	01/07/1990	0.1221705…	0.3980389…	0.5192869…	0.7792450…	0.6158618…	0.3147904…	0.8880525…	0.
9	01/08/1990	0.1164005…	0.3776819…	0.4946181…	0.7373125…	0.5834322…	0.2961869…	0.8482175…	0.
10	01/09/1990	0.1123354…	0.3643946…	0.4768562…	0.7100672…	0.5623094…	0.2847310…	0.8199797…	0.
11	01/10/1990	0.1279986…	0.4268697…	0.5415524…	0.8314705…	0.6567504…	0.3409188…	0.9324525…	0.
12	01/11/1990	0.1435780…	0.4953259…	0.6119698…	0.9807919…	0.7548439…	0.4024815…	0.1049132…	0.
13	01/12/1990	0.1429657…	0.5019500…	0.6070792…	0.9656932…	0.7619302…	0.4126247…	0.1033879…	0.
14	01/13/1990	0.1280681…	0.4402253…	0.5392115…	0.8636758…	0.6787944…	0.3613237…	0.9438596…	0.
15	01/14/1990	0.1193026…	0.3993102…	0.5052408…	0.7846928…	0.6172832…	0.3207739…	0.8764584…	0.
16	01/15/1990	0.1134263…	0.3744597…	0.4815491…	0.7328138…	0.5787608…	0.2967269…	0.8323476…	0.
17	01/16/1990	0.1095868…	0.3730120…	0.4863718…	0.7005711…	0.5654937…	0.2859636…	0.8022546…	0.
18	01/17/1990	0.1155381…	0.3977622…	0.5003648…	0.7841191…	0.6096471…	0.3232146…	0.8526495…	0.
19	01/18/1990	0.1428269…	0.4896343…	0.5755652…	0.9566166…	0.7673901…	0.4167479…	0.1022417…	0.
20	01/19/1990	0.1287979…	0.4368536…	0.5214408…	0.8585724…	0.6881925…	0.3699852…	0.9197552…	0.
21	01/20/1990	0.1186455…	0.3938497…	0.4876674…	0.7742949…	0.6180724…	0.3237937…	0.8538720…	0.
22	01/21/1990	0.1121575…	0.3731963…	0.4670284…	0.7209345…	0.5796547…	0.2963318…	0.8116342…	0.
23	01/22/1990	0.1900961…	0.6805094…	0.6592942…	0.1758676…	0.1278266…	0.6638460…	0.1452964…	0.
24	01/23/1990	0.2325017…	0.8127097…	0.7746632…	0.2036380…	0.1476901…	0.8127000…	0.1792647…	0.

Figure 5.31 Window after Selecting Select All Data Columns



Pathname Parts

A:

B: *

C:

D:

E: 1DAY

F:

Pathname: //*///1DAY//

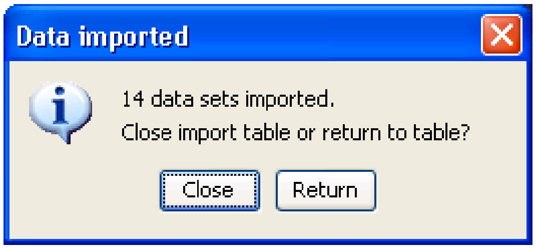
Start Date: 01Jan1990

Start Time: 24:00

Units:

Type: INST-VAL

Figure 5.32 Import Data Window



Data imported

14 data sets imported.
Close import table or return to table?

Figure 5.33 Import Data Confirmation Window

the column and selecting **Set Data Column**. This will bring up the window shown in Figure 5.32.

12. Fill in the additional information and click **Import Now** if you are finished importing data from this file, or click **OK** to select other columns. When you have selected all the columns you want to import, click **Import Now**, or click **Import** at the bottom of the table. A confirmation window will appear (see Figure 5.33, page 5-24).

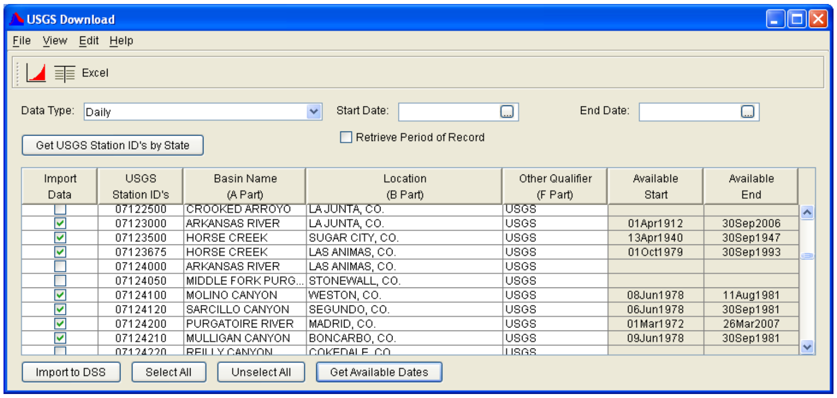
5.5 Retrieving and Importing USGS (NWIS) Data

You can retrieve and import time-series data from the USGS NWIS database via the web using the **USGS** plug-in tool. The USGS occasionally updates the NWS database with more information and the **USGS** plug-in may be updated to reflect changes; check the HEC-DSSVue plug-ins web page for updates.

The **USGS** data retrieval plug-in will retrieve daily, hourly and annual peak flows from the NWIS web site. Daily, real-time and peak data availability can be obtained from the USGS water web site at: http://waterdata.usgs.gov/nwis. Hourly data information can be obtained from the USGS IDA (Instantaneous Data Archive) at http://ida.water.usgs.gov. The IDA is in a developmental phase and only a limited number of stations have historical short-term data available.

To use the USGS plug-in, you must be connected to the internet and the file *USGS.jar* must be located in the **Plugins** directory for HEC-DSSVue.

1. Open your DSS file, from the **Data Entry** menu, point to **Import**, click **USGS Web**. This will open the USGS Download plug-in interface with an empty station identifier table.
2. You can either open a previously saved table, manually enter the station information for the stations that you want, or automatically populate the table with a list of stations available on a state by state basis. An example of the **USGS Download** window can be seen in Figure 5.34.



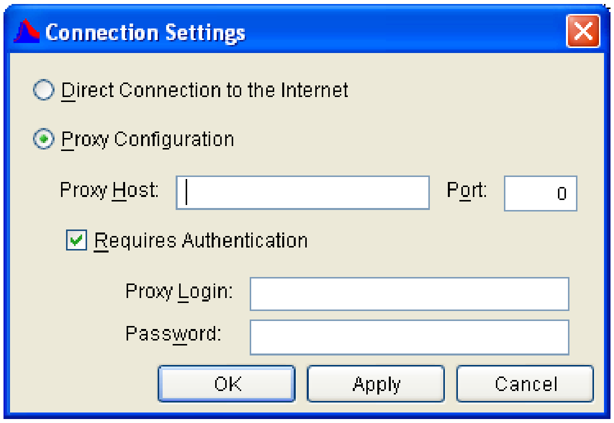
Data Type: Daily
Start Date:
End Date:
□ Retrieve Period of Record

Import Data	USGS Station ID's	Basin Name (A Part)	Location (B Part)	Other Qualifier (F Part)	Available Start	Available End
□	07122500	CROOKED ARROYO	LA JUNTA, CO.	USGS		
☑	07123000	ARKANSAS RIVER	LA JUNTA, CO.	USGS	01Apr1912	30Sep2006
☑	07123500	HORSE CREEK	SUGAR CITY, CO.	USGS	13Apr1940	30Sep1947
☑	07123675	HORSE CREEK	LAS ANIMAS, CO.	USGS	01Oct1979	30Sep1993
□	07124000	ARKANSAS RIVER	LAS ANIMAS, CO.	USGS		
□	07124050	MIDDLE FORK PURG...	STONEWALL, CO.	USGS		
☑	07124100	MOLINO CANYON	WESTON, CO.	USGS	08Jun1978	11Aug1981
☑	07124120	SARCILLO CANYON	SEGUNDO, CO.	USGS	06Jun1978	30Sep1981
☑	07124200	PURGATOIRE RIVER	MADRID, CO.	USGS	01Mar1972	26Mar2007
☑	07124210	MULLIGAN CANYON	BONCARBO, CO.	USGS	09Jun1978	30Sep1981
□	07124220	REILLY CANYON	COKEDALE, CO.	USGS		

Figure 5.34 USGS Download Window

3. To open a previously saved table, from the **File** menu, click **Open Station Table.** Browse to the directory and file that you want to access and then click **Open**. Optionally, you can select a file from the most recently used file list at the bottom of the **File** menu. To save a table, from the **File** menu, click **Save Station Table**. Browse to the directory where you want to save your table and then enter the file name and click **Save**. Tables are in text format and are saved with an extension of ".usgs".
4. To manually enter station information, select the USGS Station ID cell on a blank row. Enter the USGS station ID number and then fill in the basin or river name, location name, parameter and version in the subsequent columns or accept the defaults provided after entering the station ID. If you need to add additional rows, select the **Edit** menu and then **Insert Rows**. Fill in the identifiers for any remaining stations that you want. Once you have completed your table, save it using **Save Station Table** from the **File** menu.
5. You can sort the station table by pressing on the column header of the column that you want to sort by. A second press of the column header will perform an inverse sort.
6. You may find the list of stations easier to use if you delete those stations that you are not interested in. To do this, highlight the rows in the table that you want to delete and then select **Delete Checked Rows** from the **Edit** menu. You can save your table by selecting **Save Station Table** from the **File** menu.
7. You can query for the starting date and ending date of available data for selected stations. To do this, select the check boxes for the stations that you want and then press the **Get Available Dates** button. Each station has to be queried independently, so select only those stations that you are interested in. The date range for the selected stations will be displayed in the two right most columns of the table.
8. To retrieve data from the USGS web site and import into HEC-DSS, select the **Data Type** you want (either Daily, Instantaneous (Hourly), Real Time or Annual Peak Data) from the drop down selector near the top panel of the screen. If you want daily, you can enter a start date and end date for the time frame that you are interested in or select the **Retrieve Period of Record** check box to retrieve the period of record data. The date should be entered in the form of DDMMMYYYY (e.g., 03FEB2005). Selecting the small box in the date field will provide a calendar tool to aid in setting the date. Even though you may specify dates, only data that is available within those dates from the web site can be retrieved. If you want real time data, enter the number of days back that you want to retrieve data for. The USGS offers data for the last thrity-one days for real time data.

9. Use caution when selecting a time period for instantaneous historical data, as you might unintentionally request excessive number of values. For example, if data is given in 15 minute intervals, a year's worth of data will be 35,040 values, which can take a long time to retrieve.
10. After the data type and time span have been set, select the stations from the table that you want data for by checking the box in the **Import Data** column for those stations. If you want to retrieve data for all stations in your table, press the **Select All** button. The data retrieval process will begin when you press the **Import** button. This operation will take some time, depending on your connection speed to the internet and how much data you have requested. A **Retrieve Progress** dialog box will display the progress of the process. After data has been retrieved, the main catalog screen in HEC-DSSVue will be automatically updated.
11. Daily data is stored with an E part of "1DAY". The interval for real-time data will be set based on the frequency of the data retrieved. If the frequency varies, the data will be store in an irregular interval format. You may want to use the interval modification functions available from the **Time Functions** tab in the **Math Functions** screen to change the data to a desired interval.
12. Data can be directly plotted, tabulated or sent to Microsoft Excel from the plug-in by selecting the desired data sets and then pressing the appropriate tool bar button. You do not need to import the data first; it will be automatically imported and stored in the HEC-DSS file for you. To send the data to Microsoft Excel, you must have installed the Excel plug-in, available from the HEC-DSSVue plug-in web page, or you must have an Excel capable version (currently under development) of the program.
13. Some offices require proxy information to be set to access the internet. Proxy information can be set by selecting the **Options** item from the **Edit** menu. This will display the screen shown in Figure 5.35. Fill in the appropriate values and then click **OK**. The information will be retained between sessions.



○ Direct Connection to the Internet

● Proxy Configuration

Proxy Host: | Port: 0

☑ Requires Authentication

Proxy Login:

Password:

Figure 5.35 Connection Settings Dialog Box

14. The USGS plug-in currently has limited error detection capabilities. If you think that there are problems, you should check the HEC-DSSVue log screen for messages. The log screen is usually minimized in your task bar.

    The USGS plug-in will indicate when data cannot be retrieved, but it may not give the reason why. You should check the USGS web page with a browser to ensure that what you are requesting is available. The web address is: http://waterdata.usgs.gov/nwis for daily and real-time data, or http://ida.water.usgs.gov/ida for hourly data.

5.6 Importing NCDC Data

You can optionally import climatic data from the National Climatic Data Center using the **NCDC** plug-in tool. It cannot retrieve data from NCDC; NCDC provides climatic data via ftp. This tool will import data in the ftp file that you received.

The **NCDC** data import plug-in will import data in the following formats: 3200 and 3210 (Daily Climatic Data), 3220 (Monthly Surface Data), 3240 (Hourly Precipitation Data), 3260 (15-minute precipitation data), LCD and 3505 (Local Climatic Data), and the GSOD (Global Surface Summary of Day). The plug-in will import all data from the data file provided using default names and units. Users may want to rename parts of pathname and/or delete unwanted data sets.

5.6.1 Obtaining NCDC Data

Data can be obtained via an ftp download from the NCDC web site at: http://www.ncdc.noaa.gov/oa/ncdc.html. The 3240 data is found at the address: http://ols.nndc.noaa.gov/plolstore/plsql/olstore.prodspecific?prodnum=C00313-TAP-A0001.

You can obtain a list of the available stations (near the bottom of the page) after you click the link in the page above to enter the NCDC store. We have found it more useful to select stations in the county you are interested in and allow it to return the data that it has. To do this, select the Country. In that page, select the state and then select Counties and then select your county(ies) and the date range that you are interested in. We recommend a comma delimited format with station name.

Check the "Inventory Review" box, enter your email address and then press "Submit Request". The server will generate a data file and email you when it is ready. Click on the link in your email (or the one in the last page you were on, if you waited long enough). This should display the data file in your internet browser. Save this file to disk and select it from the NCDC import plug-in. The plug-in will remove any of the html format that might be present. Data may be free for some government and educational internet addresses.

5.6.2 Recognized NCDC Formats

The plug-in has been tested with NCDC 3200, 3210, 3220, 3240 and 3260 formats in all forms. It has also been tested with LCD, GSOD and 3505 delimited formats. It will not import data from non-delimited LCD, GSOD and 3505 formats. If you have a choice for your format, we recommend comma delimited with station names. Many of the parameter types in 3200, 3220, LCD, GSOD and 3505 formats do not have the units or type assigned by the importer and they may need to be edited in later, if needed.

Examples of these formats are:

1. 3240 Comma Delimited:

```
COOPID,CD,ELEM,UN,YEAR,MO,DA,TIME,HOUR01,F,F,TIME,HOUR02…
------,--,----,--,----,--,--,----,------,-,-,----,------
041112,00,HPCP,HI,1970,09,01,0100, 00000,g, ,0200, 00000,…
041112,00,HPCP,HI,1970,09,04,0100, 00000, , ,0200, 00000,…
041112,00,HPCP,HI,1970,10,01,0100, 00000,g, ,0200, 00000,…
…
```

2. 3240 Space Delimited with Station Name:

```
COOPID STATION NAME                   CD ELEM UN YEAR MO DA TIME
HOUR01
------ ------------------------------ -- ---- -- ---- -- -- ---- --
----041112 BROOKS FARNHAM RANCH           00 HPCP HI 1970 09 01
0100  00000
041112 BROOKS FARNHAM RANCH           00 HPCP HI 1970 09 04 0100
00000 041112 BROOKS FARNHAM RANCH           00 HPCP HI 1970 10 01
0100  00000
…
```

3. 3240 fixed:

```
HPD04111200HPCPHI19700900010010100 00000g
HPD04111200HPCPHI19700900010012500 00000
HPD04111200HPCPHI19700900040011900 00005
HPD04111200HPCPHI19700900040012500 00005
HPD04111200HPCPHI19701000010010100 00000g
```

4. 3240 variable:

```
HPD04111200HPCPHI19701000010020100 00000g 2500 00000
HPD04111200HPCPHI19701000180080400 00003  0500 00010  0600 00012
HPD04111200HPCPHI19701000200030100 99999[ 1200 99999] 2500 00000I
…
```

5. 3260 Comma Delimited with Station Name and html headings:

```
<META http-equiv=Content-Type content="text/html; charset=utf-
8"></HEAD>
<BODY><PRE>COOPID,STATION NAME
,CD,ELEM,UN,YEAR,MO,DA,TIME, VALUE,F,F
------,------------------------------,--,----,--,----,--,--,----,--
----,-,-
040897,BLUE CANYON                   ,00,QPCP,HI,1985,08,01,0015,
00000,g,
040897,BLUE CANYON                   ,00,QPCP,HI,1985,08,01,2500,
00000, ,
040897,BLUE CANYON                   ,00,QPCP,HI,1985,08,17,1500,
00003, ,
040897,BLUE CANYON                   ,00,QPCP,HI,1985,08,17,1600,
00008, ,
```

6. 3260 Variable with html headings:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD>
<META http-equiv=Content-Type content="text/html; charset=utf-
8"></HEAD>
<BODY><PRE>15M04089700QPCPHI19850800010020015 00000g 2500 00000
15M04089700QPCPHI19850800170041500 00003  1600 00008  1700 00002
2500 00013
15M04089700QPCPHI19850800290051700 00004  1800 00011  1900 00003
2000 00002  2500 00020
15M04089700QPCPHI19850800300021200 00001  2500 00001
15M04089700QPCPHI19850900010020015 00000g 2500 00000
```

7. 3260 Fixed with html headings:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD>
<META http-equiv=Content-Type content="text/html; charset=utf-
8"></HEAD>
<BODY><PRE>15M04089700QPCPHI19850800010020015 00000g
15M04089700QPCPHI19850800010022500 00000
15M04089700QPCPHI19850800170041500 00003
15M04089700QPCPHI19850800170041600 00008
15M04089700QPCPHI19850800170041700 00002
```

8. 3200 Delimited:

```
COOPID,WBANID,Prelim,year,month,day,Tmax,Tmin,Tobs,Tmean,Cdd,Hdd,Pr
cp,Flag
042294,99999, ,1989,12,29,56,37,46,47,0,18,0, ,0, ,0
042294,99999, ,1989,12,30,65,33,37,49,0,16,0, ,0, ,0
042294,99999, ,1989,12,31,52,30,32,41,0,24,0, ,0,
,0,42.8,53.0,32.6,65,25,0,636,0.03,0
```

9. 3200 Fixed:

```
DLY29002201SNWD I19480399990310118 00005 10218 00005 10318 00012
10418 00016 10518 00016
DLY29002201DYSWNA19480499990310199-99999M 0218 01100 10318 01100
10499-99999M 0518 01100   DLY29002201PRCPHI19480499990310118 00000
10218 00000 10318 00000 10418 00000 10518 00000
```

10. 3210 Fixed:

```
DLY00003017F2MNMD200401A1990310124 21018 00224 04020 00324 13023
00424 31018 00524 DLY00003017F5SCMD200401A1990310124 21023 00224
05022 00324 13025 00424 31022 00524
DLY00003017FMTMHR200401A1990310124 01810 00224 01258 00324 01341
00424 01551 00524
```

11. 3210 Fixed:

```
MLY04111202CLDD D19859999990060100 00000  0200 00003  0300 00000
0400 00032B MLY04111202DP01NA19859999990060100 00002  0200 00002
0300 00008  0400 00000  MLY04111202DP05NA19859999990060100 00000
0200 00001  0300 00001  0400 00000
```

12. 3220 Delimited:

```
COOPID,WBNID,CD,ELEM,UN,YEAR,A,S,MO,DA,  JAN ,F,F,MO,DA,  FEB
,F,F,MO,DA,  MAR
------,-----,--,----,--,----,-,-,--,--,------,-,-,--,--,------,-,-
,--,--,------041112,99999,02,CLDD, D,1985,9,9,01,00, 00000, ,
,02,00, 00003, , ,03,00, 00000
041112,99999,02,DP01,NA,1985,9,9,01,00, 00002, , ,02,00, 00002, ,
,03,00, 00008
```

13. LCD Delimited:

```
Month:, 12/2004
Station Name:, "LOS ANGELES INTERNATIONAL AIRPORT" Call Sign:, LAX
Day,Time,StationType,Maint Indic,SkyConditions,Visibility,Weather
Type,Dry Bulb 01,0050,AO2 ,-,CLR ,10SM   ,-,46   ,   7.8,40   ,
4.3,31   ,   -.6, 56 , 3   ,VRB,-, 01,0150,AO2 ,-,CLR ,10SM   ,-,46
,   7.8,38   ,   3.5,27   ,  -2.8, 47 , 0   ,000,-, 01,0250,AO2 ,-
,CLR ,10SM   ,-,45   ,   7.2,36   ,   2.3,22   ,  -5.6, 40 , 5
,010,-,
```

14. LCD Delimited:

```
Month:,09/2002
Station Name:, ASHEVILLE REGIONAL AIRPORT (AVL)
DT,Hr1, Hr2, Hr3, Hr4, Hr5, Hr6, Hr7, Hr8, Hr9, Hr10, Hr11, Hr12,
Hr13, Hr14, Hr15,
1,----,----,T   ,----,----,----,----,----,----,----,----,----,----
,----,----,----,----
15,.05,.02,.03,.06,.01,.02,.02,T   ,.01,T
,.02,.09,.06,.02,.04,.05,.01,T   ,.01,T
---,----,----,----,----,----,----,----,----,----,----,----,----,---
-,----,----,----
```

15. GSOD Delimited:

```
STN---  YEARMODA    TEMP       DEWP      SLP        STP       VISIB
WDSP     MXSPD   010010  20000501    33.2  8    29.9  8  1013.4  8
1012.3  8    8.6  6    4.5  8   13.0  010010  20000502    29.4  8
26.5  8  1015.0  8  1013.8  8    4.1  6    5.7  8   12.0  010010
20000503    31.3  8    28.7  8  1015.0  8  1013.8  8    6.0  6
2.2  8    5.1  010010  20000504    32.3  8    29.8  8  1000.4  8
999.2  8    2.4  5    8.3  8   13.0
```

16. 3505 Delimited:

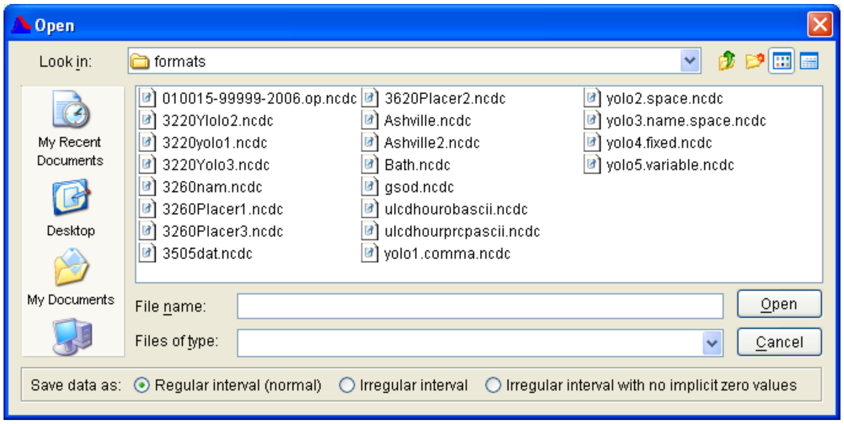
```
   AWS  WBAN YR--MODAHRMN DIR SPD GUS CLG SKC L M H  VSB WW WW WW W
TEMP DEWP    SLP   722280 13876 200301010053 140  11 ***  49 OVC *
* * 10.1 00 ** ** *   59   54 1003.8 722280 13876 200301010153 140
13 ***  25 OVC * * *  2.5 63 10 ** *   57   54 1003.7 722280 13876
200301010253 170  14  21  14 OVC * * *  3.0 63 10 ** *   57   55
1003.7
```

5.6.3 Importing

You use the NCDC plug-in after you have retrieved your NCDC data files. The file NCDC.jar must be located in the **Plugins** directory for HEC-DSSVue. You can import NCDC data by either dragging NCDC files with an extension of ".*ncdc*" from Windows Explorer and drop them into the main screen of HEC-DSSVue with an opened HEC-DSS file or by selecting **NCDC** from the **Import** menu under the **Data Entry** men

To import NCDC data files into HEC-DSS:

1. Open your DSS file, from the **Data Entry** menu, point to **Import**, click **NCDC**. An **Open** browser will open (Figure 5.36).



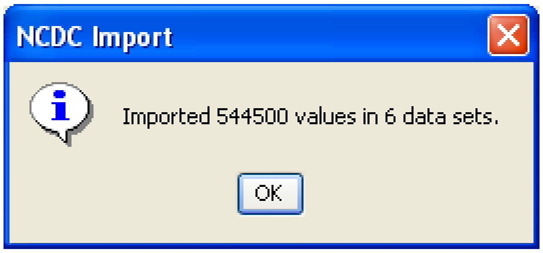
Look in: formats

Save data as: ● Regular interval (normal) ○ Irregular interval ○ Irregular interval with no implicit zero values

Figure 5.36 Open Browser - Import NCDC Files

2. Navigate to the directory with your data file, select the file and then click **Open.**
3. The plug-in will read the file, parse the data and store it into the opened DSS file. This process may take several seconds to complete, depending on the amount of data and speed of your machine. After the import is complete, a confirmation window will appear, see Figure 5.37 (page 5-33).

Or, alternatively you can import NCDC files if they have the extension ".*ncdc*" by:



Imported 544500 values in 6 data sets.

Figure 5.37 Import NCDC File(s) Confirmation

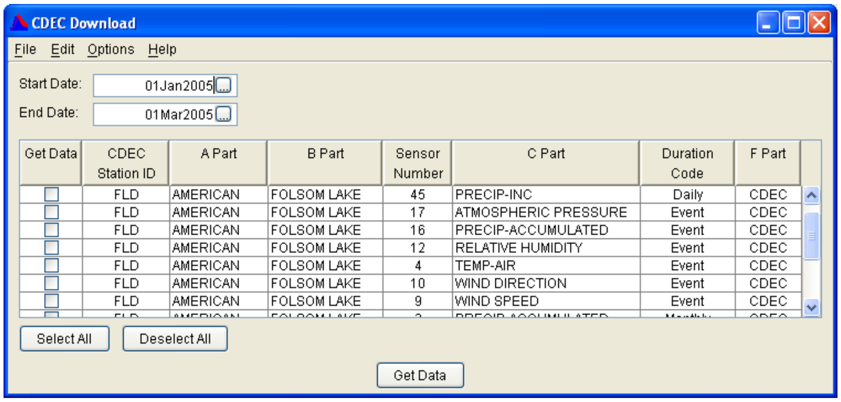
1. From Windows Explorer, select the "*ncdc*" files that you wish to import.
2. "Drag" the files onto the main HEC-DSSVue screen (with your HEC-DSS file that you want to import to opened).
3. The data is imported and a message dialog will show how many values were stored.

5.7 Retrieving and Importing CDEC Data

You can optionally retrieve and import time-series data from the CDEC (California Data Exchange Center) database via the web using the **CDEC** plug-in tool. It can retrieve a variety of hydro-meteorological and environmental data from CDEC for stations throughout the state of California. You must enter the station information (station ID, sensor, duration code, and time window) manually into the plug-in table. The plug-in has no capability to identify which stations and what data is available from CDEC. Station ID and data availability can be obtained from the CDEC web site at: http://cdec.water.ca.gov.

To use the **CDEC** plug-in you must be connected to the internet and the file *CDEC.jar* must be located in the **Plugins** directory for HEC-DSSVue.

1. Open your DSS file, from the **Data Entry** menu, point to **Import**, click **CDEC (Web)**, this will open the **CDEC Download** dialog box (Figure 5.38) with an empty station identifier table.



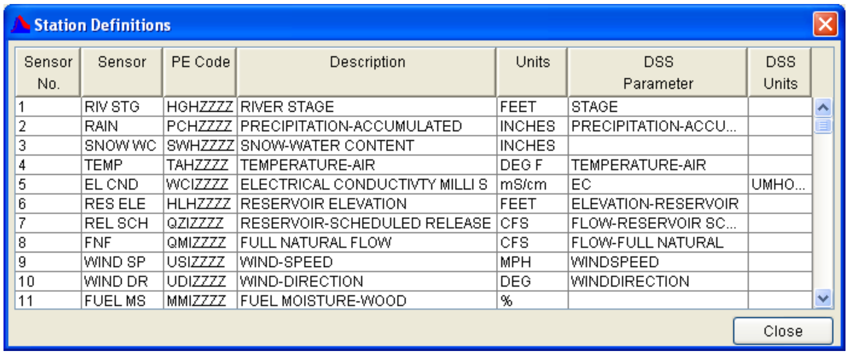
Start Date: 01Jan2005

End Date: 01Mar2005

Get Data	CDEC Station ID	A Part	B Part	Sensor Number	C Part	Duration Code	F Part
□	FLD	AMERICAN	FOLSOM LAKE	45	PRECIP-INC	Daily	CDEC
□	FLD	AMERICAN	FOLSOM LAKE	17	ATMOSPHERIC PRESSURE	Event	CDEC
□	FLD	AMERICAN	FOLSOM LAKE	16	PRECIP-ACCUMULATED	Event	CDEC
□	FLD	AMERICAN	FOLSOM LAKE	12	RELATIVE HUMIDITY	Event	CDEC
□	FLD	AMERICAN	FOLSOM LAKE	4	TEMP-AIR	Event	CDEC
□	FLD	AMERICAN	FOLSOM LAKE	10	WIND DIRECTION	Event	CDEC
□	FLD	AMERICAN	FOLSOM LAKE	9	WIND SPEED	Event	CDEC

Figure 5.38 CDEC Download Dialog Box

2. You can either open a previously saved table or manually enter the station information for the stations that you want. The CDEC plug-in cannot populate the table; you must manually enter the station information.
3. To open a previously saved table, select the **File** menu and then **Open.** Browse to the directory and file that you want to access and then press the **Open** button. Optionally, you can select a file from the most recently used file list at the bottom of the **File** menu. To save a table, select the **File** menu and then **Save**. Browse to the directory where you want to save your table and then enter the file name and press the **Save** button. Tables are in text format and are saved with an extension of ".*cdec*".
4. To manually enter station information, select the CDEC Station ID cell on a blank row. Enter the ID and then fill in the basin or river name, location name and sensor number. For each sensor, a default C part will be provided; you can change the C part, if desired. Enter the duration code, which may be "Monthly", "Hourly", "Daily" or "Event". The duration code must match the data set offered by CDEC for that location and sensor; it will not retrieve data that does not have that duration code. (You need to access the CDEC web site at http://cdec.water.ca.gov to determine what data sets are offered with what duration codes.) A default F part of "CDEC" will be set but you can change that, if desired. If you need to add additional rows, select the **Edit** menu and then **Insert Rows**. Fill in the identifiers for any remaining stations that you want. Once you have completed your table, save it using **Save** from the **File** menu.
5. You can sort the station table by pressing on the column header of the column that you want to sort by. A second press of the column header will perform an inverse sort.
6. You can get a list of all sensor numbers, PE Codes and other information that CDEC has available by selecting **Stations Definitions** from the **Options** menu, as shown in Figure 5.39. These are the codes that CDEC uses, not necessarily what is available for each station.



Sensor No.	Sensor	PE Code	Description	Units	DSS Parameter	DSS Units
1	RIV STG	HGHZZZZ	RIVER STAGE	FEET	STAGE	
2	RAIN	PCHZZZZ	PRECIPITATION-ACCUMULATED	INCHES	PRECIPITATION-ACCU...	
3	SNOW WC	SWHZZZZ	SNOW-WATER CONTENT	INCHES		
4	TEMP	TAHZZZZ	TEMPERATURE-AIR	DEG F	TEMPERATURE-AIR	
5	EL CND	WCIZZZZ	ELECTRICAL CONDUCTIVTY MILLI S	mS/cm	EC	UMHO...
6	RES ELE	HLHZZZZ	RESERVOIR ELEVATION	FEET	ELEVATION-RESERVOIR	
7	REL SCH	QZIZZZZ	RESERVOIR-SCHEDULED RELEASE	CFS	FLOW-RESERVOIR SC...	
8	FNF	QMIZZZZ	FULL NATURAL FLOW	CFS	FLOW-FULL NATURAL	
9	WIND SP	USIZZZZ	WIND-SPEED	MPH	WINDSPEED	
10	WIND DR	UDIZZZZ	WIND-DIRECTION	DEG	WINDDIRECTION	
11	FUEL MS	MMIZZZZ	FUEL MOISTURE-WOOD	%		

Figure 5.39 List of Station Definitions

7. To retrieve data from the CDEC web site and import into HEC-DSS, open or fill-in the station table as describe above, and then enter the start date and end date for the time frame that you want data for. (The date boxes cannot be empty like they can for the USGS plug-in.) The date should be entered in the form of DDMMMYYYY (e.g., 03FEB2005). Selecting the small box in the date field will provide a calendar tool to aid in setting the date. Even though you may specify dates, only data that is available within those dates from the web site can be retrieved. You should check the CDEC web site (at http://cdec.water.ca.gov) to verify what data is available for the stations that you are interested in.
8. After the data type and time span have been set, select the stations from the table that you want data for by checking the box in the **Get Data** column for those stations. If you want to retrieve data for all stations in your table, press the **Select All** button. The data retrieval process will begin when you press the **Get Data** button. This operation will take some time, depending on your connection speed to the internet and how much data you have requested. A **Retrieve Progress** dialog box will display the progress of the process. After data has been retrieved, the main catalog screen in HEC-DSSVue will be automatically updated.
9. Monthly, daily and hourly data will be stored with an E part of "1MON", "1DAY", or "1HOUR", respectively. Event data will be stored in an irregular-interval format with an E part of "IR-MONTH". You may want to use the interval modification functions available from the **Time Functions** tab in the **Math Functions** screen to change the data to a desired interval.
10. The CDEC plug-in currently has limited error detection capabilities. If you think there are problems, you should check the HEC-DSSVue log screen for messages. The log screen is usually minimized in your task bar.

    The CDEC plug-in will indicate when data cannot be retrieved, but it may not give the reason why. You should check the CDEC web page with a browser to ensure that what you are requesting is available. The web address is: http://cdec.water.ca.gov.

5.8 Importing and Exporting DSSUTL Format

You can import HEC-DSS data from the legacy DSSUTL program that was created using the "Write Data" **WD** command and export data from HEC-DSSVue to the DSSUTL "Write Data" format so that it can be read into DSSUTL with the "Read Data" **RD** command. Refer to the DSSUTL documentation for more information. The function is supplied as a plug-in and the file *Dssutl.jar* must be located in the **Plugins** directory for HEC-DSSVue.

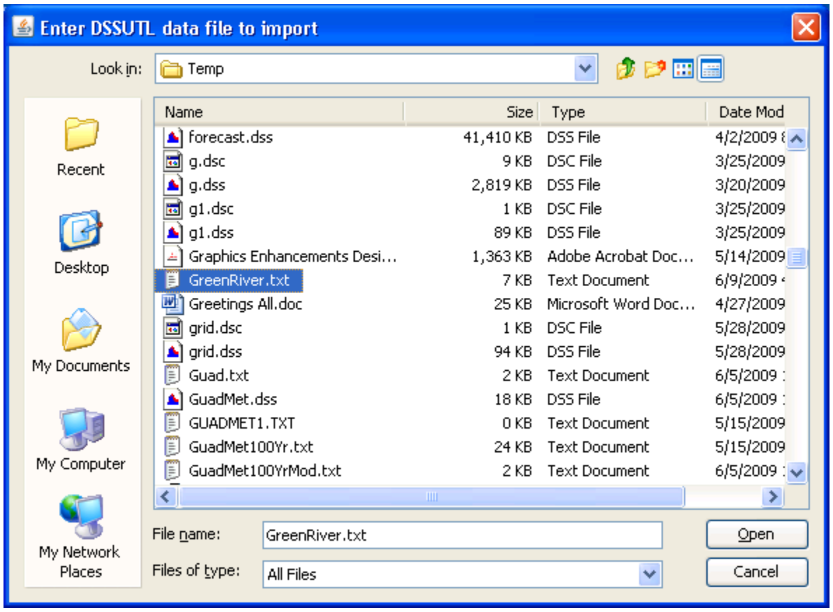
An example of the data format from DSSUTL is:

```
/MY BASIN/RIVERSIDE/FLOW/01MAY1990/1HOUR/OBS/
RTS  Ver: 44   Prog:GOES    LW:02JUN90  10:09:33   Tag:RIV-FLOW   Prec:0
Start: 01MAY1990 at 0100 hours;   End: 04MAY1990 at 1200 hours;  Number:  108
Units: CFS        Type: INST-VAL
 13190.  13820.  14630.  15080.  15260.  15530.  15620.  15710.  15800.
 15890.  15980.  14540.  14990.  15530.  15170.  14630.  14360.  14000.
 13010.  12200.  11660.  11570.  11930.  11930.  11930.  12290.  12290.
 12380.  12290.  12740.  13010.  12830.  12830.  13190.  14630.  14900.
 13910.  13010.  11930.  10740.  10050.   9750.   9620.   9685.   9685.
 10200.  10660.  10660.  10275.   9975.   9825.   9900.   9975.  10200.
 10275.  10740.  11570.  11390.  10580.  10125.  10425.  10500.   9555.
  8150.   9036.   9230.   9295.   9230.   9295.   9230.   9100.   9100.
  9100.   8908.   9036.   9100.   9165.   9295.   9425.  10125.  10275.
  9825.   9555.   9555.   9750.   9685.   9685.   9620.   9555.   9555.
  9490.   9555.   9620.   9750.   9825.   9825.  10125.  10275.  10740.
 11220.  11660.  12110.  12470.  12650.  12560.  12560.  12470.  12470.
 END FILE
```

5.8.1 Importing DSSUTL Data

To import data from the DSSUTL format:

1. Open your DSS file, from the Data Entry menu, point to Import, click **Dssutl Write Data File**, the Enter DSSUTL data file to import browser will open (Figure 5.40).



Enter DSSUTL data file to import

Look in: Temp

Name	Size	Type	Date Mod
forecast.dss	41,410 KB	DSS File	4/2/2009
g.dsc	9 KB	DSC File	3/25/2009
g.dss	2,819 KB	DSS File	3/20/2009
g1.dsc	1 KB	DSC File	3/25/2009
g1.dss	89 KB	DSS File	3/25/2009
Graphics Enhancements Desi...	1,363 KB	Adobe Acrobat Doc...	5/14/2009
GreenRiver.txt	7 KB	Text Document	6/9/2009
Greetings All.doc	25 KB	Microsoft Word Doc...	4/27/2009
grid.dsc	1 KB	DSC File	5/28/2009
grid.dss	94 KB	DSS File	5/28/2009
Guad.txt	2 KB	Text Document	6/5/2009
GuadMet.dss	18 KB	DSS File	6/5/2009
GUADMET1.TXT	0 KB	Text Document	5/15/2009
GuadMet100Yr.txt	24 KB	Text Document	5/15/2009
GuadMet100YrMod.txt	2 KB	Text Document	6/5/2009

File name: GreenRiver.txt

Files of type: All Files

Figure 5.40 Selecting a DSSUTL Write Data File to Import

2. Select your file to import. Data must be produced by the DSSUTL WD command.

3. Navigate to the directory with your data file, select the file and then click **Open.**
4. The plug-in will read the file, parse the data and store it into the opened DSS file. This process may take several seconds to complete, depending on the amount of data and speed of your machine.

5.8.2 Exporting Data to DSSUTL

To export data from the DSSUTL format:

1. Select the pathnames for the data sets that you want to export, then from the **Data Entry** menu, point to **Export** , click **Dssutl Write Data File**.
2. A file selection browser will be displayed. Navigate to the directory you wish to export the data to, type the name of file and Click **Save**. All data will be saved in one file.

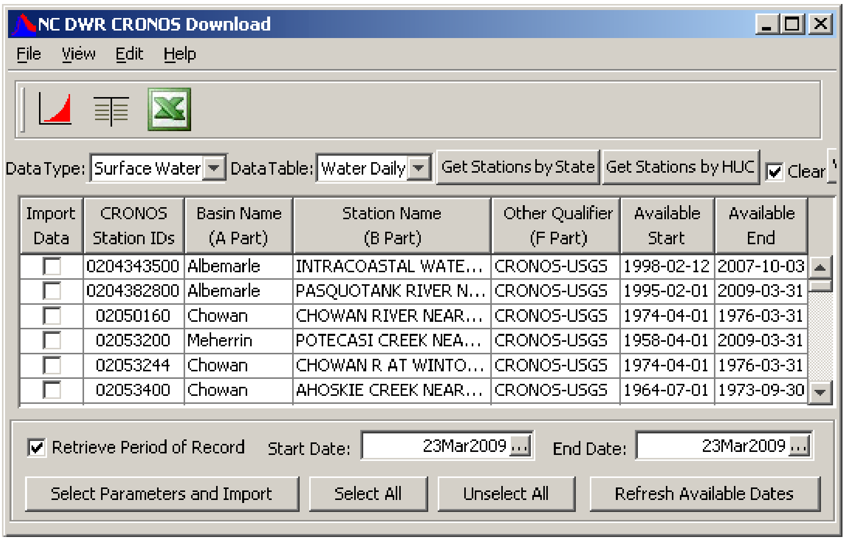
5.9 Importing North Carolina DWR CRONOS Data

The North Carolina Department of Water Resources developed a plug-in to import data from the CRONOS database into HEC-DSS. This plug-in is supported by North Carolina DWR. For questions, please contact them directly. The web site is http://www.ncwater.org/wrisars/dss.

The CRONOS Plug-in, developed by NC DWR, can search and retrieve the water and weather data for North Carolina and the surrounding states from NC Climate Retrieval and Observations Network of the Southeast Database (CRONOS) and store the data into a HEC-DSS file. The map and table search engine in Water Resources Information, Storage, Analysis, and Retrieval System (**WRISARS**) can be used with the CRONOS Plug-in to enhance the search capability. An example of the **NC DWR CRONOS Download** dialog box is shown in Figure 5.41 (page 5-38).

To import data from CRONOS:

1. In HEC-DSSVue, create a new DSS file or open an existing one.
2. In the HEC-DSSVue main window, from the **Data Entry** menu, point to **Import**, click **NC DWR CRONOS**, as shown in Figure 5.42 (page 5-38) the **NC DWR CRONOS Download** dialog box will Open (see Figure 5.41, page 5-38).

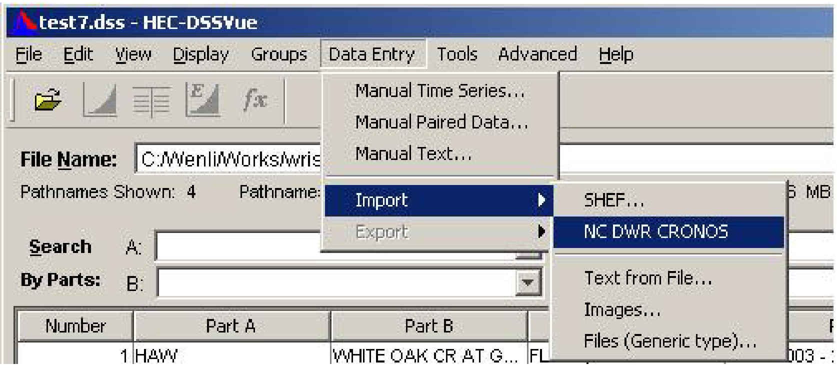


Data Type: Surface Water Data Table: Water Daily

Import Data	CRONOS Station IDs	Basin Name (A Part)	Station Name (B Part)	Other Qualifier (F Part)	Available Start	Available End
□	0204343500	Albemarle	INTRACOASTAL WATE...	CRONOS-USGS	1998-02-12	2007-10-03
□	0204382800	Albemarle	PASQUOTANK RIVER N...	CRONOS-USGS	1995-02-01	2009-03-31
□	02050160	Chowan	CHOWAN RIVER NEAR...	CRONOS-USGS	1974-04-01	1976-03-31
□	02053200	Meherrin	POTECASI CREEK NEA...	CRONOS-USGS	1958-04-01	2009-03-31
□	02053244	Chowan	CHOWAN R AT WINTO...	CRONOS-USGS	1974-04-01	1976-03-31
□	02053400	Chowan	AHOSKIE CREEK NEAR...	CRONOS-USGS	1964-07-01	1973-09-30

☑ Retrieve Period of Record Start Date: 23Mar2009 End Date: 23Mar2009

Figure 5.41 NC DWR CRONOS Download Dialog Box



File Name: C:/Wenli/Works/wris

Pathnames Shown: 4 Pathname:

Number	Part A	Part B	
1	HAW	WHITE OAK CR AT G...	FL

Figure 5.42 Importing ND DWR CRONOS Data

3. From the **NC DWR CRONOS Download** dialog box (Figure 5.41) from the **Data Type** list (Figure 5.43), select a data type for the data to be imported.

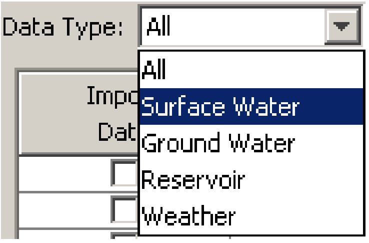


Figure 5.43 Data Type Selection

4. Set the format for the data that is to be imported, from the **NC DWR CRONOS Download** dialog box (Figure 5.41), from the t a **Data Table** list (see Figure 5.44, page 5-39), select the data table style.

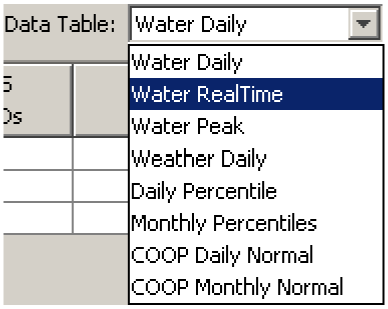
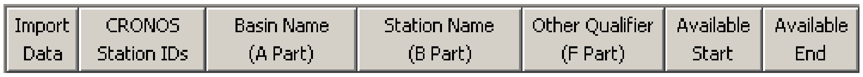


Figure 5.44 Data Table Selection

5. From the **NC DWR CRONOS Download** dialog box (see Figure 5.41, page 5-38), click **Get Stations by State**, then select a state from the **Select State** list, or click **Get Stations by HUC**, and then select a HUC. Click **OK** to search the stations and you will see the selected stations in the station table.
6. If you select **Clear Stations**, then the station table will be cleared when you select another group of stations.
7. If you want to do a more specific search, click **Use WRISARS Search Tool**, this will lead you to the WRISARS search page.
8. You can save the station table to a *.cronos* text file by from the **File** menu, click **Save Station Table**.
9. You can open a previously saved station table from an existing *.cronos* text file, from the **File** menu, click **Open Station Table**.
10. You can open a station table directly by clicking a file name (e.g. *StationTableC.cronos*) below the **Close** section from the **File** menu.
11. Select stations in the station table.
12. You can click the column title to sort the station table by that column, e.g. click **Available End** to sort by the available end timestamps. The columns in the table are shown in Figure 5.45.

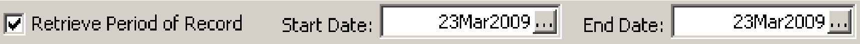


Import Data	CRONOS Station IDs	Basin Name (A Part)	Station Name (B Part)	Other Qualifier (F Part)	Available Start	Available End

Figure 5.45 Column Header Titles

13. To select or unselect all the rows in the station table, click **Select All** or **Unselect All**, respectively.
14. You can delete the checked rows from station table, from the **Edit** menu, click **Delete Checked Rows**.
15. You can also add rows in front of the current row, from the **Edit** menu, click **Insert Rows**.
16. If you want to use the Station IDs for B parts, from the **Edit** menu, click **Set B Part to Station ID**.
17. If you want to use the station names for B parts, from the **Edit** menu, click **Set B Part to Station Name**. The stations names are used for B parts by default.
18. If you just want to import a part of the data, then set up the **Start Date** and **End Date**, otherwise select **Retrieve Period of Record**.

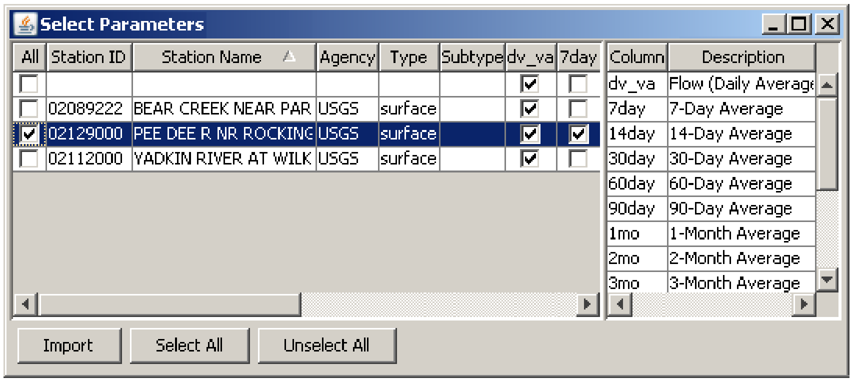
(You can click **Refresh Available Date** to get the most accurate available dates information.) See Figure 5.46 for an example of this toolbar.



☑ Retrieve Period of Record Start Date: 23Mar2009 End Date: 23Mar2009

Figure 5.46 Setting the Time Window for Data Retrieval

19. Click **Select Parameters and Import** to open the **Select Parameters** window for the selected stations. The descriptions for the parameters are shown on the right of the window. By default, all parameters are selected. Click **Unselect All** or **Select**, or you can select a few parameters manually. If you select the upper checkbox in the **All** column, then the whole row will be selected automatically. Click **Import** to import the data from the CRONOS database to the DSS file you just opened or created. An example of the **Select Parameters** window is shown in Figure 5.47.



All	Station ID	Station Name	Agency	Type	Subtype	dv_va	7day
☐						☑	☐
☐	02089222	BEAR CREEK NEAR PAR	USGS	surface		☑	☐
☑	02129000	PEE DEE R NR ROCKINC	USGS	surface		☑	☑
☐	02112000	YADKIN RIVER AT WILK	USGS	surface		☑	☐

Column	Description
dv_va	Flow (Daily Average
7day	7-Day Average
14day	14-Day Average
30day	30-Day Average
60day	60-Day Average
90day	90-Day Average
1mo	1-Month Average
2mo	2-Month Average
3mo	3-Month Average

Figure 5.47 Select Parameters Dialog Box

20. You can click **Plot**, **Tabulate**, or **Tabulate in MS Excel** open the **Select Parameters** dialog box, select some parameters, and then import the data and show the data in a plot, table, or Excel directly.
21. From the **View** menu, click **Tabulate**, **Plot**, or **Tabulate with Excel** to import the selected station-parameters and show the data in a plot, table or Excel.
22. Examine the imported or updated data in HEC-DSSVue root window, table, plot, or Excel.

CHAPTER 6

Customizing Plots

Plots are highly customizable and offer an array of information that will assist you with reviewing your data. A sample plot illustrating raw and revised data for stage and flow at a location called Beech Creek Station is shown in Figure 6.1.

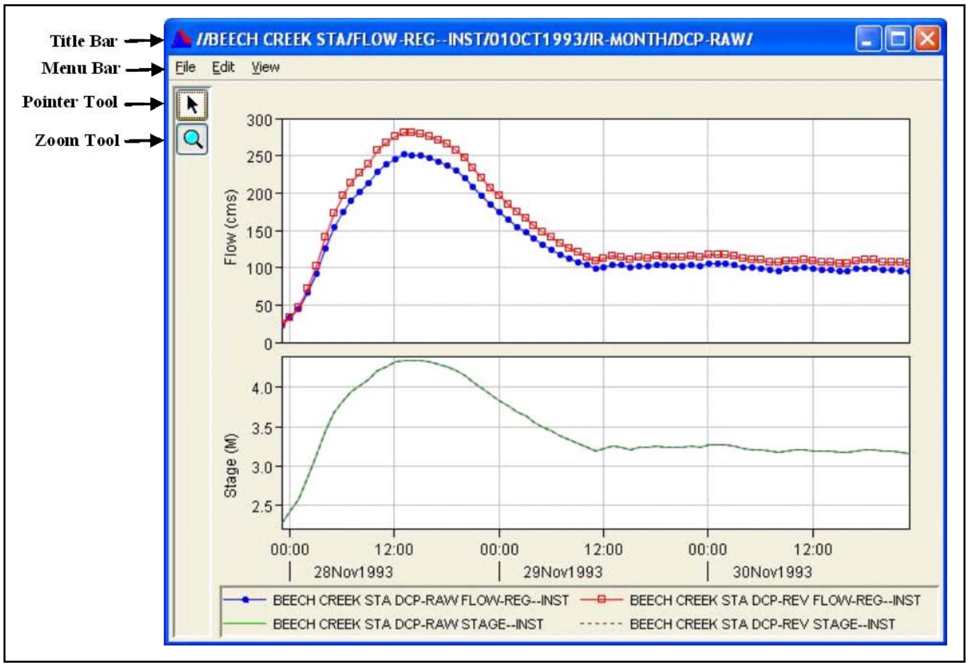

Title Bar
Menu Bar
Pointer Tool
Zoom Tool
//BEECH CREEK STA/FLOW-REG--INST/01OCT1993/IR-MONTH/DCP-RAW/
File
Edit
View
Flow (cms)
300
250
200
150
100
50
0
Stage (M)
4.0
3.5
3.0
2.5
00:00
12:00
00:00
12:00
00:00
12:00
28Nov1993
29Nov1993
30Nov1993
BEECH CREEK STA DCP-RAW FLOW-REG--INST
BEECH CREEK STA DCP-REV FLOW-REG--INST
BEECH CREEK STA DCP-RAW STAGE--INST
BEECH CREEK STA DCP-REV STAGE--INST

Figure 6.1 Plot Dialog Box

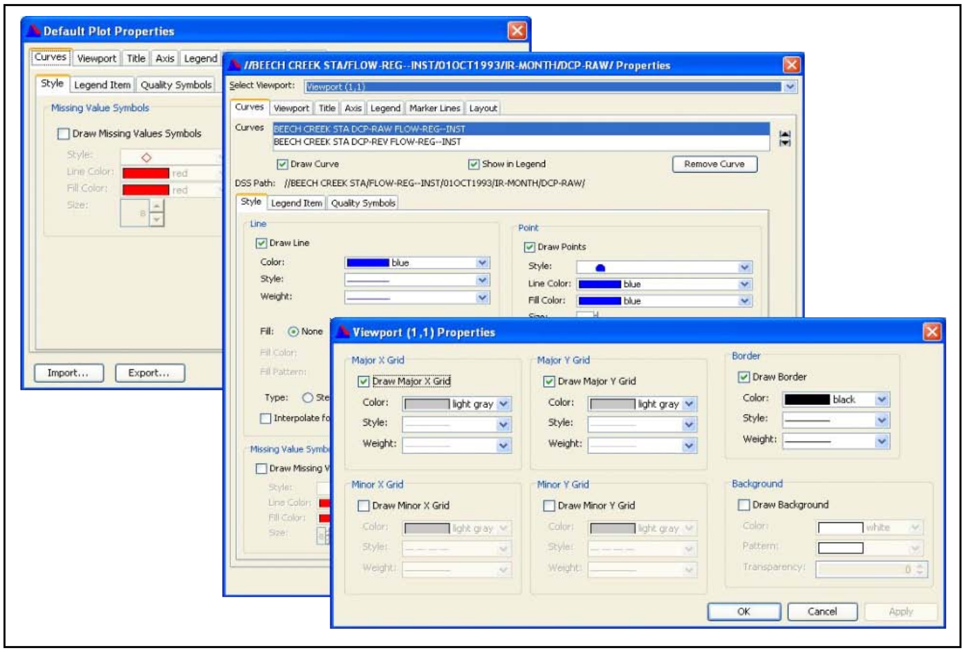
With the **Pointer Tool**, you can access shortcut menus that allow you to customize features of your plots using the plot window's editing tools. The following sections discuss these tools in detail.

The **Zoom Tool** allows you to "zoom in" and view data more precisely at smaller time intervals. To zoom in, first select the **Zoom Tool**, then hold the left mouse button down and pull down to the right forming a square around the area you wish to zoom in to. The plot will redraw, magnifying the area in the square when the mouse key is released. Right-click to zoom out or from the **View** menu, click **Zoom to all**.

If you wish to keep the plot window on top of all open windows on your desktop, from the **View** menu, click **Always on Top**, a check mark indicates this option is active.

6.1 Customizing Plots: Overview

Plot properties editors allow you to configure default properties for plots as well as customize individual plots. The features of plots that you can configure are shown in Figure 6.2.



Default Plot Properties — Missing Value Symbols: ☐ Draw Missing Values Symbols; Style: ◇; Line Color: red; Fill Color: red; Size: 8

Select Viewport: Viewport (1,1)

Curves: BEECH CREEK STA DCP-RAW FLOW-REG--INST; BEECH CREEK STA DCP-REV FLOW-REG--INST

☑ Draw Curve ☑ Show in Legend

DSS Path: //BEECH CREEK STA/FLOW-REG--INST/01OCT1993/IR-MONTH/DCP-RAW/

Line: ☑ Draw Line; Color: blue; Style: ; Weight: ; Fill: ● None; Fill Color: ; Fill Pattern: ; Type: ○ Ste; ☐ Interpolate fo

Point: ☑ Draw Points; Style: ●; Line Color: blue; Fill Color: blue; Size:

Missing Value Symb: ☐ Draw Missing V; Style: ; Line Color: ; Fill Color: ; Size:

Major X Grid: ☑ Draw Major X Grid; Color: light gray; Style: ; Weight:

Major Y Grid: ☑ Draw Major Y Grid; Color: light gray; Style: ; Weight:

Border: ☑ Draw Border; Color: black; Style: ; Weight:

Minor X Grid: ☐ Draw Minor X Grid; Color: light gray; Style: ; Weight:

Minor Y Grid: ☐ Draw Minor Y Grid; Color: light gray; Style: ; Weight:

Background: ☐ Draw Background; Color: white; Pattern: ; Transparency: 0

Figure 6.2 Configurable Features of Plots

- **Title**: Optionally, you can add a title to the plot display.
- **Panel Background Color**: You can specify the background color of the plot window. By default, the panel uses the same background color the HEC-DSSVue application uses.
- **Curve**: You can choose the line and point styles, add a label, and specify data quality symbols to curves.
- **Marker Line:** Marker lines can be added on the X and Y axis. You can customize their line styles and give them labels.
- **Viewport Spacer:** You can specify the distance between viewports.
- **Callout:** You can add descriptive callouts at specific points along a curve. Customize their fonts, borders, and backgrounds.
- **Viewport:** You can customize the borders around each viewport, their background color, pattern, and transparency, and the appearance of the gridlines.
- **Label:** Label fonts can be customized by color, style, and size. You can also give labels borders and background colors and patterns or choose to hide them.
- **Axis:** You can adjust the scale, set a minimum or maximum, major and minor tic intervals, reverse the axis, switch x and y, change label styles and text, and customize tic marks.

- **Legend:** You can add titles to the plot legend, specify the legend's location in the plot window, add left and right blocks of text or graphics, change the background color, pattern, and transparency, or remove the legend from the plot window.

6.2 Using Plot Editors

Different plot editors allow you to set specific properties for individual plots or set default plot properties for all plots. Three examples of these editors: the **Default Plot Properties** editor, the **Plot Properties** editor, and the **Viewport Properties** editor are shown in Figure 6.3.

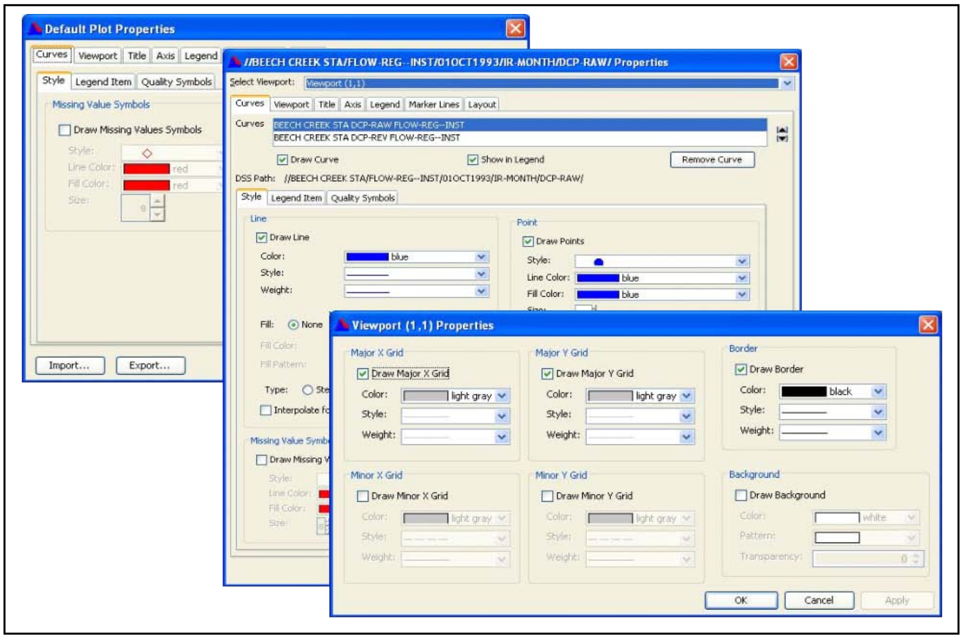


Figure 6.3 Examples of Plot Properties Editors

6.2.1 Setting Defaults vs. Customizing Individual Plots

Plot editors and tools fall into two categories in terms of function: either they allow you to specify *defaults* for *all* plots you create or they allow you to customize *individual* plots.

Across these two functional categories, the plot editors and tools allow you to edit a variety of plot properties, or they can be specialized editors that allow you to edit a single property.

To configure the *default* appearance of *all* plots, use the **Default Plot Properties** editor and **Default Line Styles** options editor. All settings you specify in these editors will apply to future plots you open. Default editors can only be accessed through the plot's **Edit** menu.

To customize *individual* plots, use the **Plot Properties** editor and the **Configure Plot** editor, accessed through plot windows using the **Edit** menu. You can display specific plot property editors by right-clicking on components displayed in the plot window.

Once you have customized an individual plot, you can export its settings as a **Template** that can be applied to other plots. See Section 6.12 for more detail on templates.

6.2.2 Accessing Editors

You can access plot property editors from shortcut menus and from the **Edit** menu of the **Plot** dialog box (Figure 6.4).

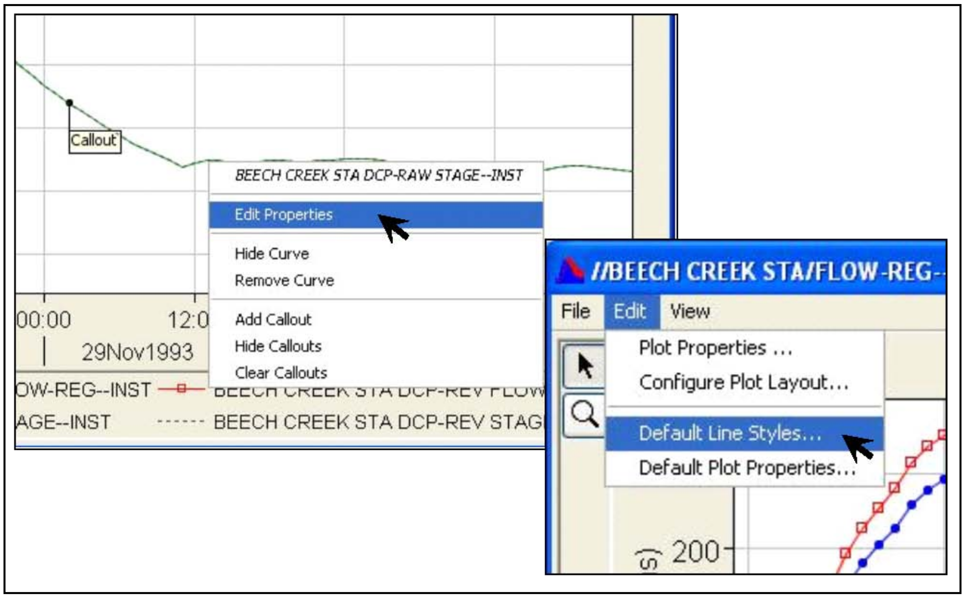


Figure 6.4 Accessing the Plot Property Editors

Use these menus according to whether you are defining properties for an individual plot or setting default properties for all plots you create.
Use shortcut menus to edit specific components of an individual plot. For example, if you want to edit axis label properties on a plot, right-click on the axis label (see Figure 6.5, page 6-5), then select **Edit Properties** from the shortcut menu.

Use the **Edit** menu of a plot window to access the **Plot Properties** editor, **Default Line Styles** options editor, **Default Plot Properties** editor, and **Configure Plot** editor. These editors allow you to edit a variety of plot properties.

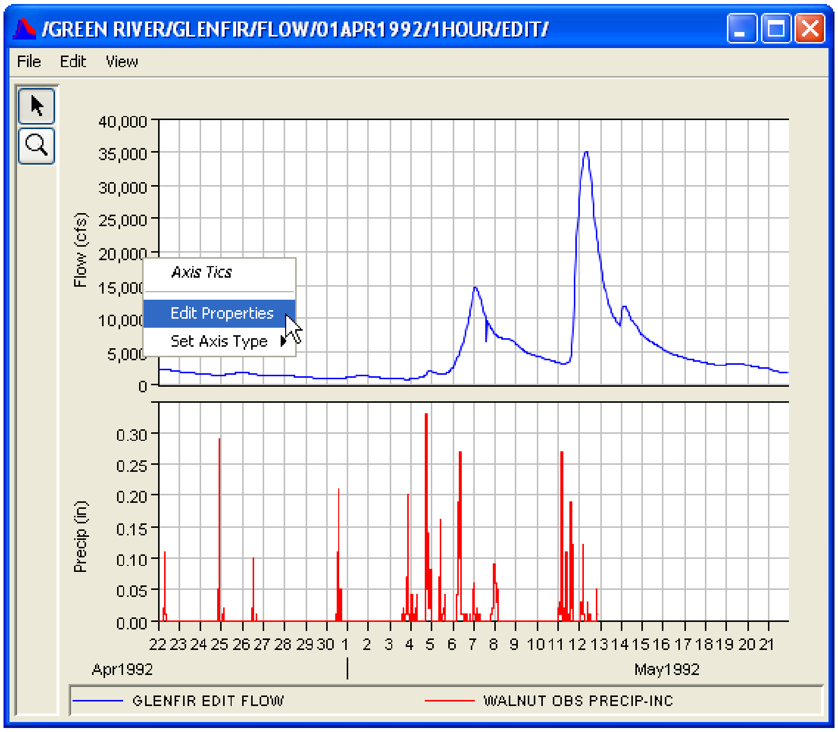


Figure 6.5 Edit Properties

6.3 Plot Editors and Tools

The following sections are an overview of the editing tools that allow you to customize plots. Later sections provide more detailed instructions on editing specific plot properties using these tools.

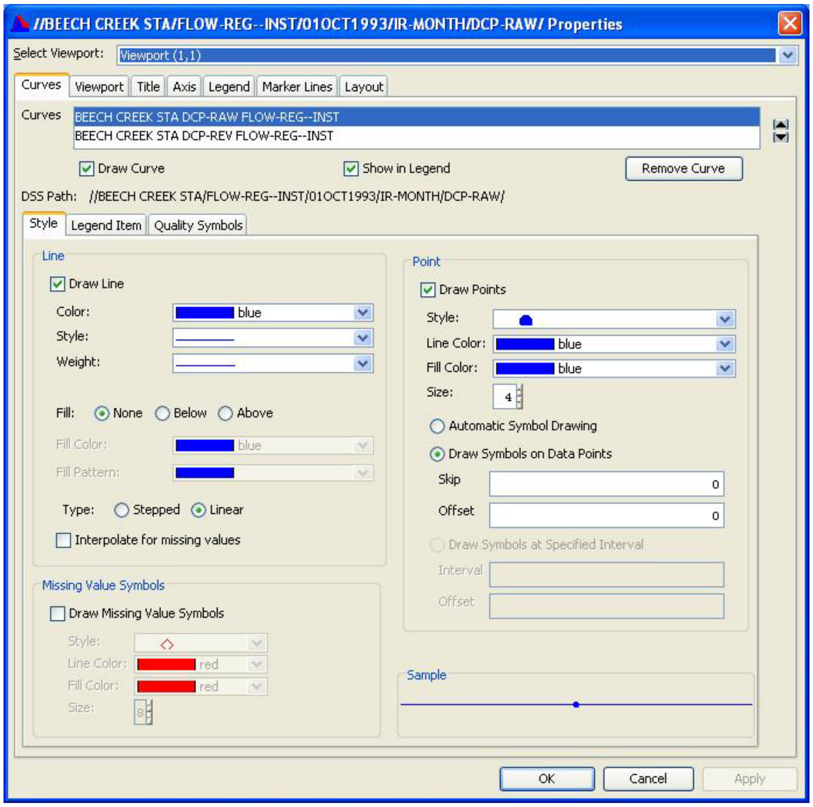
6.3.1 Plot Properties Editor

The **Plot Properties** editor (see Figure 6.6, page 6-6) is accessed from the **Edit** menu of a **Plot** dialog box by clicking **Plot Properties**. It allows you to configure multiple display properties for the individual plot, including the **Curves**, **Viewport**, **Title**, **Axis**, **Legend**, **Marker Lines**, and the **Layout** or properties of the plot window panel.

When you customize properties of a plot using the **Plot Properties** editor, your changes apply only to the individual plot displayed; unless you export the plot’s properties in a template (see Section 6.12).

6.3.2 Individual Plot Property Editors

When you want to edit a specific property of a plot without launching the **Plot Properties** editor (discussed in Section 6.3.3), you can use individual


//BEECH CREEK STA/FLOW-REG--INST/01OCT1993/IR-MONTH/DCP-RAW/ Properties

Select Viewport: Viewport (1,1)

Curves: BEECH CREEK STA DCP-RAW FLOW-REG--INST
BEECH CREEK STA DCP-REV FLOW-REG--INST

☑ Draw Curve ☑ Show in Legend

DSS Path: //BEECH CREEK STA/FLOW-REG--INST/01OCT1993/IR-MONTH/DCP-RAW/

Line
☑ Draw Line
Color: blue
Style:
Weight:
Fill: ● None ○ Below ○ Above
Fill Color: blue
Fill Pattern:
Type: ○ Stepped ● Linear
☐ Interpolate for missing values

Missing Value Symbols
☐ Draw Missing Value Symbols
Style:
Line Color: red
Fill Color: red
Size: 8

Point
☑ Draw Points
Style:
Line Color: blue
Fill Color: blue
Size: 4
○ Automatic Symbol Drawing
● Draw Symbols on Data Points
Skip 0
Offset 0
○ Draw Symbols at Specified Interval
Interval
Offset

Figure 6.6 Plot Properties Editor

plot property editors instead. These individual plot property editors correspond to the tabs of the **Plot Properties** editor.

To access an individual plot property editor, right-click on the component you want to edit and then select **Edit Properties** from the shortcut menu. For example, if you right-click inside the gridded plot area, called the *viewport*, you will see the **Viewport** shortcut menu (Figure 6.7). When you choose **Edit Properties**, the **Viewport Properties** editor will open (Figure 6.8).

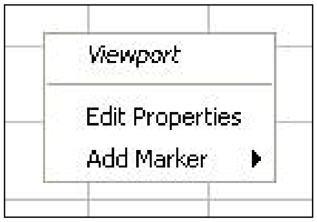
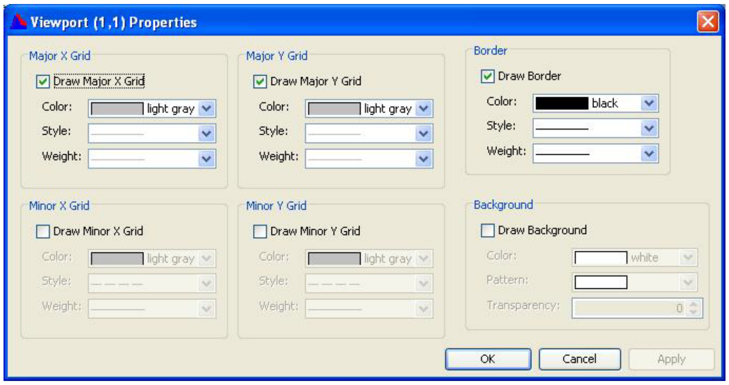


Figure 6.7 Viewport Shortcut Menu


Viewport (1,1) Properties

Major X Grid
☑ Draw Major X Grid
Color: light gray
Style:
Weight:

Major Y Grid
☑ Draw Major Y Grid
Color: light gray
Style:
Weight:

Border
☑ Draw Border
Color: black
Style:
Weight:

Minor X Grid
☐ Draw Minor X Grid
Color: light gray
Style:
Weight:

Minor Y Grid
☐ Draw Minor Y Grid
Color: light gray
Style:
Weight:

Background
☐ Draw Background
Color: white
Pattern:
Transparency: 0

Figure 6.8 Viewport Properties Editor

The **Viewport Properties** editor lets you edit only properties associated with that particular viewport, using the same options that appear for viewports in the **Plot Properties** editor.

Other individual properties editors are the Edit Title Properties, Axis Properties, Curve Properties, Label Properties, Marker Properties, and Legend Properties.

The only plot property you cannot edit using an individual property editor is the **Layout**. (See Section **Error! Reference source not found.** for more information.)

6.3.3 Configure Plot Editor

The **Configure Plot** editor (Figure 6.9) is accessed from the **Edit** menu of a plot and allows you to customize the data configuration of an individual plot. You can add and remove axes and add, remove, arrange the order of, and set the weight of viewports in the plot window.

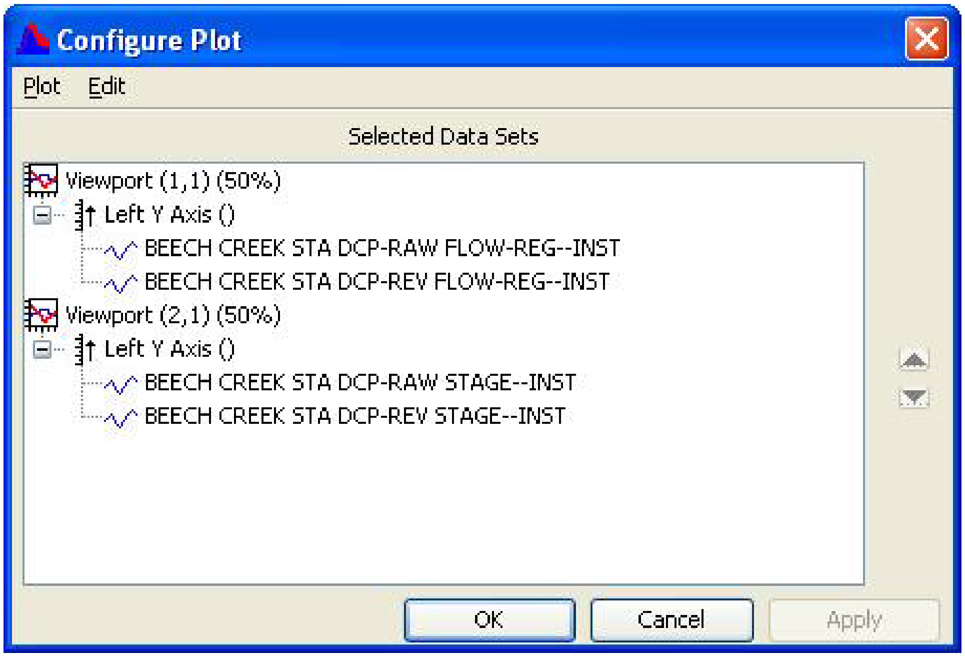

Configure Plot
Plot
Edit
Selected Data Sets
Viewport (1,1) (50%)
Left Y Axis ()
BEECH CREEK STA DCP-RAW FLOW-REG--INST
BEECH CREEK STA DCP-REV FLOW-REG--INST
Viewport (2,1) (50%)
Left Y Axis ()
BEECH CREEK STA DCP-RAW STAGE--INST
BEECH CREEK STA DCP-REV STAGE--INST
OK
Cancel
Apply

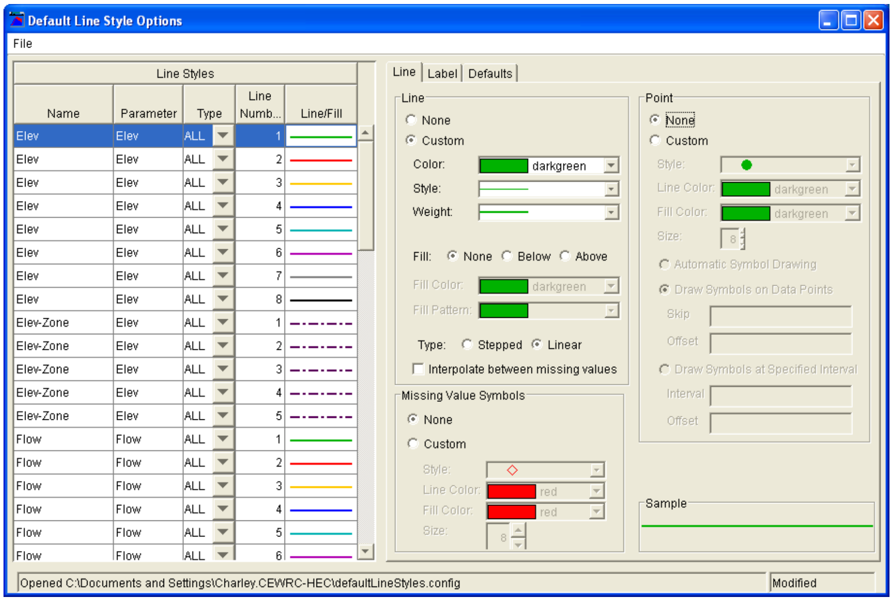
Figure 6.9 Configure Plot Editor

When you customize the layout of a plot using the **Configure Plot** editor, your changes apply only to that individual plot; unless you create a template from the plot's properties (see Section 6.12).

To access the Configure Plot editor, from the **Edit** menu, click **Configure Plot Layout**. See Section 6.11 for more information on this editor.

6.3.4 Default Line Styles Options Editor

With the **Default Line Styles** options editor (Figure 6.10), you can specify the default line and fill styles, as well as labels, used across all plots for specific parameters.



Default Line Style Options

File

Line Styles

Name	Parameter	Type	Line Numb...	Line/Fill
Elev	Elev	ALL	1	───
Elev	Elev	ALL	2	───
Elev	Elev	ALL	3	───
Elev	Elev	ALL	4	───
Elev	Elev	ALL	5	───
Elev	Elev	ALL	6	───
Elev	Elev	ALL	7	───
Elev	Elev	ALL	8	───
Elev-Zone	Elev	ALL	1	─·─·─
Elev-Zone	Elev	ALL	2	─·─·─
Elev-Zone	Elev	ALL	3	─·─·─
Elev-Zone	Elev	ALL	4	─·─·─
Elev-Zone	Elev	ALL	5	─·─·─
Flow	Flow	ALL	1	───
Flow	Flow	ALL	2	───
Flow	Flow	ALL	3	───
Flow	Flow	ALL	4	───
Flow	Flow	ALL	5	───
Flow	Flow	ALL	6	───

Line | Label | Defaults

Line
○ None
● Custom
Color: darkgreen
Style:
Weight:
Fill: ● None ○ Below ○ Above
Fill Color: darkgreen
Fill Pattern:
Type: ○ Stepped ● Linear
□ Interpolate between missing values

Missing Value Symbols
● None
○ Custom
Style: ◇
Line Color: red
Fill Color: red
Size: 8

Point
● None
○ Custom
Style: ●
Line Color: darkgreen
Fill Color: darkgreen
Size: 8
○ Automatic Symbol Drawing
● Draw Symbols on Data Points
Skip
Offset
○ Draw Symbols at Specified Interval
Interval
Offset

Sample

Opened C:\Documents and Settings\Charley.CEWRC-HEC\defaultLineStyles.config | Modified

Figure 6.10 Default Line Styles Options Editor

To access the **Default Line Styles** options editor, from the **Edit** menu, choose **Default Line Styles**.

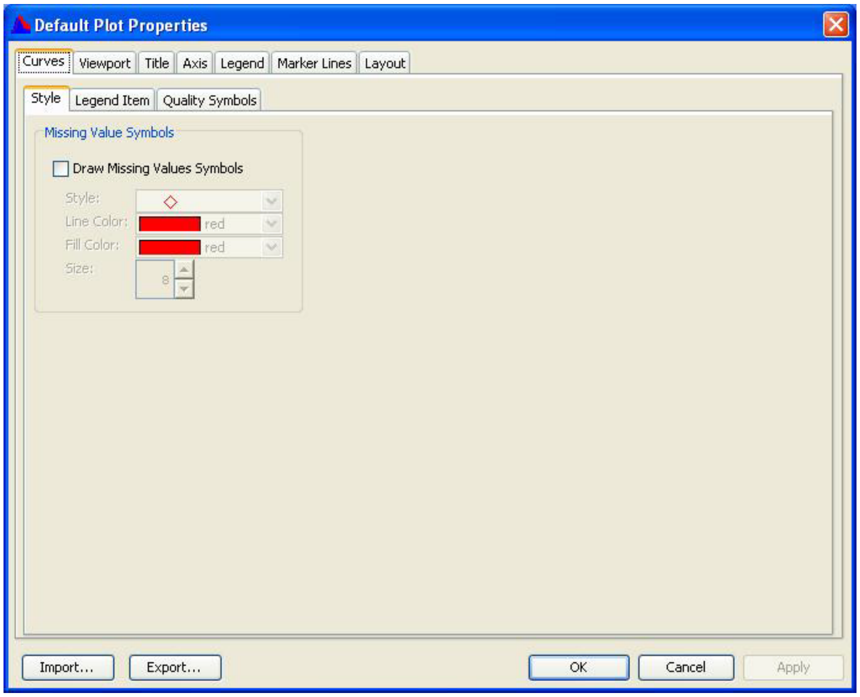
Refer to Section 6.5.2 for more information on the **Default Line Styles** options editor.

6.3.5 Default Plot Properties Editor

The **Default Plot Properties** editor (see Figure 6.11, page 6-9) allows you to configure the default display properties of all plots you create. Properties you can configure include **Curves**, **Viewport**, **Title**, **Axis**, **Legend**, **Marker Lines,** and **Layout**, which include the miscellaneous properties of the plot window panel.

When you customize plot properties using the **Default Plot Properties** editor, your changes will apply to all future plots you create. If you would like to apply them to the plot you currently have open, you will need to close and then reopen it first.

To access the **Default Plot Properties** editor, from the **Edit** menu, click **Default Plot Properties**.



□ Draw Missing Values Symbols

Style: ◇

Line Color: 🟥 red

Fill Color: 🟥 red

Size: 8

Figure 6.11 Default Plot Properties Editor

6.4 Customizing Plot Titles

You can add titles to individual plots and configure default properties for all plot titles.

To add or edit a title on an *individual* plot, you can either:

- From the **Edit** menu, click **Plot Properties**. When the **Plot Properties** editor opens, click the **Title** tab.

-Or-

- Right-click in the blank area above the plot (below the menu bar) with the **Pointer Tool** , and then click **Edit Properties** from the shortcut menu (Figure 6.12). The **Edit Title Properties** editor will open.

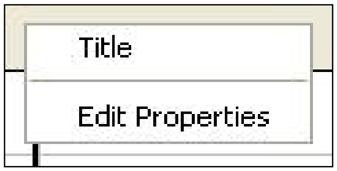
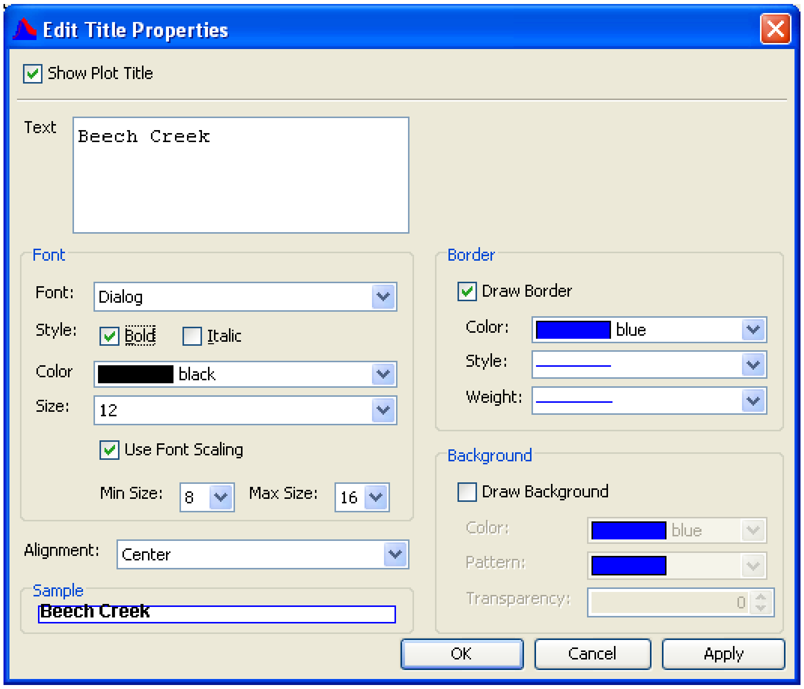


Figure 6.12 Shortcut Menu - Title Properties

To specify the appearance of titles for *all* of your plots, from the **Edit** menu, click **Default Plot Properties**, then select the **Title** tab of the **Default Plot Properties** editor.

Whether you are using the **Plot Properties** editor, the specialized **Title Properties** editor, or the **Default Plot Properties** editor, the worksheet for editing plot title properties is the same.

The **Edit Title Properties** editor is shown in Figure 6.13. This editor contains the same fields as the **Title** tab of the **Plot Properties** editor and the **Default Plot Properties** editor.



Edit Title Properties

☑ Show Plot Title

Text: Beech Creek

Font

Font: Dialog

Style: ☑ Bold ☐ Italic

Color: black

Size: 12

☑ Use Font Scaling

Min Size: 8 Max Size: 16

Alignment: Center

Sample: Beech Creek

Border

☑ Draw Border

Color: blue

Style:

Weight:

Background

☐ Draw Background

Color: blue

Pattern:

Transparency: 0

Figure 6.13 Edit Title Properties

1. To specify a title for the plot, check **Show Plot Title**.
2. In the **Text** field, type the title you want displayed in the plot window.
3. In the **Font** section of the editor, select the font type from the **Font** dropdown list. Choose the **Style**; **Bold** and/or **Italic**, **Color** of the font, and a **Size** of your text.
4. The **Font** section also includes the option to set **Use Font Scaling**. Checking this option and selecting a **Min Size** and **Max Size**, will set the minimum and maximum size your font can adjust to when resizing the plot window. If this option is not selected, the font size will remain constant regardless of the window's size.
5. Below the **Font** section on the editor is the **Alignment** setting of the title. Alignment can display the tile text **Center**, **Left**, or **Right**.
6. The **Border** section allows you to add a border around the title. You can specify the **Color**, line **Style** and **Weight**.
7. The **Background** section lets you add a background **Color** and/or **Pattern** behind your plot title. It also allows you to set the **Transparency** of the background.
8. The **Sample** box provides a preview of your plot title.

Click **Apply** to save your changes and continue adjusting the appearance of the title. Click **OK** to save your changes and close the editor.

6.5 Customizing Curves

You can customize line and point styles, add labels, and specify symbols to indicate data quality in your plots. Additionally, you can specify the parameter-based default curve styles used across all plots.

There are three different ways to edit plot curves, depending on whether you wish to customize one or more curves in an individual plot or specify defaults for all plots.

6.5.1 Customizing Curves in Individual Plots

To customize *all* curves in an *individual* plot, from the **Edit** menu, choose **Plot Properties**. Once the **Plot Properties** editor opens, select the **Curves** tab.

To customize a *specific* curve in an *individual* plot, right-click on the line or curve you wish to edit using the **Pointer Tool** ↖ , then select **Edit Properties** from the shortcut menu (Figure 6.14). The **Edit Curve Properties** editor will open.

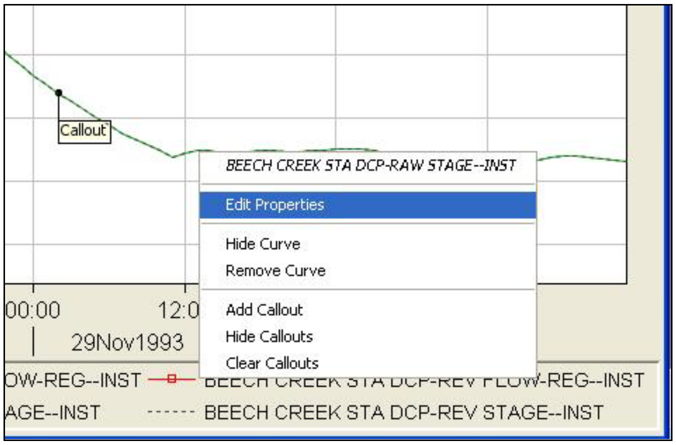


Figure 6.14 Shortcut Menu - Curve Properties

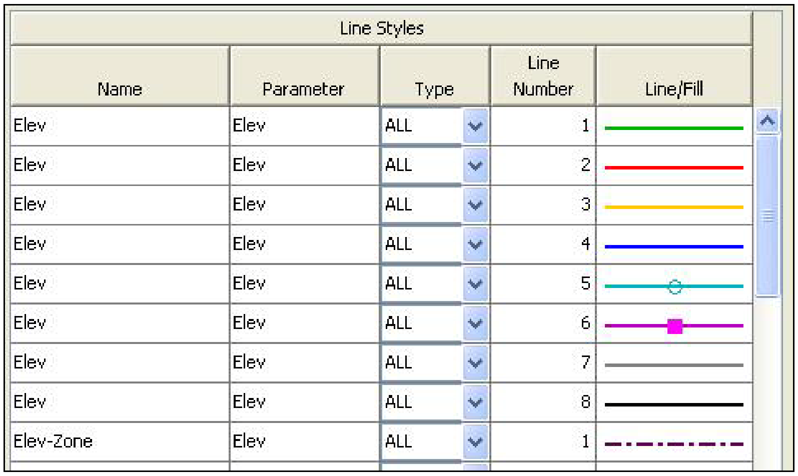
The **Curves** tab of the **Plot Properties** editor and the **Edit Curve Properties** editor are nearly identical with one exception. At the top of the **Curves** tab of the **Plot Properties** editor, there is a list of all curves contained in the plot. In contrast, when you open the **Edit Curve Properties** editor, only the selected curve’s DSS Path is displayed.

Both the **Edit Curve Properties** editor and the **Curves** tab of the **Plot Properties** editor allow you to edit **Style**, **Label, Legend Item**, and **Quality Symbols** (you can edit Quality Symbols only if the plot has quality defined data).

6.5.2 Specifying Parameter-Based Default Curve Styles

To specify parameter-based default curve styles for *all* of your plots, from the **Edit** menu, click **Default Line Styles**. The **Default Line Styles** options editor will open (see Figure 6.10, page 6-8).

The **Default Line Styles** options editor gives you a way to edit line styles from the **Line Styles** box (Figure 6.15). You can specify the default, parameter-based curve styles used for all plots.



Line Styles				
Name	Parameter	Type	Line Number	Line/Fill
Elev	Elev	ALL	1	—
Elev	Elev	ALL	2	—
Elev	Elev	ALL	3	—
Elev	Elev	ALL	4	—
Elev	Elev	ALL	5	—○—
Elev	Elev	ALL	6	—■—
Elev	Elev	ALL	7	—
Elev	Elev	ALL	8	—
Elev-Zone	Elev	ALL	1	-·-·-

Figure 6.15 Default Line Style Options Editor: Detail of Line Styles Box

The **Line Styles** table displays typical data types with default line and fill styles predefined.

You can edit all of these fields, change default line and fill styles for existing types, and add new data types to the list (currently, there is no delete option).

The **Line Styles** table columns are described below in detail:

- **Name** The **Name** corresponds to the "C" part of HEC-DSS pathnames. To edit an existing name, highlight it, and then type in the new **Name**.
- **Parameter** The **Parameter** is what distinguishes data sets and groups them together in the same viewport. For example, FLOW-IN and FLOW-OUT have different **Names** or "C" parts, but both are **Parameter** “FLOW” data sets, so they will be plotted in the same viewport.
- **Type** The **Type** describes the regular and irregular interval time series record data, either **INST-CUM** (instantaneous cumulative), **INST-VAL** (instantaneous value), **PER-AVER** (period average), **PER-CUM** (period cumulative), or **ALL**. To change the data **Type** associated with a

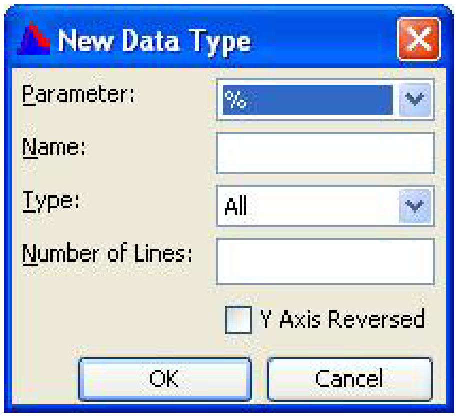
Name and **Parameter**, click the down-arrow and select from the preexisting list.

- **Line Number** The **Line Number** column indicates the number of lines/curves displayed in the viewport and what style will be associated with that data. For example, if you have three curves in a viewport each with the same Name, Parameter, and Type, the first line will be assigned the Line/Fill associated with Line Number 1; the second will use Line Number 2, the third Line Number 3, and so on. If there is only one line displayed in the viewport for that data set, it will use the Line Number 1 line style. See **Adding New Data Styles** below to add more Line Numbers to a particular data set.
- **Line/Fill** The **Line/Fill** property determines how curves associated with a particular **Name/Parameter/Type** combination will appear in all plots. To specify the **Line/Fill**, select the row, and then customize the **Line** and **Point** properties as discussed in Section 6.5.4.

6.5.3 Adding New Data Styles

To add a new data type (i.e. insert a row in the **Line Style** table):

1. From the **File** menu of the **Default Line Styles** options editor, choose **New Parameter**. The **New Data Type** dialog box will open (Figure 6.16).



Parameter: %

Name:

Type: All

Number of Lines:

□ Y Axis Reversed

Figure 6.16 New Data Type Dialog Box

2. Select a parameter from the **Parameter** list.
3. Enter a name in the **Name** box.
4. From the **Type** list, select the data type.

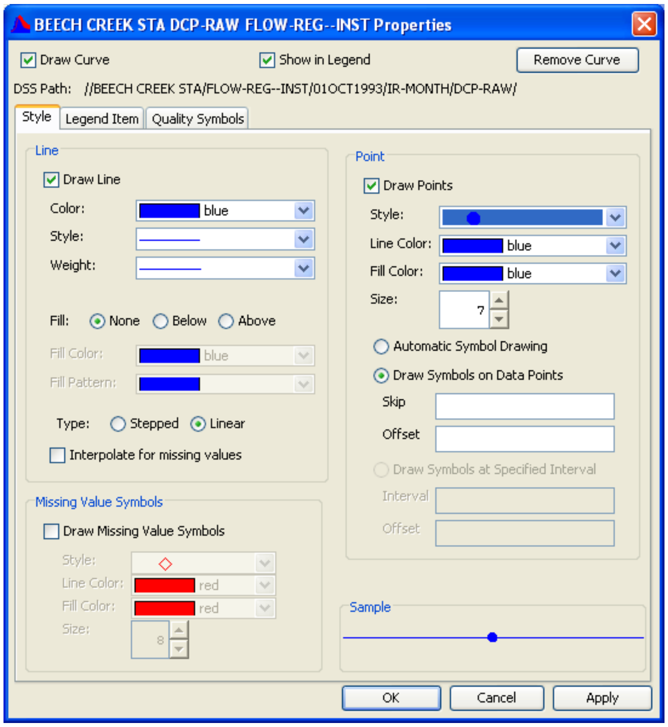
5. In the **Number of Lines** box, enter the number of curves you want to add for this new data type.
6. If you want to reverse the Y-axis, select **Y Axis Reversed**.
7. Click **OK** to save and close the dialog box.

The **Line Styles** table will display the new data type you have added, repeated as many times as you specified in the **Number of Lines** box (reflected in the **Line Number** column). You can customize the new data types as described above.

To save your changes, from the **File** menu, click **Save**. To close the **Default Line Styles** options editor, from the **File** menu, click **Close**.

6.5.4 Specifying Line and Point Styles of Curves

The curve **Line** and **Point Style** worksheet is shown in Figure 6.17. This worksheet is available from the **Curves** tabs of the **Edit Curve Properties** editor, the **Plot Properties** editor, and the **Default Line Styles** options editor (Section 6.2).



BEECH CREEK STA DCP-RAW FLOW-REG--INST Properties

☑ Draw Curve ☑ Show in Legend

DSS Path: //BEECH CREEK STA/FLOW-REG--INST/01OCT1993/IR-MONTH/DCP-RAW/

Line

☑ Draw Line

Color: blue

Style:

Weight:

Fill: ● None ○ Below ○ Above

Fill Color: blue

Fill Pattern:

Type: ○ Stepped ● Linear

☐ Interpolate for missing values

Missing Value Symbols

☐ Draw Missing Value Symbols

Style: ◇

Line Color: red

Fill Color: red

Size: 8

Point

☑ Draw Points

Style: ●

Line Color: blue

Fill Color: blue

Size: 7

○ Automatic Symbol Drawing

● Draw Symbols on Data Points

Skip

Offset

○ Draw Symbols at Specified Interval

Interval

Offset

Figure 6.17 Curve Line and Point Style Editing Interface

The **Style** tab has three main groups, **Line, Point**, and **Missing Value Symbols** which allow you to customize line and point styles, and a separate symbol that can be shown for missing values. Beneath the **Point** group, the **Sample** box provides a preview of how your line and point choices will look.

To define line styles for curves:

1. In the **Line** group, click **Draw Line**.
2. Select a **Color**, **Style**, and **Weight** for the line.
3. You can display lines with fill above or below, or without fill. A plot with line fill below is shown in Figure 6.18, whereas the same plot without line fill is shown in Figure 6.19. To set this feature use the **Fill** selections; **None**, **Below**, or **Above**. Then choose the **Fill Color** and **Fill Pattern** for the fill using the color and pattern dropdown lists located below the **Fill** section.

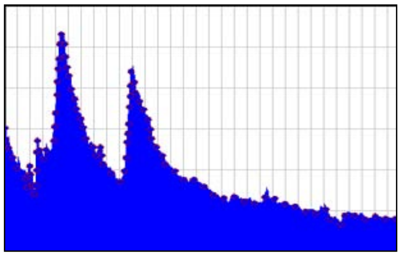


Figure 6.18 Plot with Line Fill

4. In the **Type** section, select if you want the curve drawn **Stepped,** in a stair-stepped style, or **Linear,** with a line drawn directly between each point.
5. If you would like to see missing values, check **Draw Missing Value Symbols** and select a **Style**, **Line Color**, **Fill Color**, and **Size** for the missing value points.

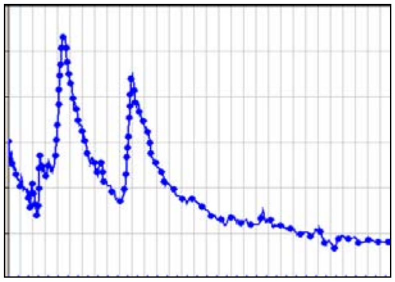


Figure 6.19 Plot without Line Fill

To define **Point** styles for curves:

1. In the **Point** group, click **Draw Points**.
2. Choose the **Style**, **Line Color**, and **Fill Color** you want. The **Line Color** is the "border" around the point symbol. The **Fill Color** is the color inside the point symbol. An example of a dark blue **Line Color** and a light blue **Fill Color** is illustrated in Figure 6.20.
3. In the **Size** box, specify the size of the point (in pixels) either by selecting a size from the list or by typing in a number from 1- 45.

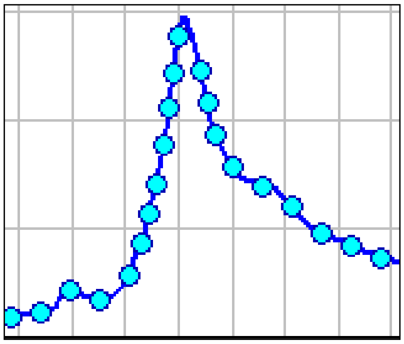


Figure 6.20 Example Line and Fill Colors

4. **Automatic Symbol Drawing** allows the program to determine the number of points to display on a curve. The program will automatically calculate how many data points can display on the curve before they start to overlap one another. Not all points will be displayed. If the points overlap at the zoom level you view the plot at, the program will determine how many points can display

separately on the curve and only display those points. As you zoom in, the plot will draw more and more points, as long as they do not touch.

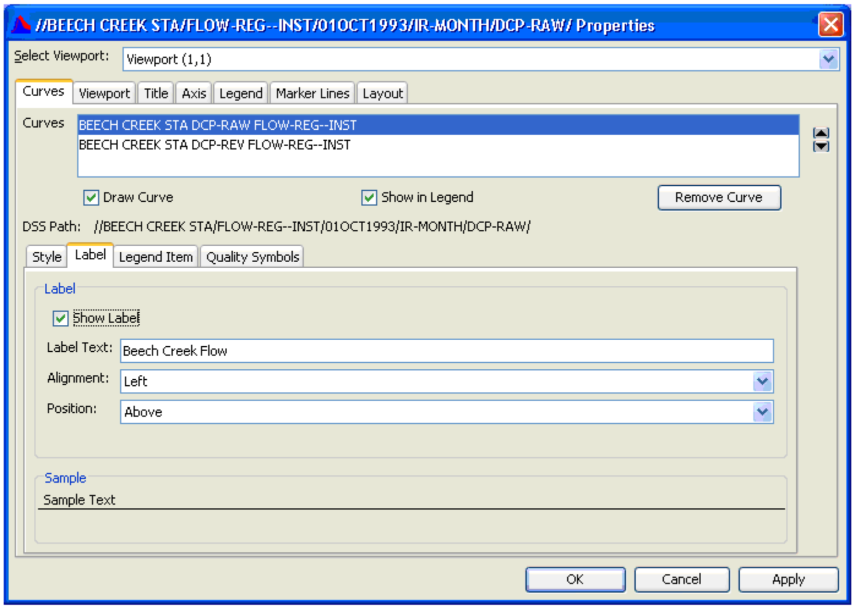
5. **Draw Symbols on Data Points** allows you to specify how to draw the points so they don't overlap. If you set the **Skip** box to one (1), then it will draw one point, skip the next, then draw the third, etc. The **Offset** box allows you to say how many points on the curve to initially skip before drawing points.

To define missing value symbols for curves:

1. In the **Missing Value Symbols** group, click **Draw Missing Value Symbols**.
2. Choose the **Style**, **Line Color**, and **Fill Color** you want. The **Line Color** is the "border" around the point symbol, whereas the **Fill Color** is the color inside the point symbol. Figure 6.20 (page 6-15) shows an example of a dark blue **Line Color** and a light blue **Fill Color**.
3. In the **Size** box, specify the size of the point (in pixels) either by selecting a size from the list or by typing in a number from 1- 45.
4. This will place the selected symbol on each missing value.

6.5.5 Customizing Curve Labels

The curve **Label** worksheet, available from the **Curves** tabs of the **Edit Curve Properties** editor, the **Plot Properties** editor, and the **Default Line Styles** options editor (Section 6.2) are shown in Figure 6.21. This worksheet allows you to customize curve labels.



//BEECH CREEK STA/FLOW-REG--INST/01OCT1993/IR-MONTH/DCP-RAW/ Properties

Select Viewport: Viewport (1,1)

Curves:
BEECH CREEK STA DCP-RAW FLOW-REG--INST
BEECH CREEK STA DCP-REV FLOW-REG--INST

☑ Draw Curve ☑ Show in Legend

DSS Path: //BEECH CREEK STA/FLOW-REG--INST/01OCT1993/IR-MONTH/DCP-RAW/

Label
☑ Show Label
Label Text: Beech Creek Flow
Alignment: Left
Position: Above

Sample
Sample Text

Figure 6.21 Curve Label Editing Tab

To customize curve labels:

1. Check the **Show Label** checkbox.
2. Enter the text you want to appear in the curve label in the **Label Text** box. You can also use text substitution for a curve label; see Section 6.16 for more information.
3. Using the **Alignment** list to select the horizontal position of the label on the curve; **Left**, **Center**, or **Right**.
4. To set the vertical position of the label on the curve, from the **Position** list, click **Above**, **Center**, or **Below**.

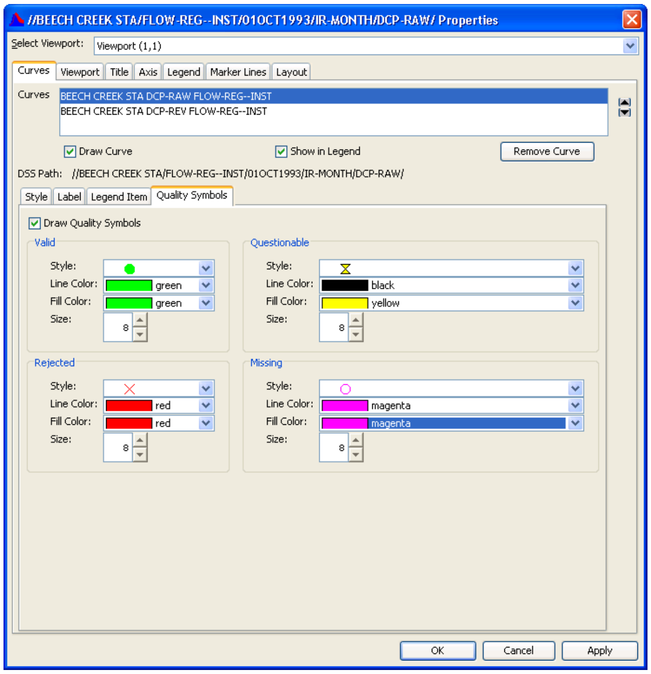
The **Sample** box at the bottom of the editor provides a preview of how your label will display.

6.5.6 Customizing Legend Items

See Section 6.9.3 for information on customizing legend items.

6.5.7 Customizing Curve Quality Symbols

The curve **Quality Symbols** worksheet, available from the **Curves** tabs of the **Edit Curve Properties** editor, the **Plot Properties** editor, and the **Default Line Styles** options editor (Section 6.2) are shown in Figure 6.22. This worksheet allows you to customize curves for plots that have quality set for their data.



//BEECH CREEK STA/FLOW-REG--INST/01OCT1993/IR-MONTH/DCP-RAW/ Properties

Select Viewport: Viewport (1,1)

Curves
BEECH CREEK STA DCP-RAW FLOW-REG--INST
BEECH CREEK STA DCP-REV FLOW-REG--INST

☑ Draw Curve ☑ Show in Legend

DSS Path: //BEECH CREEK STA/FLOW-REG--INST/01OCT1993/IR-MONTH/DCP-RAW/

☑ Draw Quality Symbols

Valid
Style: ●
Line Color: 🟩 green
Fill Color: 🟩 green
Size: 8

Questionable
Style: ⧗
Line Color: ■ black
Fill Color: 🟨 yellow
Size: 8

Rejected
Style: ✕
Line Color: 🟥 red
Fill Color: 🟥 red
Size: 8

Missing
Style: ○
Line Color: magenta
Fill Color: magenta
Size: 8

Figure 6.22 Curve Quality Symbols Editing Tab

To customize quality symbols:

1. Check the **Draw Quality Symbols** checkbox.
2. Select a symbol **Style**, **Line Color**, **Fill Color**, and **Size** for each quality of data; **Valid**, **Questionable**, **Rejected**, and **Missing**.

6.6 Customizing Viewport Properties

Viewports are the gridded areas in the plot window that contain plot curves. You can customize the border around the viewport, the background color and pattern, and the appearance of gridlines.

6.6.1 Customizing Viewport Borders and Background

- From the **Edit** menu, click **Plot Properties**. When the **Plot Properties** editor opens, click the **Viewport** tab.

-Or-

- Right-click in a blank area inside the viewport with the **Pointer Tool** , and then click **Edit Properties** from the shortcut menu (Figure 6.23). The **Viewport Properties** editor will open.

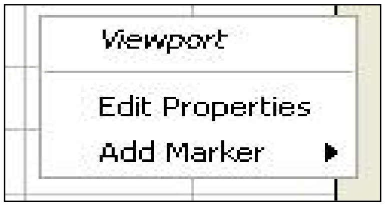
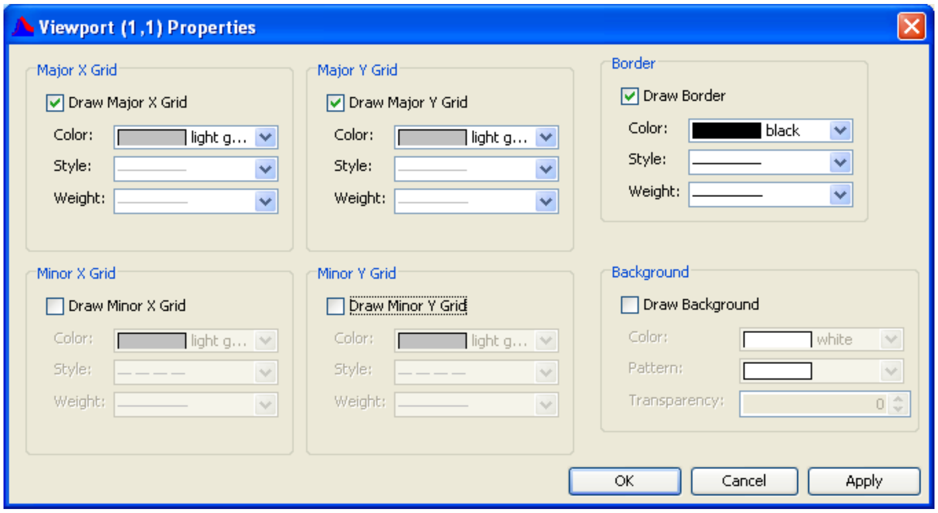


Figure 6.23 Shortcut Menu - Viewport Properties

To specify the default border and background of viewports for *all* of your plots use the **Default Plot Properties** editor; from the **Edit** menu, click **Default Plot Properties**. Once the **Default Plot Properties** editor opens, select the **Viewport** tab. Changes made in the **Default Plot Properties** editor will apply to all viewports.

The **Viewport (#,#) Properties** editor, which displays the viewport number in the title bar is shown in Figure 6.24 (page 6-19). This number reflects the position of the viewport on the **Plot** dialog box. For example: if you had two viewports in you plot, the top viewport's number will display as (1,1). The bottom plot would display as (2,1). The worksheet, accessed from the shortcut menu, contains the same items as the **Viewport** tab of the **Plot Properties** editor and the **Viewport** tab of the **Default Plot Properties** editor. In the **Plot Properties** editor there is a **Select Viewport** list when multiple viewports are displayed in the plot. Select the viewport you would like to edit using this list.

The Viewport tab has six main groups: Major X Grid, Major Y Grid, Minor X Grid, Minor Y Grid, Border and Background.



Viewport (1,1) Properties

Major X Grid
☑ Draw Major X Grid
Color: light g...
Style:
Weight:

Major Y Grid
☑ Draw Major Y Grid
Color: light g...
Style:
Weight:

Border
☑ Draw Border
Color: black
Style:
Weight:

Minor X Grid
□ Draw Minor X Grid
Color: light g...
Style:
Weight:

Minor Y Grid
□ Draw Minor Y Grid
Color: light g...
Style:
Weight:

Background
□ Draw Background
Color: white
Pattern:
Transparency: 0

Figure 6.24 Viewport Properties Editor - Patterns Tab

To customize the Border and Background of a Viewport:

1. Check the **Draw Border** checkbox at the top of the **Border** group and choose the **Color**, **Style**, and **Weight** for the border line you want to appear around the selected viewport.
2. Select **Draw Background** in the **Background** group and choose the **Color, Pattern,** and **Transparency** you want for the background.

Click **Apply** to save your changes and continue adjusting the appearance of the border and background. Click **OK** when you are finished.

6.6.2 Customizing Viewport Gridlines

To customize gridlines of viewports in an *individual* plot, you can either:

- From the **Edit** menu, click **Plot Properties**. When the **Plot Properties** editor opens, click the **Viewport** tab.

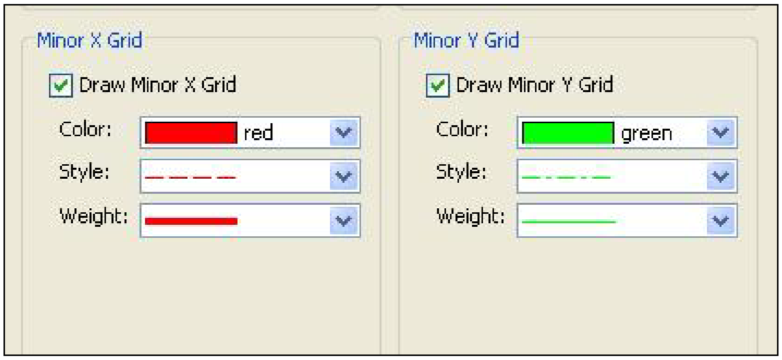
-Or-

- Right-click in a blank area inside the viewport with the **Pointer Tool** ↖, and then click **Edit Properties** from the shortcut menu (see Figure 6.23, page 6-18).

To specify viewport gridlines for *all* of your plots, from the **Edit** menu, select **Default Plot Properties**. Once the **Default Plot Properties** editor opens, choose the **Viewport** tab.

By default, the plot viewport displays gridlines only for the **Major X Grid** and **Major Y Grid**. The default color is light gray. To change the appearance of Major X and Y gridlines, select the **Color**, **Style**, and **Weight** of the gridlines located under their respective groups.

By default, the **Minor X Grid** and **Minor Y Grid** are not selected to draw and do not display in the plot viewport. If you want to display gridlines for the **Minor X Grid** and **Minor Y Grid**, check the **Draw Minor X Grid** and **Draw Minor Y Grid** checkboxes and then make your selections for **Color**, **Style**, and **Weight** (Figure 6.25).



Minor X Grid

☑ Draw Minor X Grid

Color: 🟥 red

Style: – – – –

Weight: ▬

Minor Y Grid

☑ Draw Minor Y Grid

Color: 🟩 green

Style: – · – · –

Weight: —

Figure 6.25 Viewport Properties Editor - Gridlines Tab

Click **Apply** to view your changes without closing the editor. Click **OK** when you are finished.

6.7 Adding and Customizing Marker Lines

You can add marker lines on your plot's X and Y axes and customize the appearance of these markers, as displayed in the top viewport in Figure 6.26.

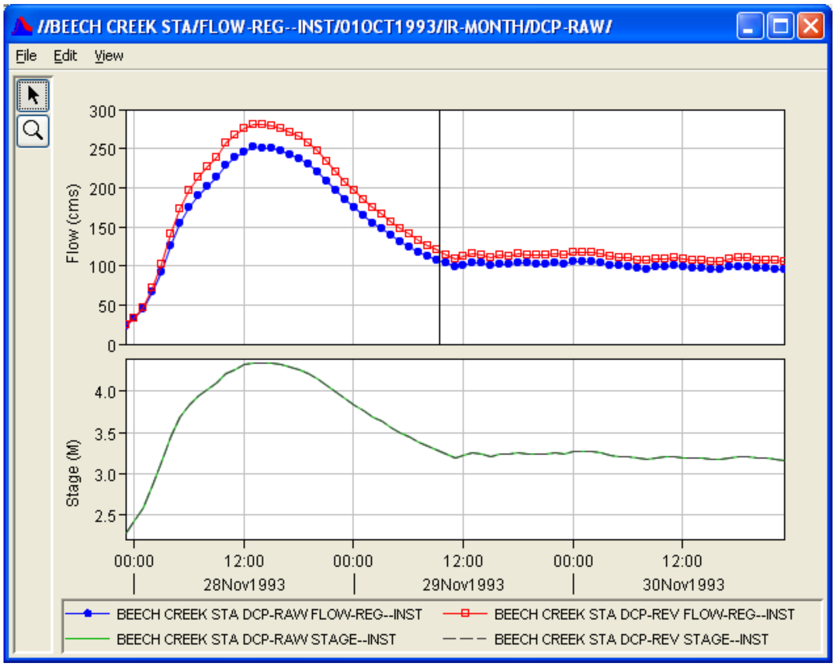

//BEECH CREEK STA/FLOW-REG--INST/01OCT1993/IR-MONTH/DCP-RAW/
File
Edit
View
Flow (cms)
300
250
200
150
100
50
0
Stage (M)
4.0
3.5
3.0
2.5
00:00
12:00
00:00
12:00
00:00
12:00
28Nov1993
29Nov1993
30Nov1993
BEECH CREEK STA DCP-RAW FLOW-REG--INST
BEECH CREEK STA DCP-REV FLOW-REG--INST
BEECH CREEK STA DCP-RAW STAGE--INST
BEECH CREEK STA DCP-REV STAGE--INST

Figure 6.26 Adding Marker Lines to Plots

6.7.1 Adding Markers

To add a marker:

1. Right-click on the location in the plot where you want the marker to appear.
2. From the **Viewport** shortcut menu (Figure 6.27), point to **Add Marker**, and then click either **On X-Axis** or **On Y-Axis**.

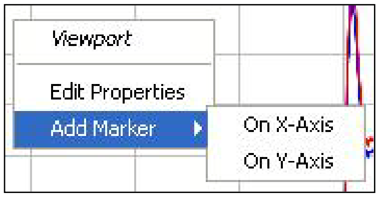


Figure 6.27 Shortcut Menu - Add Marker

The marker will now appear in the plot.

6.7.2 Deleting Markers

To delete a marker line in a plot, right click on the plot with the **Pointer Tool**, ↖ from the shortcut menu, click **Delete** (Figure 6.28).

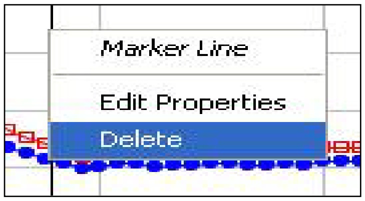


Figure 6.28 Shortcut Menu - Marker Line Properties

6.7.3 Customizing Markers

To edit the properties of a marker in an *individual* plot, you can either:

- From the **Edit** menu, click **Plot Properties**. When the **Plot Properties** editor opens, click the **Marker Lines** tab. This worksheet is available only if a marker exists in the plot. Choose the marker you want to edit from the **Marker Lines** list.

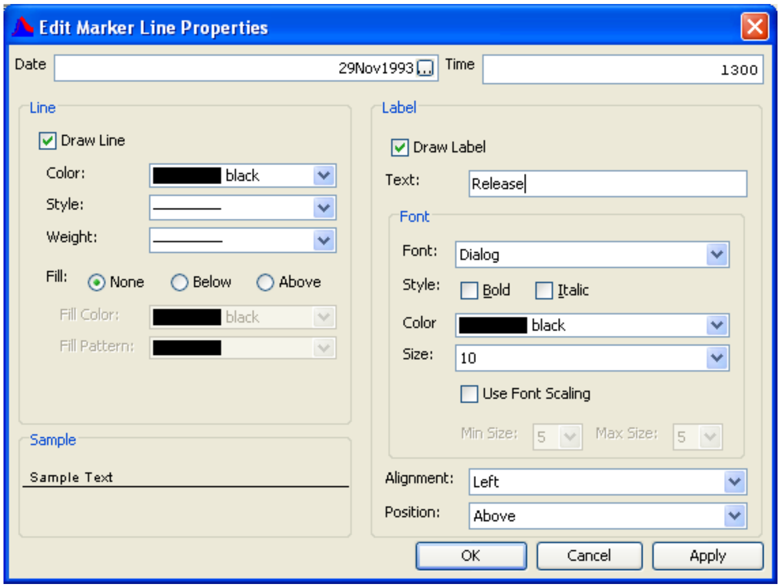
-Or-

- With the **Pointer Tool** ↖ , right-click on the marker you want to edit. From the **Marker Line** shortcut menu, click **Edit Properties** (Figure 6.28). The **Edit Marker Line Properties** dialog box will open.

To specify the default appearance of markers for *all* of your plots, from the **Edit** menu, click **Default Plot Properties**. Once the **Default Plot Properties** editor opens, choose the **Marker Lines** tab.

The interfaces for the specialized **Edit Marker Line** editor and the **Default Plot Properties** editor are very similar. However, the **Plot Properties Editor** dialog box differs in two ways. First, the **Marker Lines** worksheet is available only if a marker exists in the current plot. Second, at the top of the editing panel there is a **Marker Lines** list containing all markers that exist in the current plot.

You can edit the line style and label of a marker through this worksheet. The **Edit Marker Line Properties** editor is shown in Figure 6.29. The **Marker Line** tabs in the **Plot Properties** editor and the **Default Plot Properties** editor use similar worksheets as well.



Edit Marker Line Properties

Date: 29Nov1993 Time: 1300

Line

☑ Draw Line

Color: black

Style:

Weight:

Fill: ● None ○ Below ○ Above

Fill Color: black

Fill Pattern:

Sample

Sample Text

Label

☑ Draw Label

Text: Release

Font

Font: Dialog

Style: □ Bold □ Italic

Color: black

Size: 10

□ Use Font Scaling

Min Size: 5 Max Size: 5

Alignment: Left

Position: Above

Figure 6.29 Marker Line Properties Editor - Style Tab

To edit the line style for markers:

1. Select the **Color**, **Style**, and **Weight** for the marker line.
2. You can display marker lines with a **Fill Above** or **Below** the line. A plot with the **Fill Above** and a hatched **Fill Pattern** selected using a red **Fill Color** is shown in Figure 6.30.

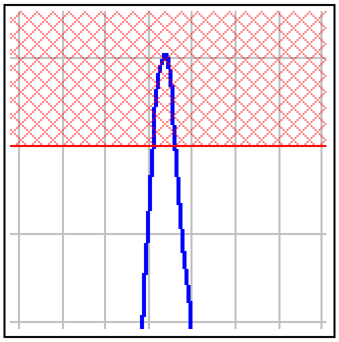


Figure 6.30 Marker Line with Fill Above

The **Sample** box, located in the lower left-hand corner of the worksheet, provides a preview of the new marker line.

Click **Apply** to save your changes without closing the editor.

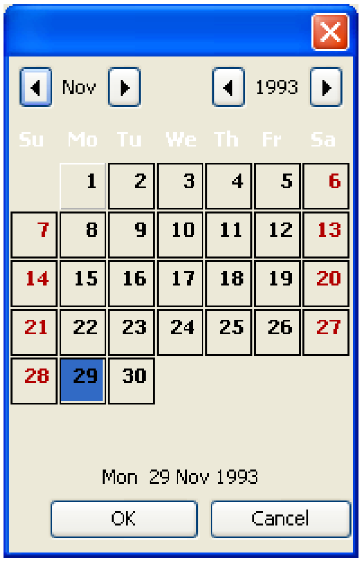
To add a label to a marker line:

1. In the **Label** group, check the **Draw Label** checkbox and type the label name into the **Text** field.
2. In the **Font** section, select the labels **Font**, **Style**, **Color**, and **Size**.
3. Check **Use Font Scaling** if you would like to limit the size of the font when zooming in and out of the plot. Set the **Min Size** and **Max Size** limits.

4. From the **Alignment** list, select the justification of the label; **Left**, **Center**, or **Right** of the marker line.
5. Set the position of the label from the **Position** list, either **Above**, **Center**, or **Below** the marker line.

Set the position of the label from the **Position** list, either **Above**, **Center**, or **Below** the marker line.

1. If the marker line is on a date, the **Date** field will be displayed on the top of the worksheet.
2. To change the date, either enter the date in the format DDMMYYYY (*e.g.*, 21Dec1993) or click the ellipse button … to access the **Calendar Tool** (Figure 6.31).
3. For the **Time** value, enter the time in 24-hour military format (*e.g.*, for 5:08 pm, enter "1708").
4. If the marker line appears on a specific value, the **Value** field will display at the top of the worksheet instead of the **Date** field. To change this value, simply enter a new value in the field.



Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

Mon 29 Nov 1993

Figure 6.31 Calendar Tool

6.7.4 Editing Callouts

You can add descriptive callouts at specific points along a curve (Figure 6.32). To add callouts:

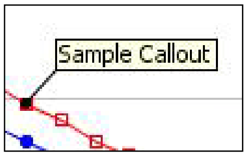


Figure 6.32 Callout

1. Right-click on the location on the curve where you want the callout to appear.
2. From the shortcut menu (Figure 6.33), select **Add Callout**.

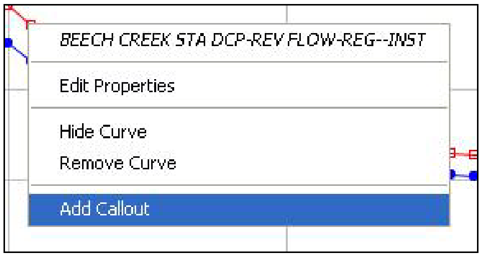


Figure 6.33 Shortcut Menu - Add Callout

3. In the **Add Callout** dialog box (see Figure 6.34, page 6-24), enter the text you want to appear in the callout, and then click **OK**.

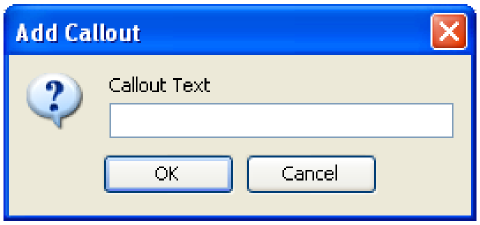


Figure 6.34 Add Callout Dialog Box

To hide all callouts on a curve, right-click on the curve in the plot, or in the legend, and select **Hide Callouts** from the shortcut menu.

Once you have hidden callouts, **Hide Callouts** in the shortcut menu changes to **Show Callouts**; allowing you to return callouts to the plot. To permanently remove all callouts from a curve, right-click on a curve in the plot, or in the legend, and click **Clear Callouts** from the shortcut menu.

6.8 Customizing Axes

You can choose either a linear or log axis type, specify the axis scale, modify tic marks, and customize axis labels. Probability plots are generated for paired data sets with a type of "PROB", and cannot be changed without changing the data type.

6.8.1 Changing Axis Type

By default, plots display using a linear scale (**Linear Axis**), in which the axis increases and decreases by x. You can also use the log scale, which allows you to view curves that grow exponentially in a near straight line, because the axis increases or decreases by the log (x). For example, you might wish to use the log scale when the axis has evenly-spaced major tics with values of 1,10,100,1000, and so on, such as in a performance history plot showing many years of data.

To change the axis type of an individual plot, right-click on an axis, then from the **Axis Tics** shortcut menu, point to **Set Axis Type,** click **Log Axis** or click **Linear Axis**; depending on which axis type is already in use. In Figure 6.35, the axis is using a **Linear Axis**, so the choice available is **Log Axis**.

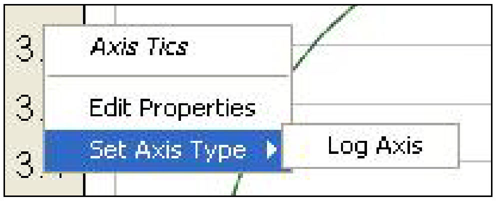


Figure 6.35 Shortcut Menu - Set Axis Type

6.8.2 Specifying Axis Scale

You can specify the axis scale and tic interval for individual plots. To do this, select the editor either:

- From the **Edit** menu, click **Plot Properties**. When the **Plot Properties** editor opens, click the **Axis** tab. From the **Axis** worksheet, select the axis you want to edit from the **Axis** list. Then select the **Scale** tab.

  -Or-

- Right-click on the axis using the **Pointer Tool** and from the **Axis Tics** shortcut menu (Figure 6.36), click **Edit Properties**. When the **Axis Properties** dialog box opens, click the **Scale** tab.

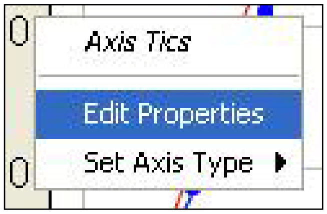
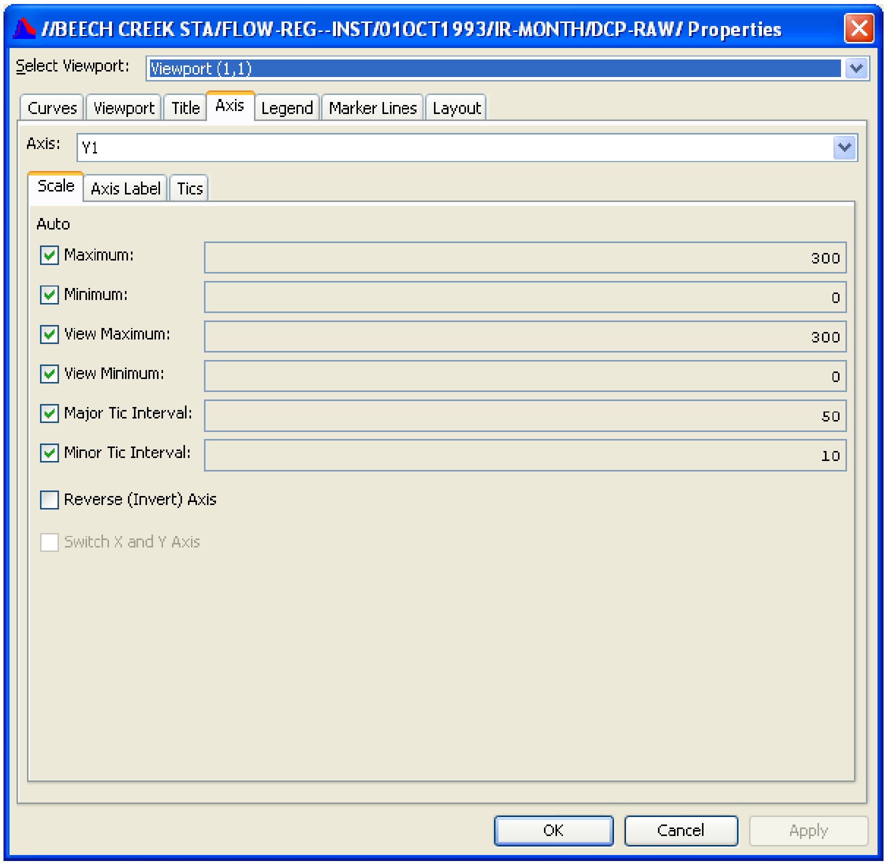


Figure 6.36 Shortcut Menu - Axis Tics

The **Plot Properties** editor with the **Axis** tab selected and the **Scale** worksheet open is shown in Figure 6.37. The **Axis** worksheets of the **Plot Properties** editor and the **Default Plot Properties** editor are nearly identical, except for the **Axis** list on the **Plot Properties** editor. The **Axis Label** and **Tics** sub-worksheets are available in all three editors.



Select Viewport: Viewport (1,1)

Axis: Y1

Auto

☑ Maximum: 300

☑ Minimum: 0

☑ View Maximum: 300

☑ View Minimum: 0

☑ Major Tic Interval: 50

☑ Minor Tic Interval: 10

☐ Reverse (Invert) Axis

☐ Switch X and Y Axis

Figure 6.37 Plot Properties Editor - Scale Worksheet

With the **Scale** worksheet, you can specify the range of the scale, the amount of the scale that is visible, and the tic intervals.
If the checkboxes under the **Auto** heading are checked, the plot will automatically select the scale. Otherwise, as you zoom in and out of the plot, its view values change while the minimum and maximum scale range values remain fixed:

- **Maximum:** enter the value of the maximum range of the scale.
- **Minimum:** enter the value of the minimum range of the scale.
- **View Maximum:** enter the maximum visible range of the scale.
- **View Minimum:** enter the minimum visible range of the scale.

Tic intervals are the distances between tics on the axis scale:

- **Major Tic Interval:** specify the distance between each major tic.
- **Minor Tic Interval:** specify a value less than or equal to the major tic value.

You can also choose to reverse the axis and invert the data by selecting **Reverse (Invert) Axis**. If the data set is paired, you can switch the X and Y axis, so what is plotted on the X axis becomes plotted on the Y axis instead.

Click **Apply** to save and view your changes without closing the editor. Click **OK** to save your changes and close the editor.

6.8.3 Modifying Tic Marks

You can modify the color of tic marks, choose whether or not major and minor tic marks display, and specify whether labels display. (See also "**Specifying Axis Scale**" in Section 6.8.2 for information about modifying tic intervals.)

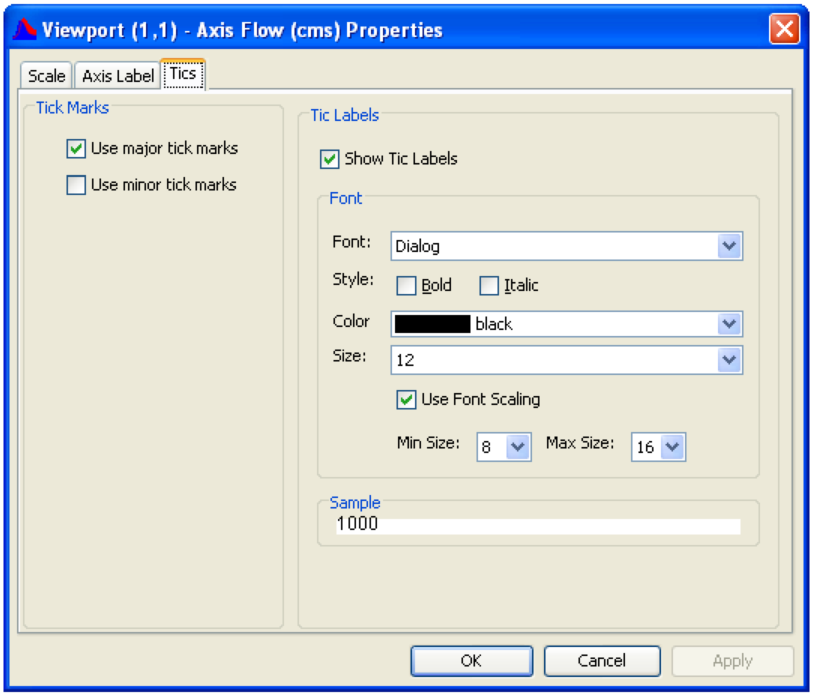
To modify tic marks in an *individual* plot, you can either:

- From the **Edit** menu, click **Plot Properties**. When the **Plot Properties** editor opens, click the **Axis** tab. From the **Axis** worksheet, select the axis you want to edit from the **Axis** list. Once you have this set, click the **Tics** tab to set tic marks.

  -Or-
- Right-click on the axis using the **Pointer Tool** and from the **Axis Tics** shortcut menu (see Figure 6.36, page 6-25), click **Edit Properties**. When the **Axis Properties** dialog box opens, click the **Tics** tab.

To specify default settings for axis tics in *all* of your plots, click **Default Plot Properties** from the plot **Edit** menu. Once the **Default Plot Properties** editor opens, choose the **Axis** tab, then the **Tics** tab.

The **Tics** worksheet of the **Axis Properties** editor is shown in Figure 6.38 above. The **Tics** worksheet is nearly identical on the **Axis Properties** editor, **Plot Properties** editor, and **Default Plot Properties** editor.



☑ Use major tick marks
□ Use minor tick marks

☑ Show Tic Labels

Font: Dialog
Style: □ Bold □ Italic
Color ■ black
Size: 12
☑ Use Font Scaling
Min Size: 8 Max Size: 16

1000

Figure 6.38 Axis Properties Editor - Tics Tab

However, at the top of the **Plot Properties** editor is an **Axis** list containing all of the axes available for editing in the selected viewport. In the **Plot Properties** editor, you must choose an axis to edit before you can make any changes to the tics.

By default, plot axes display major tic marks with labels. To turn these defaults off, uncheck the **Use major tick marks** and **Show Tic Labels** check boxes by clicking on them.

You can also select **Use minor tic marks**, to display the minor tic marks in the viewport.

To change the Tic Label:

1. The tic and tic labels use the same color. To change their color, select a new color from the **Color** list in the **Tic Labels** group.
2. Select the label's typeset using the **Font** list. The font can display as **Bold** or **Italic** by checking the checkboxes next to either **Style** option.

3. By default the **Size** is ignored, and **Use Font Scaling** is selected. **Use Font Scaling** allows the program to determine the best font size to display, using the **Min Size** and **Max Size** to limit the range of the font size.

Click **Apply** to save and view changes without closing the editor. Once you are done making changes in the editor, click **OK** to save the changes and close the editor.

6.8.4 Customizing Axis Labels

You can add borders and backgrounds to axis labels. To customize axis labels in an *individual* plot, you can either:

- From the **Edit** menu, click **Plot Properties**. When the **Plot Properties** editor opens, select the axis label's viewport in the **Select Viewport** list at the top of the editor. Then click the **Axis** tab. On the **Axis** worksheet, select the axis you want to edit from the **Axis** list, and then click the **Axis Label** tab to set axis label properties.

-Or-

- Right-click on the axis label with the **Pointer Tool** . From the ***Axis Label*** shortcut menu (Figure 6.39), click **Edit Properties**. The **Label Properties** editor will open.

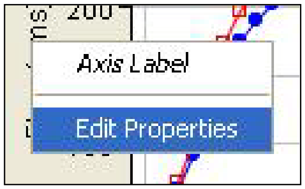


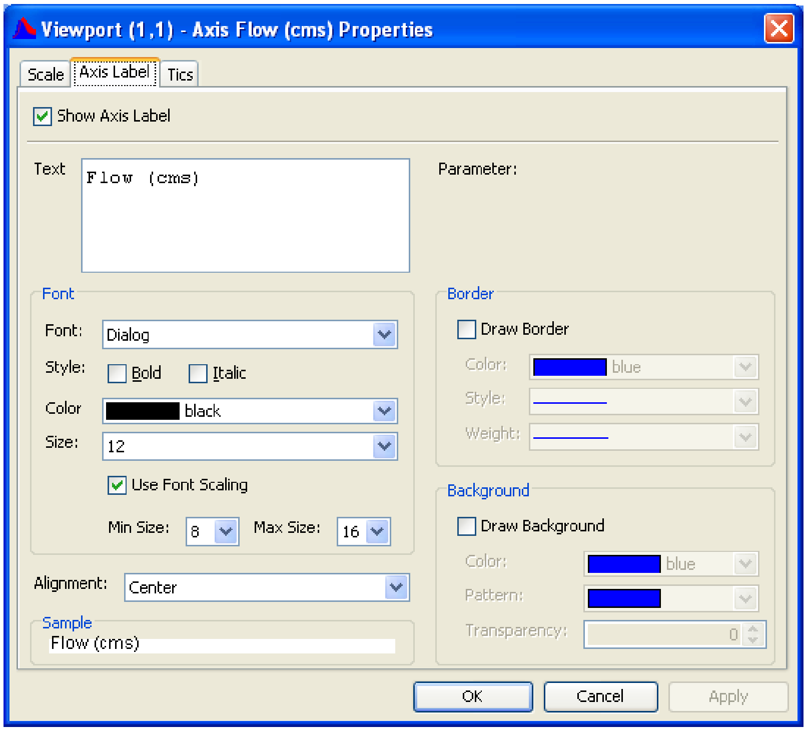
Figure 6.39 Shortcut Menu - Axis Label

To specify default settings for axis labels in *all* of your plots, select **Default Plot Properties** from the plot’s **Edit** menu. Once the **Default Plot Properties** editor opens, choose the **Axis** tab, then the **Axis Label** tab.

The **Label Properties** editor is shown in Figure 6.40 (page 6-29). The same worksheet is available on the **Axis Label** tabs of the **Plot Properties** editor and **Default Plot Properties** editor, with a few differences. The **Plot Properties** editor has an **Axis** list containing all of the axes available for editing in the current selected viewport, which is displayed in the **Select Viewport** list.

The **Axis Label** worksheet has three groups: **Font**, **Border** and **Background**. The **Sample** box, located at the bottom left of the editor, provides a preview of the label changes.

The current label text is displayed in the **Text** box. To change the wording, highlight the displayed text in the box and type in the new axis



Viewport (1,1) - Axis Flow (cms) Properties

☑ Show Axis Label

Text: Flow (cms)

Parameter:

Font

Font: Dialog

Style: □ Bold □ Italic

Color: 🟥 black

Size: 12

☑ Use Font Scaling

Min Size: 8 Max Size: 16

Alignment: Center

Sample: Flow (cms)

Border

□ Draw Border

Color: blue

Style:

Weight:

Background

□ Draw Background

Color: blue

Pattern:

Transparency: 0

Figure 6.40 Label Properties Editor

label name. Text substitution can also be used here. See section 6.16 for more information on text substitution.

The font of the axis label can be customized using the tools located under the **Font** group. The text typeset can be changed using the **Font** list. The **Style** of the font can be changed by checking either the **Bold** or **Italic** checkboxes. Choose the color of your font in the **Color** list. By default the **Use Font Scaling** checkbox is set. This option allows the program to determine the best size of the font, limiting the range between the selected **Min Size** and **Max Size**. Otherwise, you can un-checking the **Use Font Scaling** and set a constant font size in the **Size** list.

The **Alignment** list allows you to position the label, Left, Right, or Center, relative to the axis.

To add a border around the axis label, check the **Draw Border** checkbox in the **Border** group, and then select the **Color**, **Style**, and **Weight** for the borderline.

To add a background to the axis label, check the **Draw Background** checkbox in the **Background** group, and then select a **Color**, **Pattern**, and the **Transparency** level.

Use the **Show Axis Label** checkbox to add or remove the axis label from the plot window.

Click **Apply** to save and view your changes without closing the editor. Click **OK** when you are finished editing axis label properties.

6.9 Customizing Legends

As illustrated in Figure 6.41, you can add titles to a plot legend, add text and graphics to the right and left sides of the Legend box, and customize the curve labels in the Legend box. You can also specify whether the legend appears below, to the right, or inside the plot's viewports. The legend can even be displayed in a separate window or hidden from display.

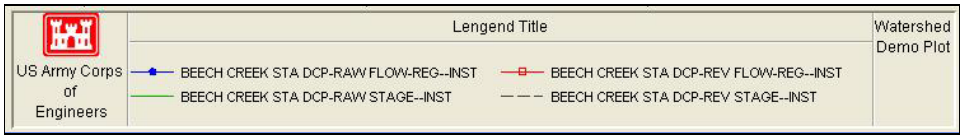

US Army Corps
of
Engineers
Lengend Title
BEECH CREEK STA DCP-RAW FLOW-REG--INST
BEECH CREEK STA DCP-RAW STAGE--INST
BEECH CREEK STA DCP-REV FLOW-REG--INST
BEECH CREEK STA DCP-REV STAGE--INST
Watershed
Demo Plot

Figure 6.41 Customizing Legends

To add a title or side blocks to an *individual* plot's legend, you can either:

- From the **Edit** menu, click **Plot Properties**. When the **Plot Properties** editor opens, select the **Legend** tab.

-Or-

- Right-click in a blank area inside the legend panel of the plot with the **Pointer Tool** ↖. From the ***Legend Panel*** shortcut menu (Figure 6.42), click **Edit Properties**. The **Legend Properties** editor will open.

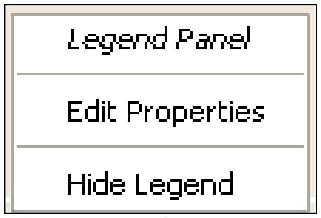


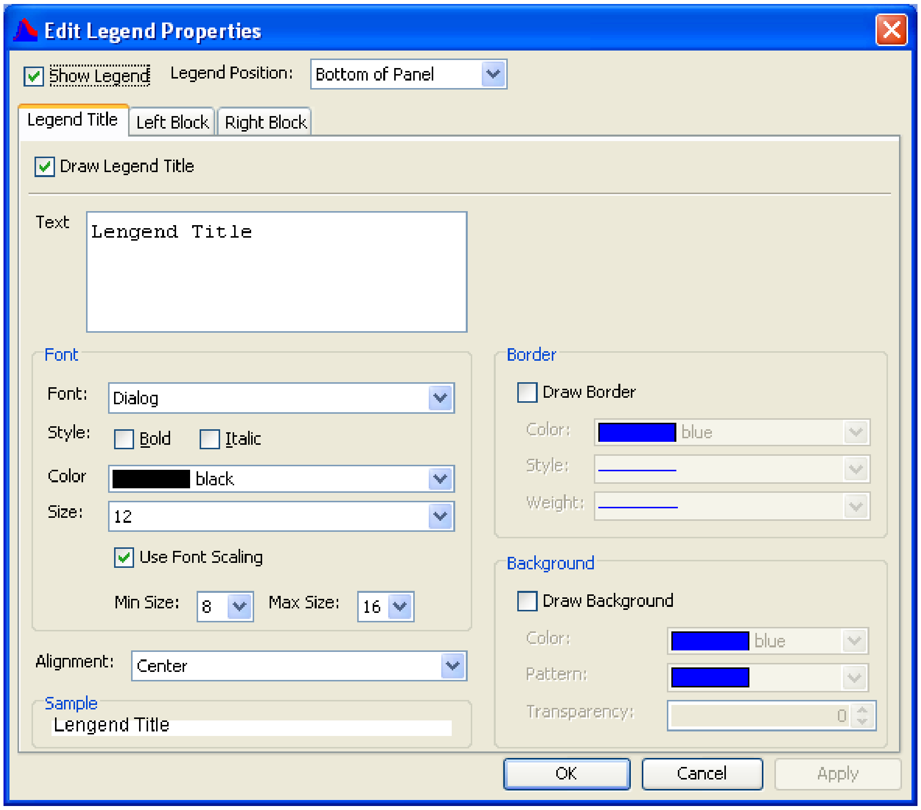
Figure 6.42 Shortcut Menu - Legend Panel

To specify default settings for *all* of your plot legends, from the **Edit** menu, click **Default Plot Properties**. Once the **Default Plot Properties** editor opens, select the **Legend** tab.

Whether you are using the **Plot Properties** editor, **Legend Properties** editor, or **Default Plot Properties** editor, the worksheet for editing legend titles is the same.

The **Legend Properties** editor is shown in Figure 6.43 (page 6-31). This worksheet, accessed from the shortcut menu, contains the same fields as the **Legend** worksheets of both the **Plot Properties** editor and the **Default Plot Properties** editor.

To remove the legend from the plot window, uncheck the **Show Legend** checkbox on the editor, or from the plot window's **View** menu select **Hide Legend**. If you would like to add the legend back to the plot window,



☑ Show Legend Legend Position: Bottom of Panel

☑ Draw Legend Title

Text: Lengend Title

Font: Dialog

Style: □ Bold □ Italic

Color: ■ black

Size: 12

☑ Use Font Scaling

Min Size: 8 Max Size: 16

Alignment: Center

Sample: Lengend Title

□ Draw Border

Color: ■ blue

Style:

Weight:

□ Draw Background

Color: ■ blue

Pattern:

Transparency: 0

Figure 6.43 Legend Properties Editor

either re-check the **Show Legend** box on the editor, or on the plot window from the **View,** click **Show Legend**.

6.9.1 Legend Title

To add and customize a title for the legend:

1. On the **Legend Title** tab, select the **Draw Legend Title** checkbox. Then enter the title you want to appear along the top of the legend in the **Text** box.
2. In the **Font** group, select the typeset for the title in the **Font** list. The **Style** of the font can also be changed by checking either the **Bold** or **Italic** checkboxes.
3. By default the **Use Font Scaling** checkbox is set. This option allows the program to determine the best size of the font, limiting the range between the selected **Min Size** and **Max Size**. Otherwise, you can un-checking the **Use Font Scaling** and set a constant font size in the **Size** list.
4. Select the **Alignment** to position the title to the **Left**, **Right**, or **Center** of the legend.
5. If you would like the title to use a custom border check the **Draw Border** checkbox in the **Border** group. Then select the **Color**, **Style**, and **Weight** for the borderline.

6. Similarly, if you would like the title to use a custom background check the **Draw Background** checkbox in the **Background** group. Then select the **Color**, **Pattern**, and **Transparency** for the title's background.
7. The **Sample** box in the lower left corner will display a preview of your title.

6.9.2 Legend Blocks

To add and customize a **Left** or **Right Block** to the legend:

1. First select the **Left Block** or **Right Block** tab in the **Legend Properties** editor and check the **Draw Left Legend Block** or the **Draw Right Legend Block** checkbox; depending on which workbook you are editing. The workbooks are identical, except for these **Draw** checkboxes.
2. Type in the text for the block in the **Text** box.
3. In the **Font** group, select the typeset for the block in the **Font** list. The **Style** of the font can also be changed by checking either the **Bold** or **Italic** checkboxes.
4. By default the **Use Font Scaling** checkbox is set. This option allows the program to determine the best size of the font, limiting the range between the selected **Min Size** and **Max Size**. Otherwise, you can un-checking the **Use Font Scaling** and set a constant font size in the **Size** list.
5. Select the **Alignment** to position the block's text to the **Left**, **Right**, or **Center** of the legend.
6. If you wish to display a graphic in the block, under the **Icon** group either type the exact path* and filename of the image in the **File** field, or use the ellipse button ... to use a file browser to find and select the image.
7. If you would like the block to use a custom border, check the **Draw Border** checkbox in the **Border** group. Then select the **Color**, **Style**, and **Weight** for the borderline.
8. Similarly, if you would like the block to use a custom background, check the **Draw Background** checkbox in the **Background** group. Then select the **Color**, **Pattern**, and **Transparency** for the title's background.
9. The **Sample** box in the lower left corner will display a preview of your legend block.

To move the legend to a new location, either select the location in the **Legend Position** list on the **Legend** tab of the editor or from the plot window, from the **View** menu, click **Legend Placement**. By default the legend is placed at the **Bottom of Panel**. Other options include: **Right of Panel**, **Separate Window**, **Viewport Upper Left**, and **Viewport Upper**

Right. If the legend displays in the viewport, the title and legend blocks will not appear, (Figure 6.44).

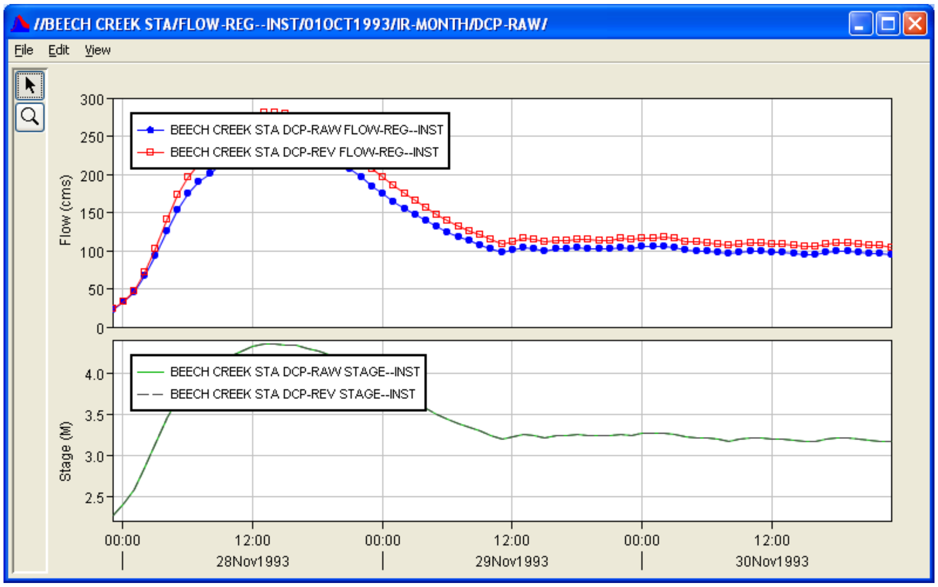

//BEECH CREEK STA/FLOW-REG--INST/01OCT1993/IR-MONTH/DCP-RAW/
File
Edit
View
Flow (cms)
300
250
200
150
100
50
0
BEECH CREEK STA DCP-RAW FLOW-REG--INST
BEECH CREEK STA DCP-REV FLOW-REG--INST
Stage (M)
4.0
3.5
3.0
2.5
BEECH CREEK STA DCP-RAW STAGE--INST
BEECH CREEK STA DCP-REV STAGE--INST
00:00
12:00
00:00
12:00
00:00
12:00
28Nov1993
29Nov1993
30Nov1993

Figure 6.44 Viewport Upper Left

If the **Legend Position** is set for **Right of Panel** and the legend contains right and left blocks, the left block will be removed and the right block will display on the bottom of the legend, (Figure 6.45).

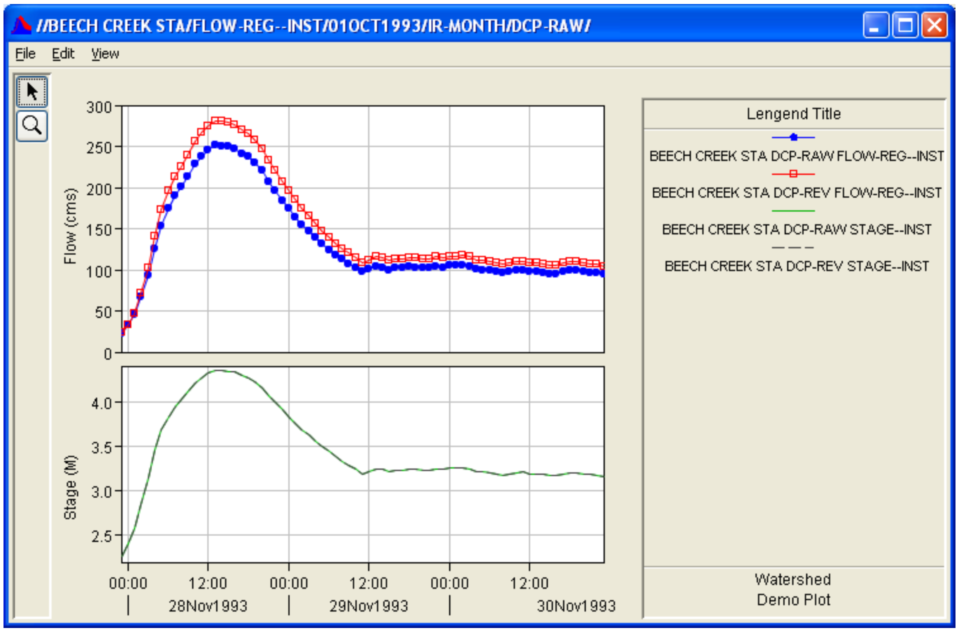

//BEECH CREEK STA/FLOW-REG--INST/01OCT1993/IR-MONTH/DCP-RAW/
File
Edit
View
Flow (cms)
300
250
200
150
100
50
0
Stage (M)
4.0
3.5
3.0
2.5
00:00
12:00
00:00
12:00
00:00
12:00
28Nov1993
29Nov1993
30Nov1993
Lengend Title
BEECH CREEK STA DCP-RAW FLOW-REG--INST
BEECH CREEK STA DCP-REV FLOW-REG--INST
BEECH CREEK STA DCP-RAW STAGE--INST
BEECH CREEK STA DCP-REV STAGE--INST
Watershed
Demo Plot

Figure 6.45 Viewport Right of Panel

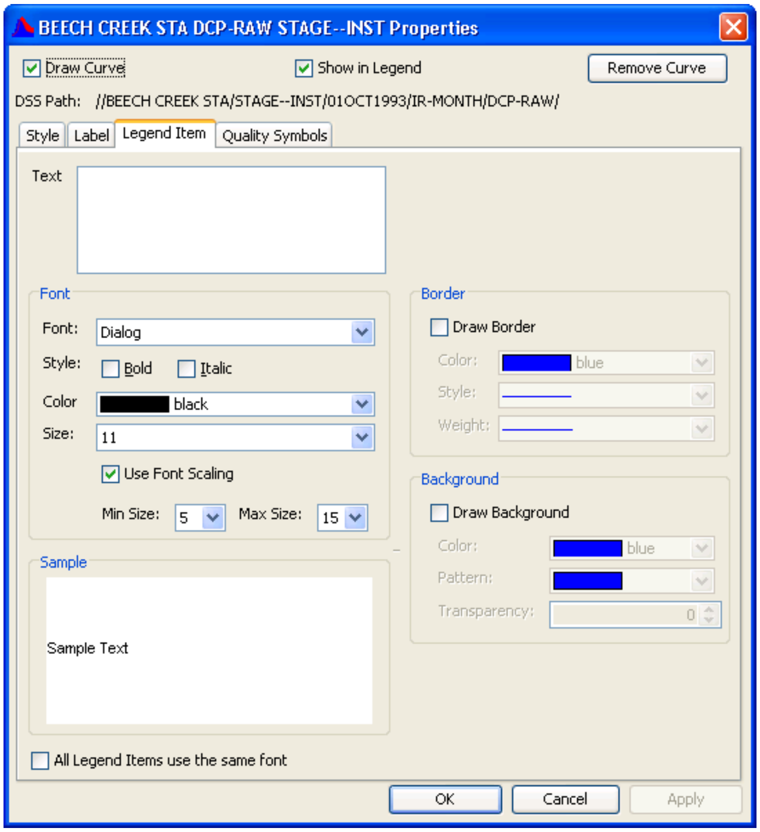
Click **Apply** to view your changes without closing the editor. When you are done making changes to the legend click **OK** to save your changes and close the editor.

6.9.3 Customizing Legend Items

By default, the curves display their DSS Pathnames inside the legend. However, the legend labels, named **Legend Items**, can be customized for individual curves.

To change a curve's **Legend Item** in an *individual* plot’s legend, you can either:

- From the **Edit** menu, click **Plot Properties**. When the **Plot Properties** editor opens, select the **Curves** tab, then the **Legend Item** tab (Figure 6.46).



BEECH CREEK STA DCP-RAW STAGE--INST Properties

☑ Draw Curve ☑ Show in Legend

DSS Path: //BEECH CREEK STA/STAGE--INST/01OCT1993/IR-MONTH/DCP-RAW/

Font: Dialog
Style: □ Bold □ Italic
Color: black
Size: 11
☑ Use Font Scaling
Min Size: 5 Max Size: 15

□ Draw Border
Color: blue

□ Draw Background
Color: blue
Transparency: 0

□ All Legend Items use the same font

Figure 6.46 Legend Item Curve Properties Editor

-Or-

- Right-click on a Legend Item (curve label) in the legend with the **Pointer Tool** ↖. From the shortcut menu (Figure 6.47), select **Edit Properties**. The **Curve Properties** editor will open.

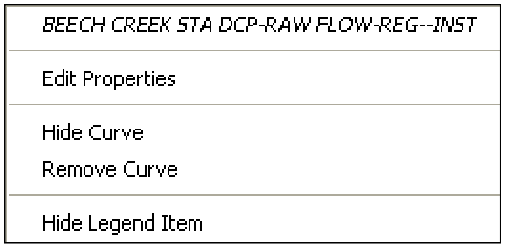


Figure 6.47 Shortcut Menu - Legend Curves

To specify default settings for *all* plot **Legend Items**, from the **Edit** menu, click **Default Plot Properties**. Once the **Default Plot Properties** editor opens, select the **Curves** tab, then the **Legend Item** tab. Whether you are using the **Plot Properties** editor, **Legend Properties** editor, or **Default Plot Properties** editor, the worksheet for editing Legend Items is the same.

To customize a **Legend Item**:

1. Enter the text you would like to display in the legend for the curve in the **Text** box. Text Substitution can also be used here, see Section 6.16 for more information.
2. In the **Font** group, select the typeset for the legend item in the **Font** list. A **Style** can be used for the font by checking the **Bold** and/or **Italic** checkboxes.
3. By default, the **Use Font Scaling** checkbox is set. This option allows the program to determine the best size of the font, limiting the range between the selected **Min Size** and **Max Size**. Otherwise, you can un-checking the **Use Font Scaling** and set a constant font size in the **Size** list.
4. If you would like the Legend Item to have a custom border around it, check the **Draw Border** checkbox in the **Border** group. Then select the **Color**, **Style**, and **Weight** for the borderline.
5. Similarly, if you would like the **Legend Item** to use a custom background check the **Draw Background** checkbox in the **Background** group. Then select the **Color**, **Pattern**, and **Transparency** for the item's background.
6. The **Sample** box in the lower left corner will display a preview of your Legend Item.
7. If you would like the font changes to apply to all of the Legend Items in the plot, click the **All Legend Items use the same font** checkbox at the bottom of the worksheet.

Click **Apply** to save your changes. Once you are done with the editor, click **OK** to save the changes and close the window.

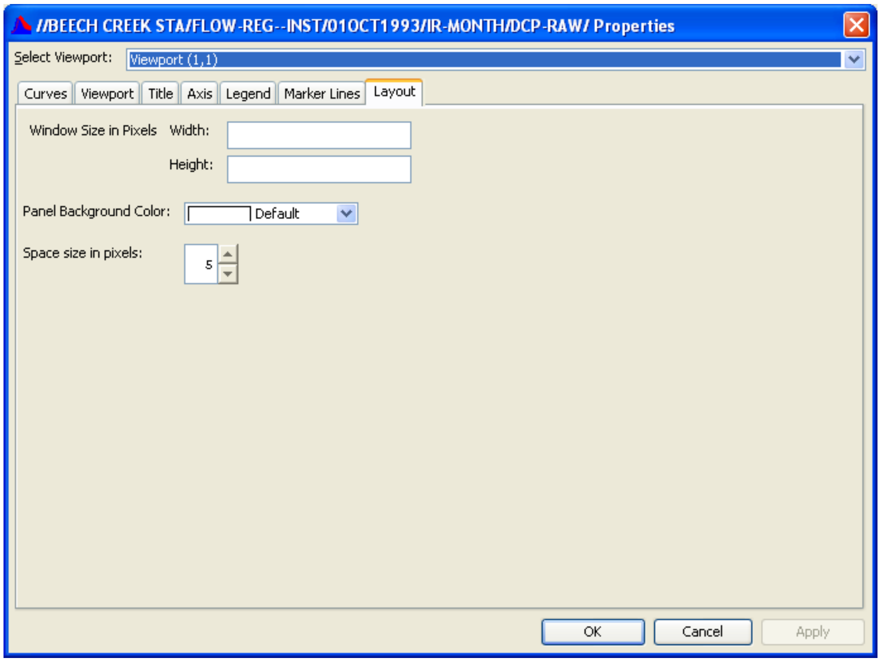
6.10 Customizing Window Panels

You can customize the background color, size, and viewport spacing the window plots are displayed in.

To customize panel properties of an *individual* plot, from the **Edit** menu, click **Plot Properties**. Once the **Plot Properties** editor opens, select the **Layout** tab.

To specify panel properties for *all* of your plots, from the **Edit** menu, click **Default Plot Properties**. Once the **Default Plot Properties** editor opens, choose the **Layout** tab.

Whether you are using the **Plot Properties** editor or the **Default Plot Properties** editor, the **Layout** worksheet is the same. The **Layout** worksheet for the **Plot Properties** editor is shown in Figure 6.48.



//BEECH CREEK STA/FLOW-REG--INST/01OCT1993/IR-MONTH/DCP-RAW/ Properties

Select Viewport: Viewport (1,1)

Window Size in Pixels Width:

Height:

Panel Background Color: Default

Space size in pixels: 5

Figure 6.48 Default Plot Properties Editor - Layout Tab

6.10.1 Customizing the Panel Background Color

To customize the panel background color, select a color in the **Panel Background Color** list. The default color is named **Default**. The **Default** color uses the background color set by your computer's operating system.

6.10.2 Customizing the Horizontal Spacer Size

In the **Plot Properties** editor and **Default Plot Properties** editor, horizontal spacer size refers to the space between viewports in plots with multiple viewports.

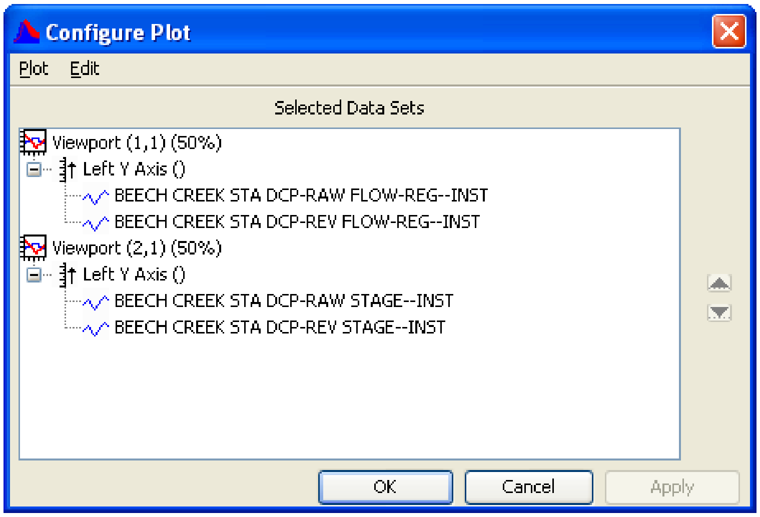
To specify the horizontal spacer size using the **Layout** tab of the **Plot Properties** editor or **Default Plot Properties** editor, in the **Space size in pixels** field, either manually type in a size between 0 and 50 or use the up and down arrows to select a size.

6.10.3 Customizing the Window Size

In the **Plot Properties** editor and **Default Plot Properties** editor, on the **Layout** tab enter the **Width** and **Height** of your plot window. By default these values are empty, implying that the program will determine the best size for each plot window.

6.11 Customizing Plot Layout

The **Configure Plot** editor (Figure 6.49) displays plot components in a "tree" structure and allows you to customize the layout of an individual plot. You can add and remove axes as well as add, remove, arrange the order of, and set the weight of viewports in the plot window panel.



Viewport (1,1) (50%)
Left Y Axis ()
BEECH CREEK STA DCP-RAW FLOW-REG--INST
BEECH CREEK STA DCP-REV FLOW-REG--INST
Viewport (2,1) (50%)
Left Y Axis ()
BEECH CREEK STA DCP-RAW STAGE--INST
BEECH CREEK STA DCP-REV STAGE--INST

Figure 6.49 Configure Plot Editor

When you customize the layout of a plot using the **Configure Plot** editor, your changes apply only to that individual plot unless you export the plot's properties (see Section 6.12).

To access the **Configure Plot** editor, from the **Edit** menu, select **Configure Plot Layout**.

6.11.1 Adding and Removing Viewports

To add a new viewport to a plot, from the **Configure Plot** editor's **Edit** menu click **Add Viewport**. The new viewport will appear at the bottom of the "tree" in the **Selected Data Sets** box.

To remove an empty viewport, you can either:

- Click on the name of the viewport in the **Selected Data Sets** "tree". From the **Edit** menu, click **Remove Viewport**.

  -Or-

- Right-click on the viewport's name in the **Selected Data Sets** "tree". From the shortcut menu (Figure 6.50), click **Remove**.

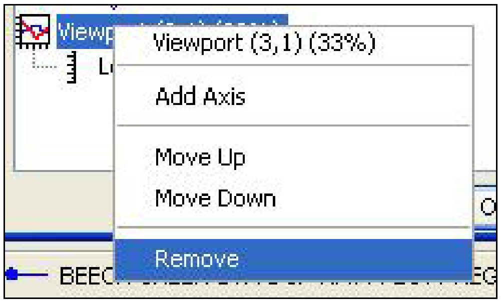
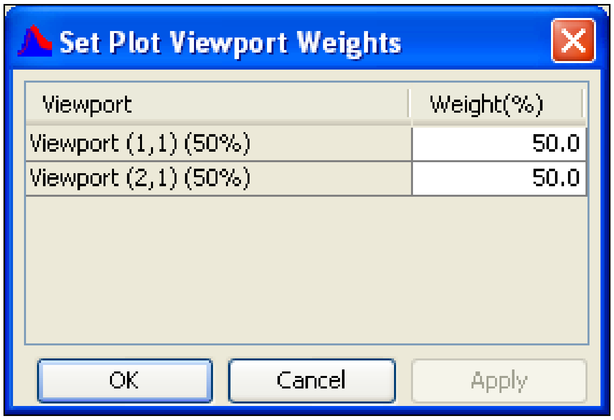


Figure 6.50 Shortcut Menu - Viewport

6.11.2 Setting Viewport Weights

You can customize the relative sizes of viewports in your plots. To do this:

1. From the **Edit** menu of the **Configure Plot Layout editor**, click **Set Viewport Weights**. The **Set Plot Viewport Weights** dialog box will open.
2. In the **Set Plot Viewport Weights** dialog box (Figure 6.51); you can specify the relative size of each viewport as a percentage value, with all of the weights adding up to 100%. Two viewports of equal weight at 50% each is shown in Figure 6.51.



Viewport	Weight(%)
Viewport (1,1) (50%)	50.0
Viewport (2,1) (50%)	50.0

Figure 6.51 Set Plot Viewport Weights Dialog Box

Click **Apply** to save and view your changes without closing the dialog box or click **OK** save your changes and close the dialog box.

6.11.3 Adding and Removing Axes

By default, viewports have a left Y-axis. You can add a right Y-axis, remove both the left and right Y-axes, and add a new left Y-axis if you have previously removed it. Viewports can have a maximum of two axes. You cannot remove an axis when a data set is associated with it.

To add an axis to a viewport, from the **Configure Plot Editor** you can either:

- Click on the viewport's name in the tree in the **Selected Data Sets** box. From the **Edit** menu, click **Add Axis**.

-Or-

- Right-click on the viewports name in the tree in the **Selected Data Sets**. From the shortcut menu (see Figure 6.50, page 6-38), click **Add Axis**.

The tree now displays another axis for the viewport you selected.

To remove an axis that has no data associated with it, you can either:

- Click on the name of the axis you wish to remove in the tree in the Selected Data Sets box. From the **Edit** menu, click **Remove Axis**.

-Or-

- Right-click on the axis you wish to remove in the tree in the Selected Data Sets box. From the shortcut menu (see Figure 6.50, page 6-38), click **Remove**.

6.11.4 Arranging Viewports and Axes

You can rearrange the vertical order of viewports in a plot window and move axes (with their associated data sets) to different viewports.

To move a viewport or axis:

1. From the tree in the **Selected Data Sets** box, right-click on the name of a viewport or axis you want to move.
2. From the shortcut menu, click **Move Up** or **Move Down**.
3. Either click **OK** or **Apply** for the change to take effect in the plot window.

Note that you cannot move individual data sets in the **Configure Plot Layout** editor. You can move only the axis with which a data set is associated.

6.11.5 Reversing Axes (Invert Data)

To reverse the direction of an axis so that the data is inverted, in the **Configure Plot Layout** editor, right-click on the name of the axis in the tree. From the shortcut menu, click **Reverse**. Either click **OK** or **Apply** for the change to appear in the plot window.

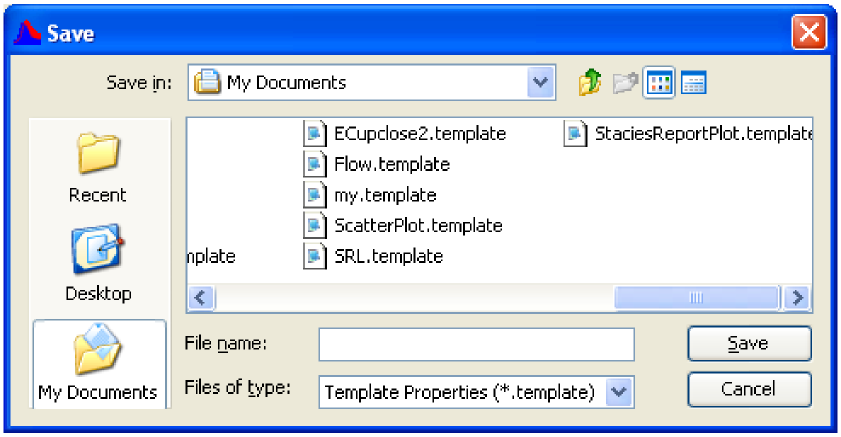
6.12 Saving and Applying Templates

After you have customized a plot, you can save its settings as a template for use in other plots.

Generally, you will use templates when scripting plots. For example: generating a plot of flow, stage, and precipitation via a script every day, and then apply a template that has all of the correct formatting, such as viewport placement, size, line colors, and fills. For more information about using templates with scripts, refer to Chapter 8 on **Scripting**.

To create a template from a plot:

1. From the **File** menu, click **Save Template**.
2. A **Save** browser will open (Figure 6.52).


Save in: My Documents

nplate
ECupclose2.template
Flow.template
my.template
ScatterPlot.template
SRL.template
StaciesReportPlot.template

File name:

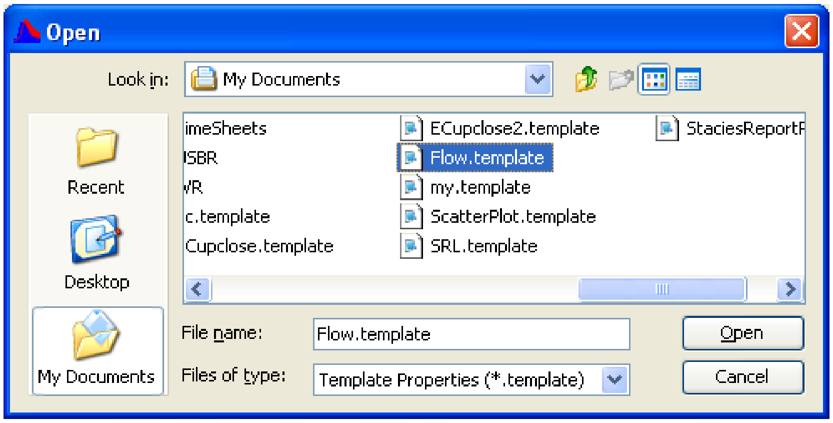
Files of type: Template Properties (*.template)

Figure 6.52 Save Browser

3. Give the template a name in the **File name** field and click the **Save** button. A **.template* file will be saved.

To apply a template created from a previous plot:

1. From the **File** menu, click **Apply Template**.
2. An **Open** browser will open (see Figure 6.53, page 6-41).
3. Move to the **.template* file's location and select it by clicking on it.



File name: Flow.template

Files of type: Template Properties (*.template)

Figure 6.53 Open Browser

4. When you select a template, its name will display in the **File Name** field.
5. Click **Open** to apply the template to the current plot.

6.13 Additional Viewing Options for Plots

The **File** menu of plots (Figure 6.54) contains several commands that allow you to view plot data in tabular form, save, copy, paste, and print plots. Copy can be used to copy the plot into other applications available on your computer.

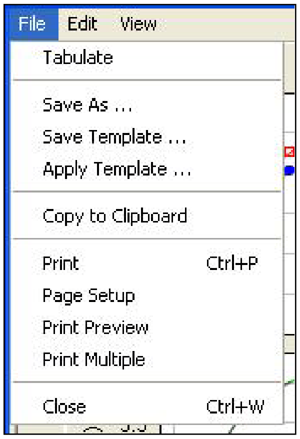
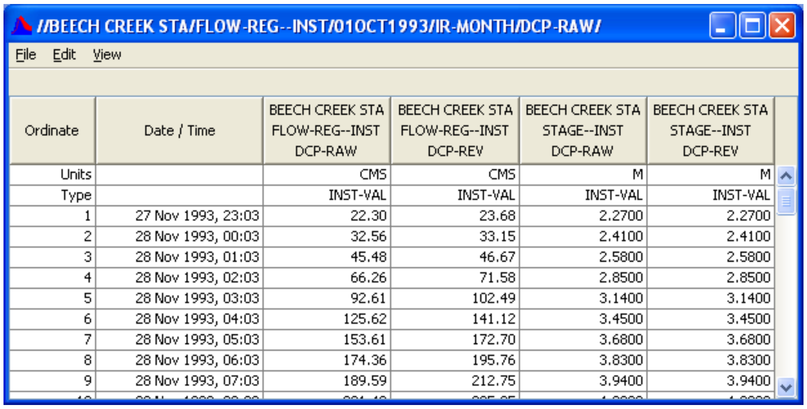


Figure 6.54 File Menu

6.13.1 Viewing Data in Tabular Form

To view plot data in tabular form, from the **File** menu, click **Tabulate**. A dialog box will open displaying the data in tabular form (Figure 6.55). For more information about tables, refer to Chapter 4 on **Utilities**.



//BEECH CREEK STA/FLOW-REG--INST/01OCT1993/IR-MONTH/DCP-RAW/

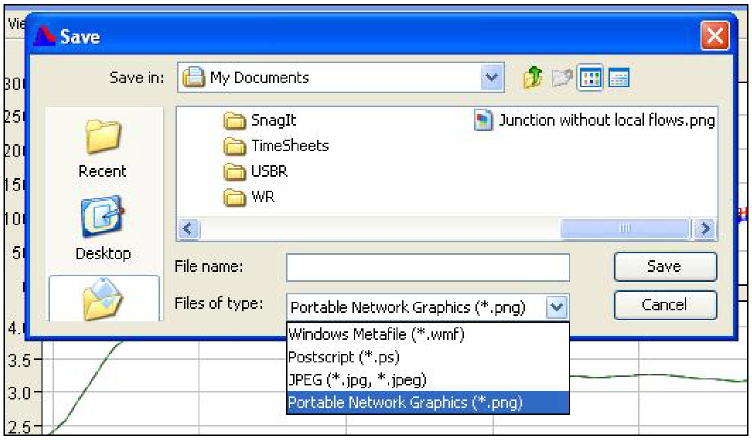
Ordinate	Date / Time	BEECH CREEK STA FLOW-REG--INST DCP-RAW	BEECH CREEK STA FLOW-REG--INST DCP-REV	BEECH CREEK STA STAGE--INST DCP-RAW	BEECH CREEK STA STAGE--INST DCP-REV
Units		CMS	CMS	M	M
Type		INST-VAL	INST-VAL	INST-VAL	INST-VAL
1	27 Nov 1993, 23:03	22.30	23.68	2.2700	2.2700
2	28 Nov 1993, 00:03	32.56	33.15	2.4100	2.4100
3	28 Nov 1993, 01:03	45.48	46.67	2.5800	2.5800
4	28 Nov 1993, 02:03	66.26	71.58	2.8500	2.8500
5	28 Nov 1993, 03:03	92.61	102.49	3.1400	3.1400
6	28 Nov 1993, 04:03	125.62	141.12	3.4500	3.4500
7	28 Nov 1993, 05:03	153.61	172.70	3.6800	3.6800
8	28 Nov 1993, 06:03	174.36	195.76	3.8300	3.8300
9	28 Nov 1993, 07:03	189.59	212.75	3.9400	3.9400

Figure 6.55 Data in Tabular Form

6.13.2 Saving Plots

You can save a plot as an image as a Windows Metafile (*.wmf), Postscript (*.*ps*), JPEG (*.*jpg*, *.*jpeg*), or Portable Network Graphics (*.*png*).

To save a plot as a graphic file, from the **File** menu, click **Save As**. From the **Save** browser (Figure 6.56), select the location where you want to save the plot, enter a filename in the **File name** box, and select the file type you want from the **Files of type** list, then click **Save**.



Save in: My Documents

SnagIt
TimeSheets
USBR
WR
Junction without local flows.png

File name:

Files of type: Portable Network Graphics (*.png)

Windows Metafile (*.wmf)
Postscript (*.ps)
JPEG (*.jpg, *.jpeg)
Portable Network Graphics (*.png)

Figure 6.56 Save Browser - Plots

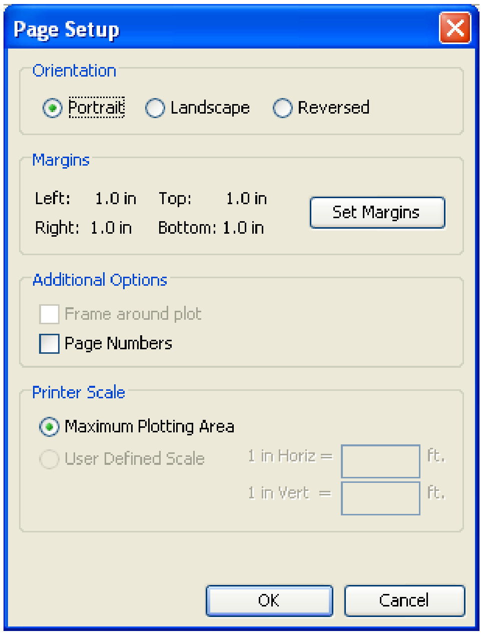
6.13.3 Copying Plots to the Clipboard

Use **Copy to Clipboard** from the **File** menu to copy a plot to the clipboard. You can then paste the plot as an image into other applications that are available on your computer.

6.14 Printing Plots

The **Print** command, available from the **File** menu of a plot, opens a standard, Windows-style print dialog box.

From the **File** menu, click **Page Setup**. The **Page Setup** dialog box will open (Figure 6.57). Here you



Orientation
● Portrait ○ Landscape ○ Reversed

Margins
Left: 1.0 in Top: 1.0 in
Right: 1.0 in Bottom: 1.0 in

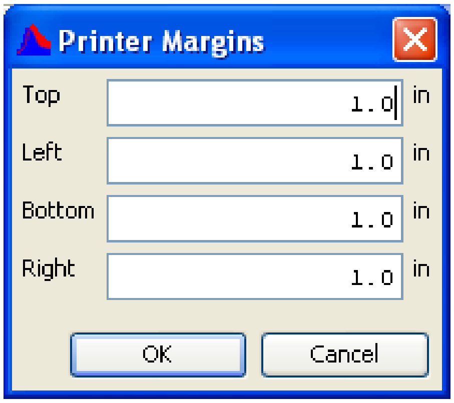
Additional Options
□ Frame around plot
□ Page Numbers

Printer Scale
● Maximum Plotting Area
○ User Defined Scale 1 in Horiz = ft.
1 in Vert = ft.

Figure 6.57 Page Setup Dialog Box

can set the page **Orientation**, **Margins**, **Page Numbers**, and **Printer Scale**. **Set Margins** opens the **Printer Margins** dialog box (Figure 6.58).

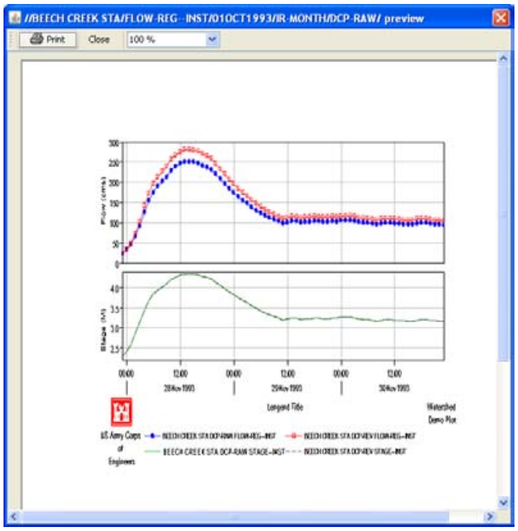
To view the plot as it will be printed, from the **File** menu, click **Print Preview**. An example is shown in Figure 6.59.



Top 1.0 in
Left 1.0 in
Bottom 1.0 in
Right 1.0 in

Figure 6.58 Printer Margins Dialog Box

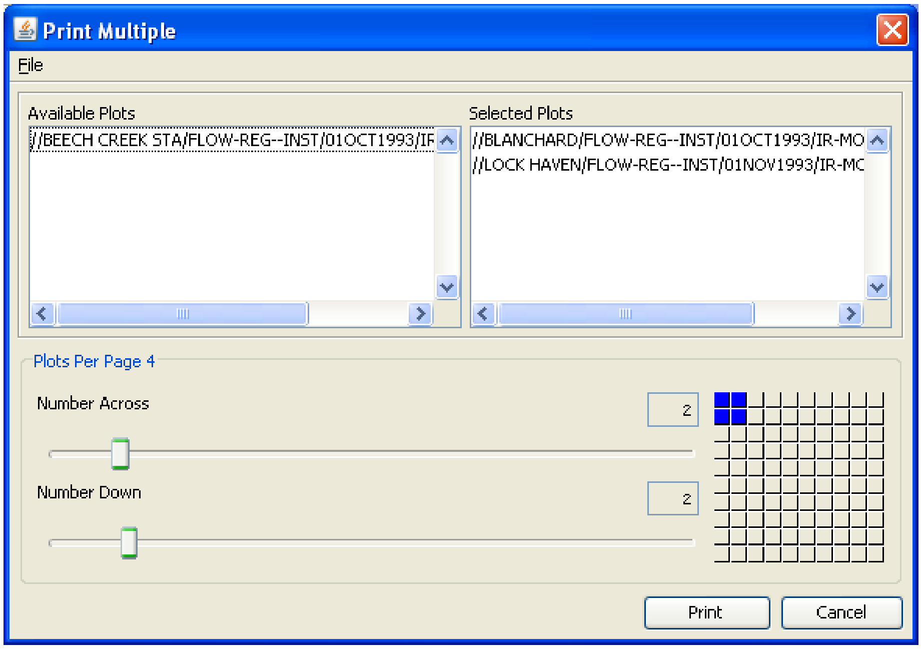
Finally, the **Print Multiple** command allows you to print several plots on one page. The **Print Multiple** dialog box (Figure 6.60) shows all of the currently open plots in the **Available Plots** box. To select plots for printing, double-click on the plots in the **Available Plots** box and the selected plots will move to the **Selected Plots** box. Next, use the slider bars to specify the number of plots you wish to appear horizontally and vertically on the page. The grid to the right of the sliders reflects your choices.



//BEECH CREEK STA/FLOW-REG--INST/01OCT1993/IR-MONTH/DCP-RAW/ preview

100 %

Figure 6.59 Example Print Preview of a Plot



Available Plots
//BEECH CREEK STA/FLOW-REG--INST/01OCT1993/IF

Selected Plots
//BLANCHARD/FLOW-REG--INST/01OCT1993/IR-MO
//LOCK HAVEN/FLOW-REG--INST/01NOV1993/IR-MC

Plots Per Page 4

Number Across 2

Number Down 2

Figure 6.60 Print Multiple Dialog Box

A preview of the plots set up above is shown in Figure 6.61. From the **Print Multiple** dialog box, you can also use the **Page Setup** and **Print Preview** commands. The two commands are available from the **File** menu of the **Print Multiple** dialog box.

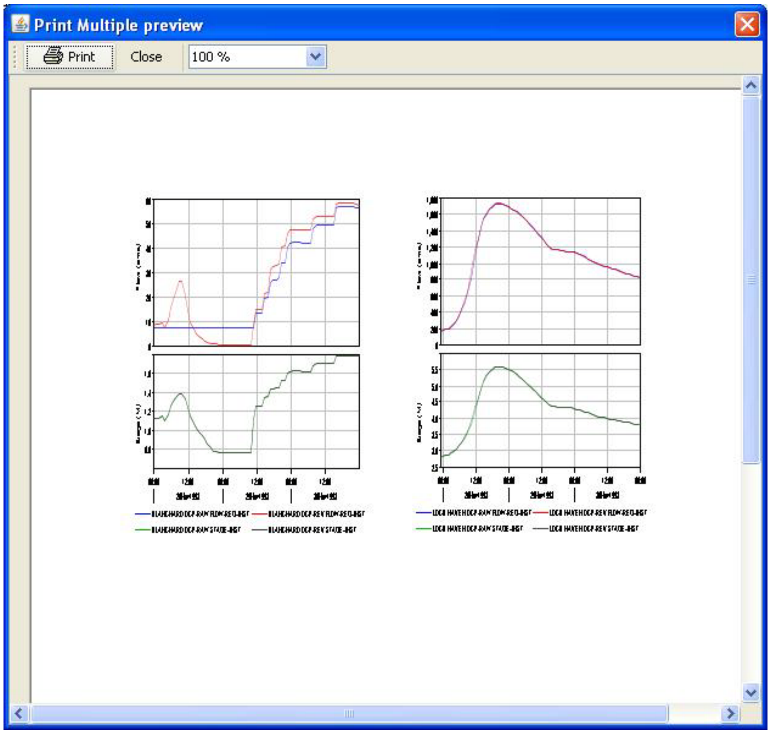


Figure 6.61 Print Multiple Preview Dialog Box (Example)

6.15 Importing and Exporting Default Plot Properties

Plot property changes made in a plot's **Default Plot Properties** editor apply to all plots used within the **HEC-DSSVue** application. However, if you would like to use the same defaults for another HEC application, you can import or export the properties. The properties must be moved from/to a compatible HEC "Next Generation" java based software application.

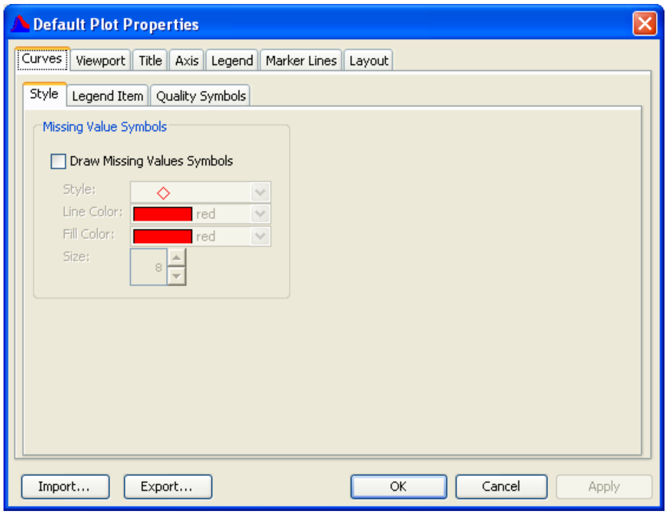
Examples of compatible software are: HEC-DSSVue, HEC-FIA, HEC-ResSim, HEC-RAS, HEC-HMS, CWMS, etc.

6.15.1 Exporting Default Plot Properties

Before you can import default plot properties, the properties must first by exported. By exporting the properties, you create a **.zip* file of all the necessary default plot property files needed.

To **Export** default plot properties:

1. In a plot window from the **Edit** menu, click **Default Plot Properties**. In the **Default Plot Properties** dialog box, click **Export** (Figure 6.62).



☐ Draw Missing Values Symbols

Style: ◇

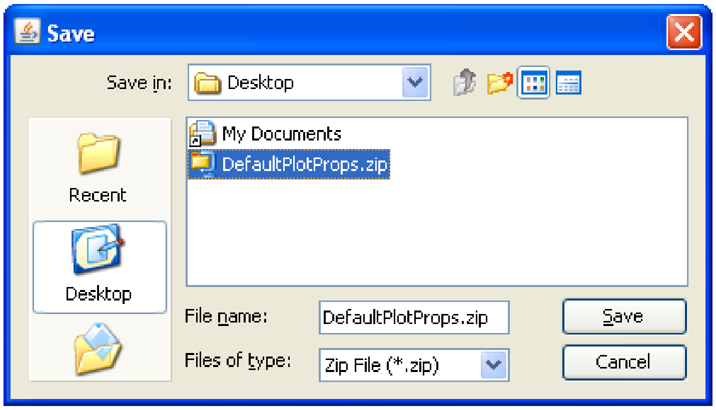
Line Color: 🟥 red

Fill Color: 🟥 red

Size: 8

Figure 6.62 Default Plot Properties Dialog Box

2. A **Save** browser will open (Figure 6.63). In the **Save** browser, type in a name for your default plot properties in the **File name** field, then click **Save**.



Save in: Desktop

File name: DefaultPlotProps.zip

Files of type: Zip File (*.zip)

Figure 6.63 Save Browser - Exporting Plot Properties

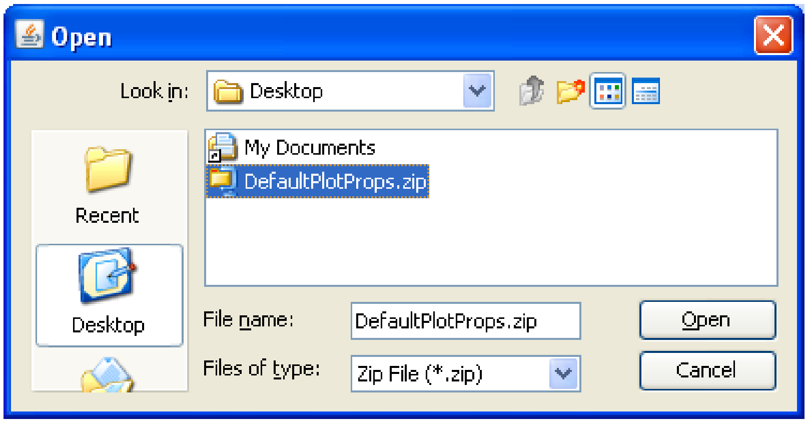
3. A **.zip* file will be created containing all the files needed to import the properties to a different HEC application.

6.15.2 Importing Default Plot Properties

Once you have exported the default plot properties from a compatible HEC application, you can then **Import** them to a different HEC application.

To **Import** default plot properties:

1. In a plot window from the **Edit** menu, click **Default Plot Properties**. In the **Default Plot Properties** dialog box, click **Import** (see Figure 6.62, page 6-45).
2. An **Open** browser will open (Figure 6.64). Using the **Open** browser, select the **.zip* default plot properties file, and then click **Open**.



Look in: Desktop

File name: DefaultPlotProps.zip

Files of type: Zip File (*.zip)

Figure 6.64 Open Browser - Importing Plot Properties

3. The default properties in the **.zip* file will be imported and extracted into the current application. The changes will be reflected in the **Default Plot Properties** editor.

6.16 Text Substitution

In all locations where the plot properties editors allow you to enter text, for example: the **Plot Title**, **Legend Title**, **Axis Labels**, **Legend Labels**, etc. you can use text substitution. Text substitution is when you set the text using predefined keywords based on Date/Time and DSS related information. These keywords will be used as a look-up and replaced when the text is used in the plot window according to the selected DSS data.

In order for Keywords to be recognized as such, they must be bracketed on each side by % characters. For example: %BPART% would result in the "BPART" of the DSS path being written in the final display text. If you would like to have the actual percentage sign in the display text, use two % characters in a row %% and the second one will be displayed.

By default, the first DSS path provided in the plot will be used for the text substitution. However, adding a colon and integer value to the substitution keyword will instruct the substitution algorithm to use a different path. For example %BPART:3% would show the "BPART" of the third DSS path provided in the plot. If there were less than three paths proved in the plot, the last DSS path listed would be used for the substitution.

Text substitution keywords may be mixed with fixed text.

For example:
The text: Final Results at %BPART%
Would display: Final Results for SHASTA
Using the DSS path: //SHASTA/ELEV/1HOUR/01JAN2000/ALT1/

The following is a list of available text substitution keywords:

DSS Path Parts:

APART	DSS path A part
BPART	DSS path B part
CPART	DSS path C part
DPART	DSS path D part
EPART	DSS path E part
FPART	DSS path F part
DSSPATH	full DSS path

DSS File:

DSSFILENAME	DSS file name only (without *.dss* extension)
DSSFILEPATH	Full path to DSS file, including file name and extension
DSSFILEDIR	Directory containing DSS file

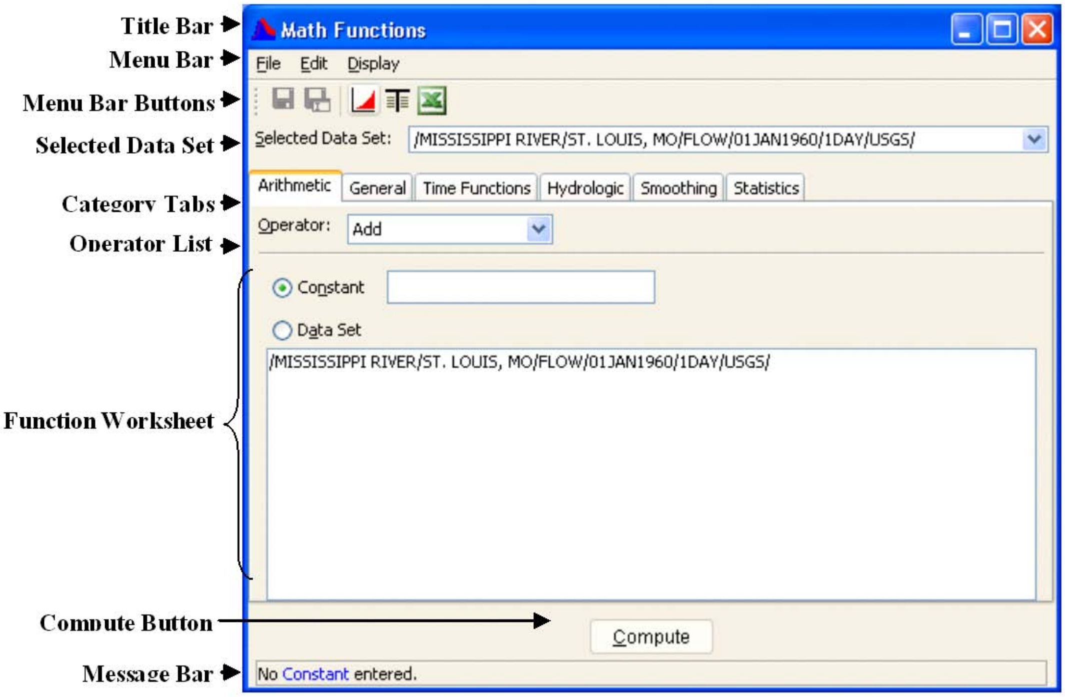
Current Date/Time:

CURDATE	Current Date (DDMMMYYYY)
CURMON	Current Month
CURDAY	Current Day
CURYEAR	Current Year
CURTIME	Current Time

CHAPTER 7

Math Functions

The Math functions are available from the **Tools** menu or fx Math Functions Toolbar button and are organized into six categories: **Arithmetic**, **General**, **Time Functions**, **Hydrologic**, **Smoothing**, and **Statistics**. Each category is a tab on the **Math Functions** screen (Figure 7.1). This chapter describes math functions according to these categories.



Selected Data Set: /MISSISSIPPI RIVER/ST. LOUIS, MO/FLOW/01JAN1960/1DAY/USGS/

Operator: Add

● Constant

○ Data Set

/MISSISSIPPI RIVER/ST. LOUIS, MO/FLOW/01JAN1960/1DAY/USGS/

No Constant entered.

Figure 7.1 Math Functions Screen

7.1 Math Functions Screen

Figure 7.1 shows the basic appearance of the **Math Functions** screen. The HEC-DSSVue Math Functions screen enables the mathematical manipulation of time series and paired data selected in the HEC-DSSVue Data Selection List window.

7.1.1 Menu Bar

Menu options in the Math Functions screen allow you to save or rename data computed with the Math functions, and to display the computed and original data in plots and tables. The Math Functions menus are as follows:

File **File** menu commands are **Save, Save As**, and **Close.**

Edit The **Edit** menu contains the **Restore Original Data** command.

Display Use the **Display** menu to open plots and tables with the **Plot** and **Tabulate** commands, and set options for data to display with the **Original Data with computed** and **All Data Sets** commands.

7.1.2 Menu Bar Buttons

Menu bar buttons provide shortcuts to frequently used Menu commands:

Saves computed data to a HEC-DSS file (same as **Save** in the **File** menu).

Saves and Renames computed data to a HEC-DSS file (same as **Save As** in the **File** menu).

Displays data as a Plot (same as **Plot** in the **Display** menu)

Displays data in Tabular form (same as **Tabulate** in the **Display** menu).

Displays data in Microsoft Excel (same as **Excel** in the **Display** menu).

7.1.3 Other Math Features of the Math Functions Screen

The **Operator** and **Selected Data Set** items appear for all function types.

Use the **Category Tabs** to access each of the six categories of math functions.

To select a function, use the **Operator** list.

Use the **Selected Data Set** list to choose a data set to apply each function. This list contains the names of data sets you have chosen in the HEC-DSSVue Data Selection List window.

Use the **Compute** button located near the bottom of the screen to apply a function to selected data sets. If data on the function screen is incomplete, the Compute button is unavailable. A message appears in the **Message Bar** at the bottom of the screen indicating which box is incomplete. In

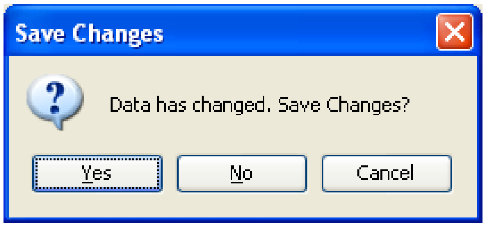
Figure 7.1, for example, the message indicates that no constant has been entered.

7.2 Managing Data

Most functions modify the values in the selected data. However, the **Math Functions** screen retains a copy of the original data, which you may use for comparison plotting with the computed result or for "undoing" the compute action. Other functions, such as "Merge Time Series", may generate a new data set, which is appended to the data set list in the **Selected Data Set** list.

Once a data set has been modified or generated by a function compute, you can save the data to the file, plot it, or tabulate it using the menu options or menu bar buttons located at the top of the **Math Functions** screen. A computed data set is not automatically saved to the file until you explicitly save the operation.

When you exit the **Math Functions** screen, if one or more data sets have been modified but not saved, you will be prompted with the **Save Changes** message shown in Figure 7.2. Save the changes by clicking **Yes**.


Save Changes

Data has changed. Save Changes?

Figure 7.2 Save Changes Dialog Box

In plots and tables, the F-Part of the pathname is identified as "MODIFIED-". This is not retained when the data set is saved to the file and saving the resulting data will overwrite records in the file. In Figure 7.3 (page 7-4), the "NATRONA" time series was added to the "BRADDOCK" time series. The resulting data set is "NATRONA MODIFIED".

7.2.1 Selecting Paired Data Curves for Function Operations

Typically paired data has a set of x-values and a corresponding set of y-values (e.g. a stage-flow rating table). However, a paired data set may have multiple sets of y-values, or curves, which share the same x-ordinates (e.g. frequency-damage curves for multiple categories). Math functions that operate on paired data sets allow the user to select one or all of the paired data curves for the function application. For example, with the Add function, a number may be added to y-values in one curve or to y-values in all curves.

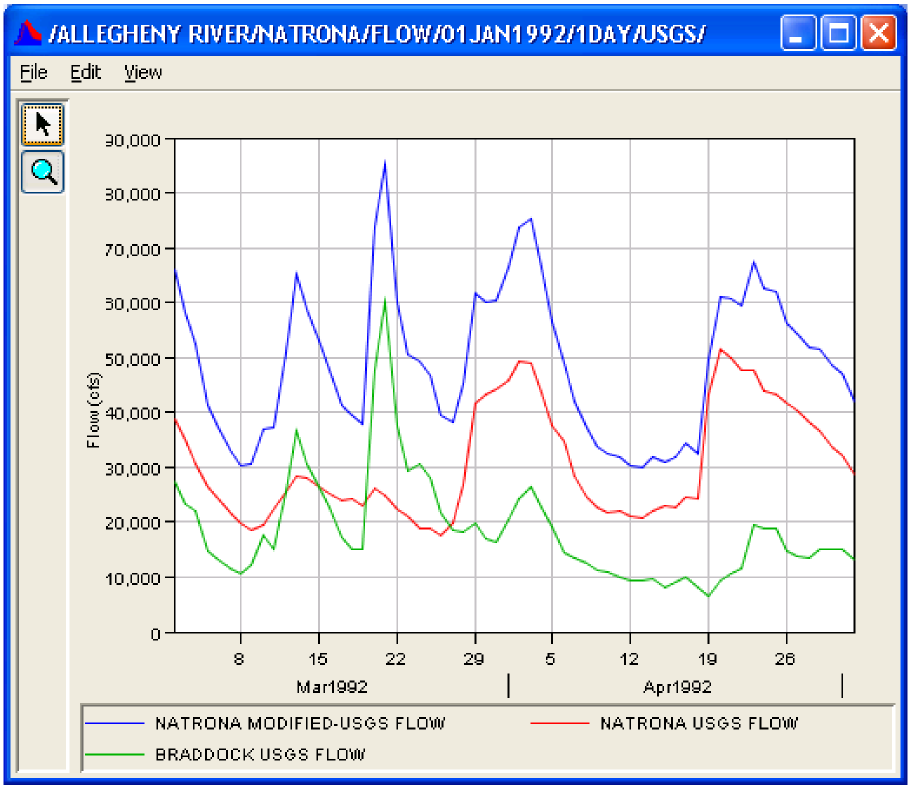
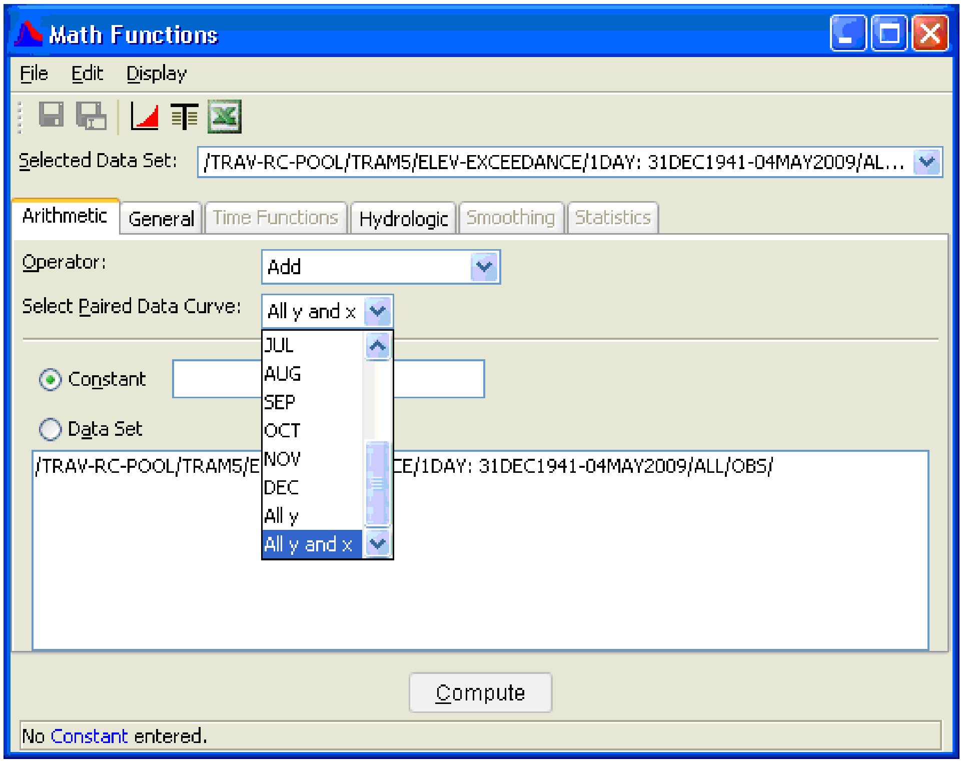


Figure 7.3 Plot of Time Series Data from the "Add" Function

Figure 7.4 shows the appearance of the Math Functions screen when a paired data set is selected for an "Add" operation.



Selected Data Set: /TRAV-RC-POOL/TRAM5/ELEV-EXCEEDANCE/1DAY: 31DEC1941-04MAY2009/AL...

Operator: Add

Select Paired Data Curve: All y and x

● Constant

○ Data Set

/TRAV-RC-POOL/TRAM5/E...E/1DAY: 31DEC1941-04MAY2009/ALL/OBS/

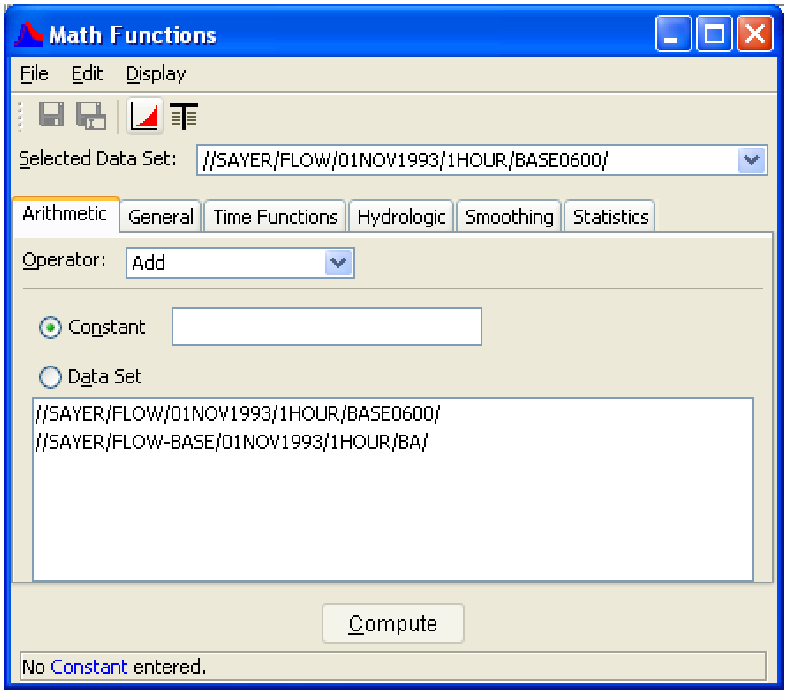
No Constant entered.

Figure 7.4 Selection of Paired Data Curve

The paired data curve to apply the add function is picked using the **Select Paired Data Curve** list. The list is filled with the paired data curve labels. In this case, there is a curve for each month of the year. The "All" selection in the list refers to the option to pick all curves for the function operation. If the paired data curves are not explicitly labeled in the paired data set, the curve list is then filled with the curve numbers. The **Select Paired Data Curve** list only appears if the selected data set is paired data.

7.3 Arithmetic Functions

The **Arithmetic** tab (Figure 7.5) contains the following functions: **Add**, **Subtract**, **Multiply**, **Divide**, **Exponentiation**, **Absolute Value**, **Square Root**, **Log**, **Log Base 10**, **Sine**, **Cosine**, **Tangent**, **Inverse**, **Accumulation**, **Successive Differences**, and **Time Derivative**.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Add

● Constant

○ Data Set

//SAYER/FLOW/01NOV1993/1HOUR/BASE0600/
//SAYER/FLOW-BASE/01NOV1993/1HOUR/BA/

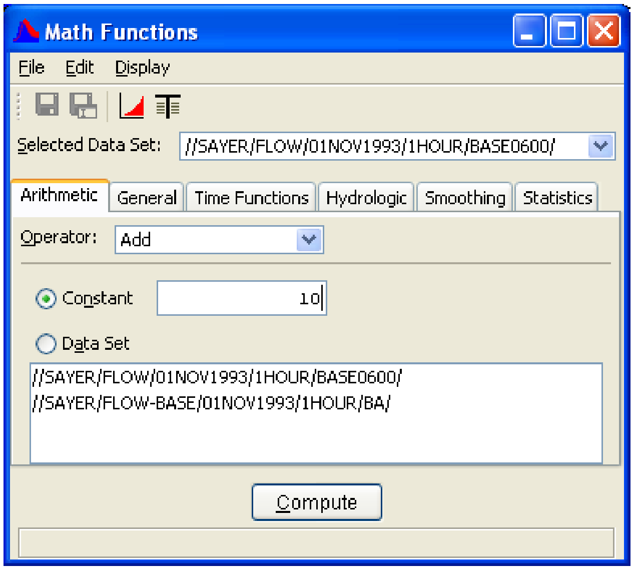
No Constant entered.

Figure 7.5 Math Functions - Arithmetic Tab

7.3.1 Add

To **Add** a value to each data point in a data set:

1. Select **Add** in the **Operator** dropdown list.
2. Click **Constant** and enter a value to add in the box to the right, as shown in Figure 7.6 (page 7-6).
3. Click **Compute**.


Math Functions
File Edit Display
Arithmetic General Time Functions Hydrologic Smoothing Statistics
Compute

Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Add

● Constant 10

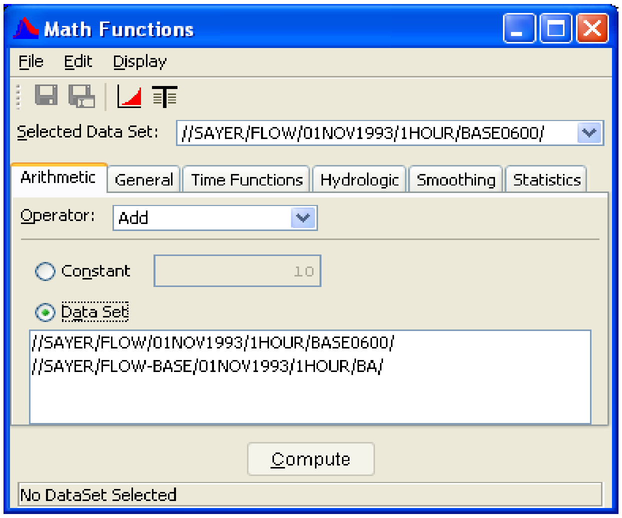
○ Data Set

//SAYER/FLOW/01NOV1993/1HOUR/BASE0600/
//SAYER/FLOW-BASE/01NOV1993/1HOUR/BA/

Figure 7.6 Adding a Constant to a Data Set

To **Add** data sets together (time series data sets only):

1. Select **Add** in the **Operator** dropdown list.
2. From the **Selected Data Set** list, select a data set to apply the function. This data set will contain the result of the addition operation.
3. Click **Data Set**.
4. From the **Data Set** list, select the data sets to be added. If you include the data set selected in the top **Selected Data Set**, it will be added by itself. See Figure 7.7 for an example.
5. Click **Compute**.


Math Functions
File Edit Display
Arithmetic General Time Functions Hydrologic Smoothing Statistics
Compute

Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Add

○ Constant 10

● Data Set

//SAYER/FLOW/01NOV1993/1HOUR/BASE0600/
//SAYER/FLOW-BASE/01NOV1993/1HOUR/BA/

No DataSet Selected

Figure 7.7 Add Data Sets Together

Caution: Do not include the **Selected Data Set** (at the top of the dialog) in the lower window unless you want the data set added to itself. This is a common mistake.

7.3.2 Subtract

To **Subtract** a value from each data point in a data set:

1. Select **Subtract** in the **Operator** dropdown list.
2. Click **Constant** and enter a value to subtract in the box to the right.
3. Click **Compute**.

To **Subtract** a data set from anther data set (time series data sets only):

1. From the **Selected Data Set** list, select a data set to apply the function. This data set will contain the result of the subtraction operation.
2. Click **Data Set**.
3. From the **Data Set** list, select the data sets to be subtracted. If you include the data set selected in the **Selected Data Set**, it will be subtracted from itself.
4. Click **Compute**.

7.3.3 Multiply

To **Multiply** a value to each data point in a data set:

1. Select **Multiply** in the **Operator** dropdown list.
2. Click **Constant** and enter a value to multiply by in the box to the right.
3. Click **Compute**.

To **Multiply** data sets (time series data sets only):

1. From the **Selected Data Set** list, select a data set to apply the function. This data set will contain the result of the multiply operation.
2. Click **Data Set**.
3. From the **Data Set** list, select the data sets to be multiplied. If you include the data set selected in the top **Selected Data Set**, it will be multiplied by itself.
4. Click **Compute**.

7.3.4 Divide

The **Divide** function divides all valid values in a time series or paired data set by a number, or divides the values in a data set by the values in one or more data sets of the same type. For time series data, missing values are kept as missing.

When you are dividing data sets, times in the data sets need not match exactly. Therefore, only values with coincident times will be divided. Times in the primary time series data set that cannot be matched to times in the other data set result in missing values for those times. Values in the data set(s) selected from the lower list form the divisor(s). Values in the resultant data set are set to missing if there is a zero divisor. Missing values in the primary data set are kept as missing. Data sets may be regular or irregular interval time series.
Presently the Divide function does not allow the division of paired data by another data set.

To divide all values in the selected data set(s) by a number:

1. Choose the **Arithmetic** tab of the Math Functions Screen and select the **Divide** operator.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. If the data set is paired data, use the **Select Paired Data Curve** list to select a single paired data curve or all curves for the operation (see Section 7.2.1 for more details). Constants can be divided by only one paired data set at a time.
4. Click **Constant** and enter the value to divide by in the box to the right.
5. Click **Compute**.

To divide the selected data set by one or more data sets (time series only):

1. From the **Selected Data Set** list, select a data set to apply the function. This data set will contain the result of the divide operation.
2. Click **Data Set**.
3. From the **Data Set** list, select the divisor data set(s). . If you include the data set selected in the top **Selected Data Set**, it will be divided by itself.
4. Click **Compute**.

7.3.5 Exponentiation

The **Exponentiation** function raises values in a time series or paired data set to a user specified power, or exponent. For time series data, missing values are kept as missing.

For exponentiation of the values:

1. Choose the **Arithmetic** tab of the Math Functions Screen and select the **Exponentiation** operator.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. If the data set is paired data, use the **Select Paired Data Curve** list to select a single paired data curve or all curves for the operation (see Section 7.2.1 for more details).
4. In the **Power** box, enter the value for the power, or exponent.
5. Click **Compute**.

7.3.6 Absolute Value

The **Absolute Value** function computes the absolute value of values in a time series or paired data set. For time series data, missing values are kept as missing.

To compute the absolute value of values:

1. Choose the **Arithmetic** tab of the Math Functions Screen and select the **Absolute Value** operator.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. If the data set is paired data, use the **Select Paired Data Curve** list to select a single paired data curve or all curves for the operation (see Section 7.2.1 for more details).
4. Click **Compute**.

7.3.7 Square Root

The **Square Root** function computes the square root of valid values in a time series or paired data set. If a value is less than 0.0, the value is set to missing in the resultant data set. For time series data, if the original value is missing, the value remains missing in the resultant data set.

To compute the square root of values:

1. Choose the **Arithmetic** tab of the Math Functions Screen and select the **Square Root** operator.

2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. If the data set is paired data, use the **Select Paired Data Curve** list to select a single paired data curve or all curves for the operation (see Section 7.2.1 for more details).
4. Click **Compute**.

7.3.8 Log

The **Log** function computes the natural logarithm (log base "e") of valid values in a time series or paired data set. If a value is less than or equal to 0.0, the value is set to missing in the resultant data set. For time series data, if the original value is missing, the value remains missing in the resultant data set.

To compute the natural logarithm of values:

1. Choose the **Arithmetic** tab of the Math Functions Screen and select the **Log** operator.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. If the data set is paired data, use the **Select Paired Data Curve** list to select a single paired data curve or all curves for the operation (see Section 7.2.1 for more details).
4. Click **Compute**.

7.3.9 Log Base 10

The **Log Base 10** function computes the log base 10 of valid values in a time series or paired data set. If a value is less than or equal to 0.0, the value is set to missing in the resultant data set. For time series data, if the original value is missing, the value remains missing in the resultant data set.

To compute the log base 10 of values:

1. Choose the **Arithmetic** tab of the Math Functions Screen and select the **Log Base 10** operator.

2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. If the data set is paired data, use the **Select Paired Data Curve** list to select a single paired data curve or all curves for the operation (see Section 7.2.1 for more details).
4. Click **Compute**.

7.3.10 Sine

The **Sine** function computes the sine of valid values in a time series or paired data set. The resulting data set will be in radians. For time series data, missing values are kept as missing.

To compute the sine of values:

1. Choose the **Arithmetic** tab of the Math Functions Screen and select the **Sine** operator.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. If the data set is paired data, use the **Select Paired Data Curve** list to select a single paired data curve or all curves for the operation (see Section 7.2.1 for more details).
4. Click **Compute**.

7.3.11 Cosine

The **Cosine** function computes the cosine of valid values in a time series or paired data set. The resulting data set will be in radians. For time series data, missing values are kept as missing.

To compute the cosine of values:

1. Choose the **Arithmetic** tab of the Math Functions Screen and select the **Cosine** operator.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.

3. If the data set is paired data, use the **Select Paired Data Curve** list to select a single paired data curve or all curves for the operation (see Section 7.2.1 for more details).
4. Click **Compute**.

7.3.12 Tangent

The **Tangent** function computes the tangent of valid values in a time series or paired data set. The resulting data set will be in radians. If the cosine of a value is 0.0, the value is set to missing in the resultant data set. For time series data, if the original value is missing, the value remains missing in the resultant data set.

To compute the tangent of values:

1. Choose the **Arithmetic** tab of the Math Functions Screen and select the **Tangent** operator.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. If the data set is paired data, use the **Select Paired Data Curve** list to select a single paired data curve or all curves for the operation (see Section 7.2.1 for more details).
4. Click **Compute**.

7.3.13 Inverse

The **Inverse** function computes a new value from 1 divided by the value (1/x) in a time series or paired data set. If a value is equal to 0.0, the value is set to missing in the resultant data set. For time series data, if the original value is missing, the value remains missing in the resultant data set.

To compute the inverse (1/x) of values:

1. Choose the **Arithmetic** tab of the Math Functions Screen and select the **Inverse** operator.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.

3. If the data set is paired data, use the **Select Paired Data Curve** list to select a single paired data curve or all curves for the operation (see Section 7.2.1 for more details).
4. Click **Compute**.

7.3.14 Accumulation

The **Accumulation** function computes a running accumulation of values for a regular or irregular interval time series data set. For a missing value in the time series data, the value in the accumulation time series remains constant (i.e., missing values treated as zero).

Note: The data type of the time series data set governs how values are added. See also **Accumulation over interval** in the **Time Conversion** function **Min/Max/Avg/… Over Period**.

To compute the running accumulation of values for time series data sets:

1. Choose the **Arithmetic** tab of the Math Functions Screen and select the **Accumulation** operator.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. Click **Compute**.

7.3.15 Successive Differences

The **Successive Differences** function computes the difference between successive values in a regular or irregular interval time series data set. The time series data must be of type "INST-VAL" or "INST-CUM". A value in the resultant time series is set to missing if either the current or previous value in the original time series is missing (need to have two consecutive valid values). If the data type of the original data set is "INST-CUM" the resultant time series data set is assigned the type "PER--CUM", otherwise the data type does not change.

To compute the successive differences for time series data sets:

1. Choose the **Arithmetic** tab of the Math Functions Screen and select the **Successive Differences** operator.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set

selected in the top **Selected Data Set**, the operation on that set will only be done once.

3. Click **Compute**.

7.3.16 Time Derivative

The **Time Derivative** function computes the successive differences per unit time for a regular or irregular interval time series data set. For the time "t",

$$TS2(t) = (TS1(t) – TS1(t\text{-}1)) / DT$$

where DT is the time difference between t and t-1. For the current form of the function, the units of DT are minutes.

A value in the resultant time series is set to missing if either the current or previous value in the original time series is missing (need to have two consecutive valid values). By default, the data type of the resultant time series data set is assigned as "PER-AVER".

To compute the time derivative for time series data sets:

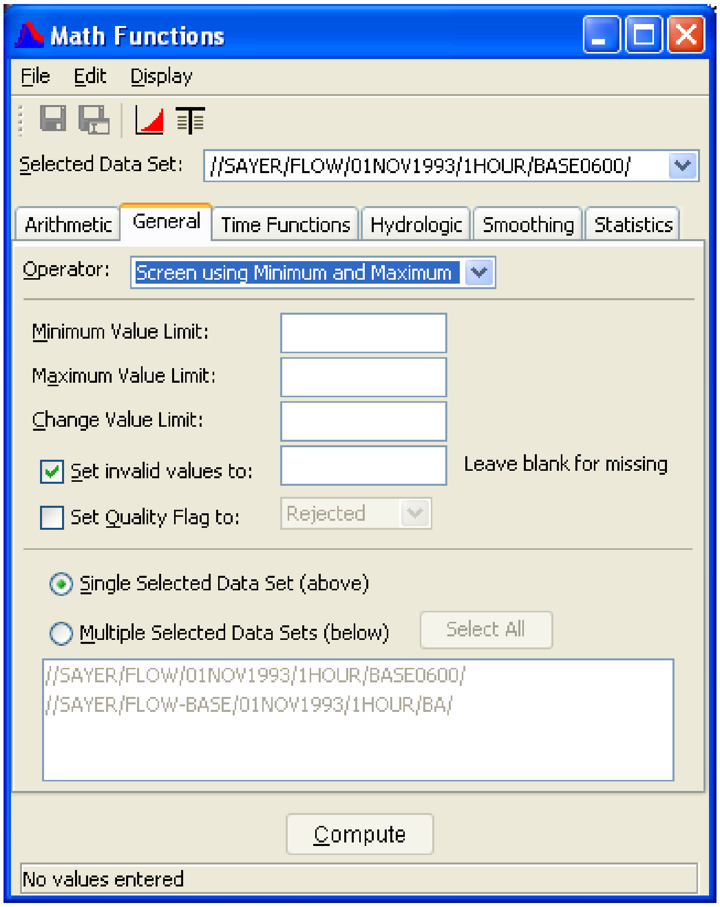
1. Choose the **Arithmetic** tab of the Math Functions Screen and select the **Time Derivative** operator.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. Click **Compute**.

7.4 General Functions

The **General** tab (see Figure 7.8, page 7-15) contains the following functions: **Units Conversion**, **Set Units**, **Set Type**, **Round to Nearest Whole Number**, **Truncate to Whole Number**, **Round Off**, **Estimate Missing Values**, **Replace Specific Values**, **Screen Using Maximum and Minimum**, **Screen Using Forward Moving Average**, **Merge Time Series**, **Merge Paired Data**, and **Generate Data Pairs**.

7.4.1 Units Conversion

The **Units Conversion** function converts SI (metric) unit data to English units, or English unit data to SI units. The Units Conversion function may be applied to either time series or paired data sets.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Screen using Minimum and Maximum

Minimum Value Limit:

Maximum Value Limit:

Change Value Limit:

☑ Set invalid values to: Leave blank for missing

□ Set Quality Flag to: Rejected

● Single Selected Data Set (above)

○ Multiple Selected Data Sets (below)

//SAYER/FLOW/01NOV1993/1HOUR/BASE0600/
//SAYER/FLOW-BASE/01NOV1993/1HOUR/BA/

No values entered

Figure 7.8 Math Functions - General Tab

To convert units for time series or paired data sets:

1. Choose the **General** tab of the Math Functions screen and select the **Time Units Conversion** operator.
2. Choose the type of conversion by clicking **Convert to SI** or **Convert to English**.
3. From the **Selected Data Set** list, select one or more data sets for units conversion.
4. From the **Available Data Sets To Convert** list you may select additional data sets for units conversion. This list is affected by the type of units conversion selected. If Convert to SI is selected, only data sets that are currently in English units will appear in the list. Similarly, if Convert to English is selected only data sets currently in SI units will appear.
5. Click **Compute** to perform the units conversion of the selected data sets.

7.4.2 Set Units

The **Set Units** function sets the units label in the selected data set. It will not convert the data to the new units; unless a conversion multiplier is used that correctly converts from the current units to the new units. The **Set Units** function is often used to change the units label in incorrectly labeled data sets.

To set units for time series or paired data sets:

1. Choose the **General** tab of the Math Functions screen and select the **Set Units** operator.
2. The current units are provided in the **Units** box. Enter the units label you want to set the data set to.
3. If you want to convert the data besides just setting the units label, enter the conversion multiplier.
4. From the **Selected Data Set** list, select one or more data sets.
5. Click **Compute**.
6. Save your data using the **Save** button.

7.4.3 Set Type

The **Set Type** function sets the type label in the selected data set. It will not convert the data to the new type; you can use an operator in the **Time Functions** tab for that, if appropriate. The **Set Type** function is often used to change the type label in incorrectly labeled data sets.

To set the type for time series data sets:

1. Choose the **General** tab of the Math Functions screen and select the **Set Type** operator.
2. From the **Selected Data Set** list, select one or more data sets.
3. The current units are provided in the **Current Type** label. Enter the type that you want to set the data set to. The type choices are:
    a. PER-AVER
    b. PER-CUM
    c. INST-VAL
    d. INST-CUM
4. Click **Compute**.
5. Save your data using the **Save** button.

7.4.4 Round to Nearest Whole Number

The **Round to Nearest Whole Number** function rounds values in a time series or paired data set to the nearest whole number.

The function rounds up the decimal portion of a number if equal to or greater than .5 and rounds down decimal values less than .5. For example:

10.5 is rounded to 11.
10.499 is rounded to 10.
-10.499 is rounded to -10.
-10.500 is rounded to -10.
-10.501 is rounded to -11.

The x-values in paired data sets are unaffected by the function, only the y-value data are rounded. For time series data sets, missing values are kept missing.

To round values in time series or paired data sets to the nearest whole number:

1. Choose the **General** tab of the Math Functions screen and select the **Round to Nearest Whole Number** operator.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. Click **Compute**.

7.4.5 Truncate to Whole Number

The **Truncate to Whole Number** function truncates values in a time series or paired data set to the nearest whole number. For example:

10.99 is truncated to 10.
10.499 is truncated to 10.
-10.001 is truncated to -10.
-10.999 is truncated to -10.

The x-values in paired data sets are unaffected by the function, only the y-value data are truncated. For time series data sets, missing values are kept missing.

To truncate values in time series or paired data sets to the nearest whole number:

1. Choose the **General** tab of the Math Functions screen and select the **Truncate to Whole Number** operator.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. Click **Compute**.

7.4.6 Round Off

The **Round Off** function rounds values in a time series or paired data set to a specified number of significant digits and/or power of tens place. For the power of tens place, -1 specifies rounding to one-tenth (0.1), while +2 rounds to the hundreds (100). For example, 1234.123456 will round to:

1230.0 for number of significant digits = 3, power of tens place = -1
1234.1 for number of significant digits = 6, power of tens place = -1
1234 for number of significant digits = 6, power of tens place = 0
1230 for number of significant digits = 6, power of tens place = 1

The x-values in paired data sets are unaffected by the function, only the y-value data are rounded. For time series data sets, missing values are kept missing.

To round values in time series or paired data sets:

1. Choose the **General** tab of the Math Functions screen and select the **Round Off** operator.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. In the **Number Significant Digits** box, set digits precision.
4. In the **Power of 10s Place** box, set the magnitude of 10 to which to round.
5. Click **Compute**.

7.4.7 Estimate Missing Values

The **Estimate Missing Values** function linearly interpolates estimates for missing values in a regular or irregular interval time series data set. Linear interpolation will occur for those portions of the time series data set where the number of consecutive missing values exceeds a specified user limit.

If the time series data set has type "INST-CUM", a special check box appears to optionally enable the following rules intended for cumulative precipitation:

- If the values bracketing the missing period are increasing with time, only interpolate if the number of successive missing values does not exceed the value of the user specified limit.

- If the values bracketing the missing period are decreasing with time, do not estimate any missing values.
- If the values bracketing the missing period are equal, then estimate any number of missing values.

To estimate for missing values in time series data sets:

1. Choose the **General** tab of the Math Functions screen and select the **Estimate Missing Values** operator.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. In the **Maximum Consecutive Number of Missing** box, enter a value to set the limit for consecutive missing values allowed for interpolation.
4. If the time series data set is of type cumulative precipitation, select **Interpolate Cumulative Precip** to apply the modified rules for the interpolation of missing precipitation data.
5. Click **Compute** to perform the linear interpolation fill of the missing values in the data set.

7.4.8 Replace Specific Values

The **Replace Specific Values** function replaces all occurrences of a specified value with another. For example, you can use this function to change all occurrences of "1000”" to "2000". If the data set contains precision information, then values within that precision limit will be replaced. For example, if the precision of the data set is "1" (number of digits to the right of the decimal), then a value of "12.3" will replace all values between "12.25" and "12.35". If no precision is set, the values have to be exact to be replaced (not necessarily what is shown in a tabulation). This function works on both time-series and paired data sets.

To replace specific values in time series or paired data sets:

1. Choose the **General** tab of the Math Functions screen and select the **Replace Specific Values** operator.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.

3. In the **Value to be replaced** box, enter the value in the data sets that you want to be replaced. If you want to replace missing values, leave this box empty.
4. In the **Value to replace with**, enter the new number that you want to replace the specified number with. If you want the value to be replaced to be set to missing, leave this box empty.
5. Click **Compute** to perform the operation.

7.4.9 Screen Using Minimum and Maximum

The **Screen Using Minimum and Maximum** function screens regular or irregular interval time series data sets for possible erroneous values based on user specified minimum-maximum value limits, and maximum absolute change. The maximum absolute change is tested only when the previous time series value is screened as valid. Only one test need to be specified.

Data values failing the screening test can be assigned a user specified quality flag and/or set to a specific value or to missing. The data sets may or may not contain prior quality flags. If the user specifies setting quality flags for screened data, they will be added to the resultant data sets if none already exists.

1. Choose the **General** tab of the Math Functions screen and select the **Screen Using Minimum and Maximum** operator.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. In the **Minimum Value Limit** and/or **Maximum Value Limit** boxes, enter the minimum and/or maximum valid value limits, respectively.
4. In the **Change Value Limit** box, enter a value to set maximum absolute change allowed from the previous time series value if you want to test for this.
5. Select the **Set Invalid values to** checkbox and fill enter the value you want to replace with for time series values failing the screening test. To set the value to missing, leave that box empty.
6. Select the **Set Quality Flag** box if the data quality flag is to be set for data values failing the screening test. If this box is checked, invalid data will be flagged with the quality selected in the list to the right. The available quality settings are: R(ejected), M(issing) and Q(uestionable).
7. Click **Compute**.

7.4.10 Screen with Forward Moving Average

The Screen with Forward Moving Average function screens a time series data set for possible erroneous values based on the change from the forward moving average computed at the previous point.

Data values failing the screening test are assigned a user specified quality flag and/or are set to the missing value. The data set may or may not currently have quality flags assigned. The forward moving average is computed over a user specified number of values. Missing values and values failing the screening test are not counted in the moving average and the divisor of the average is less one for each such value.

To screen data in time series data sets:

1. Choose the **General** tab of the Math Functions screen and select the **Screen with Forward Moving Average** operator.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. In the **Number to Average Over** box, enter a value to set the size of the moving average interval.
4. In the **Change Value Limit** box, enter a value to set maximum absolute change allowed from the forward moving average.
5. Select the **Set Invalid Values to Missing** box if time series values failing the screening test are to be set to the missing value.
6. Select the **Set Quality Flag** box if the data quality flag is to be set for data values failing the screening test. If this box is checked, invalid data will be flagged with the quality selected in the list to the right. The available quality settings are: R(ejected), M(issing) and Q(uestionable).
7. Click **Compute** to apply the function to the selected data set.

7.4.11 Paired Data Operations

The Paired Data Operations section provides a means of performing several utility operations on paired data curves. The operations include which parameter to show on the horizontal axis, an operation to reorder points in ascending order, re-sampling the curve to contain fewer points, and swapping the parameters. Generally, only one operation should be preformed at a time. However, if desired, the order of the operations is: 1) Reorder in ascending; 2) Delete Curve(s); 3) Re-sample; 4) Swap parameters, and 5) Setting the parameter associated with the horizontal axis.

1) Setting the parameter associated with the horizontal axis. By default, when a paired data curve is created, the first, or independent, parameter is displayed on the horizontal axis. This is the same as the first parameter in the C Part. For some plots it may be more desirable to have the independent parameter display on the vertical axis instead of the horizontal axis. This flag does not change how data is interpreted by programs; only how it is displayed in plots and tables.

2) Reorder points to be ascending and removing duplicates. This operation will take the points in the curve and reorder them so that the primary parameter is ascending. Any duplicate primary (independent) parameter values will be removed. The function will not remove dependent parameters. If you wish to accomplish this, you should swap the parameter sets, run this function, and then swap back.

3) Resample points. This operation reduces the number of points by resampling the curve according to the number specified to skip. If the value entered is 2, then every other point will be skipped. If the value entered is 3, then 2 values will be skipped and the third kept. If the value entered is one, no points will be skipped. The first and last points will always be retained.

4) Swap parameters. This operation moves the X (independent) ordinates to the Y (dependent) position and the Y ordinates to the X position. The units and type are moved also, as well as the names in the C part. This function is only valid for data sets with one curve (one dependent parameter).

5) Delete curve. This operation will delete the specified curves from a multi-curve data set. To select curves, press the down arrow on the combination box and click on the curves that are to be deleted. Selected curves will be shown with a checkbox at the beginning. This function is not valid with single curve data sets, nor can you delete all the curves.

Note: You can also delete curves, as well as insert them, from an edit-enabled table by selecting a column and then selecting Delete Column (or Insert Column) from the Edit menu.

To modify a paired data set with these functions:

1. Choose the **General** tab of the Math Functions screen and select the **Paired Data Operations** operator.
2. From the **Selected Data Set** list, select one or more data sets.

3. Select the operation(s) that you want to perform. It is recommended to select only one at a time. If you select resample, enter the number of points you want the new set to correspond to. If you want to delete a curve(s), select the curve(s) from the pull down box. For multiple selections, hold down the control key when you select the curve.
4. Click **Compute** to perform the operation(s).

7.4.12 Merge Time Series

The **Merge Time Series** function merges data from one time series data set with another time series data set. The resultant time series data set includes all the data points in the two time series, except where the data points occur at the same time. When data points from the two data sets are coincident in time, valid values in the primary time series take precedence over valid values in the second selected time series. However if a coincident point is missing in the primary time series, a valid value in the second time series will be used for the data point in the resultant data set. If the value is missing for both time series data sets, the value is missing in the resultant data set.

The data sets for merging may be either regular or irregular time interval. The resultant data set is tested to determine if the times have a regular time interval. If not, it is typed as an irregular time interval data set.

Data of any type (i.e., "INST-CUM") or units may be merged with data of any other type or units. The resultant time series receives the data type and units of the primary time series.

To merge two time series data sets:

1. Choose the **General** tab of the Math Functions screen and select the **Merge Time Series** operator.
2. From the **Selected Data Set** List, select the *primary* time series data set.
3. From the **Time Series** list, select the *second* time series data set for merging.
4. Click **Compute** to merge the two data sets.

7.4.13 Merge Paired Data

The **Merge Paired Data** function merges two paired data sets. The resultant paired data set includes all the paired data curves from the first data set and a single selected paired data curve or all curves from the second data set. The *x-values* for the two paired data sets *must match exactly*.

To merge two paired data sets:

1. Choose the **General** tab of the Math Functions screen and select the **Merge Paired Data** operator.
2. From the **Selected Data Set** list, select the *primary* paired data set for merging.
3. From the **Paired Data** list, select the *second* paired data set for merging.
4. From the **Select Paired Data Curve** list, set the curve to be merged from the second paired data set (see Section 7.2.1 for more details).
5. Click **Compute** to merge the two data sets.

7.4.14 Generate Data Pairs

The **Generate Data Pairs** function generates a paired data set by pairing values (by time) from two time series data sets. The data pairs in the paired data set may optionally be sorted by ascending x-value. An example use of the function would be to mate a time series record of stage to one of flow to generate a stage-flow paired data set.

The *times* in the two time series data sets *must match exactly*. If a value for a time is missing in either time series, no data value pair is formed and added to the paired data set.

Units and parameter type from the primary time series data set are assigned to the paired data set x-units and x-parameter type. Units and parameter type from the second time series are assigned to the paired data set y-units and y-parameter type.

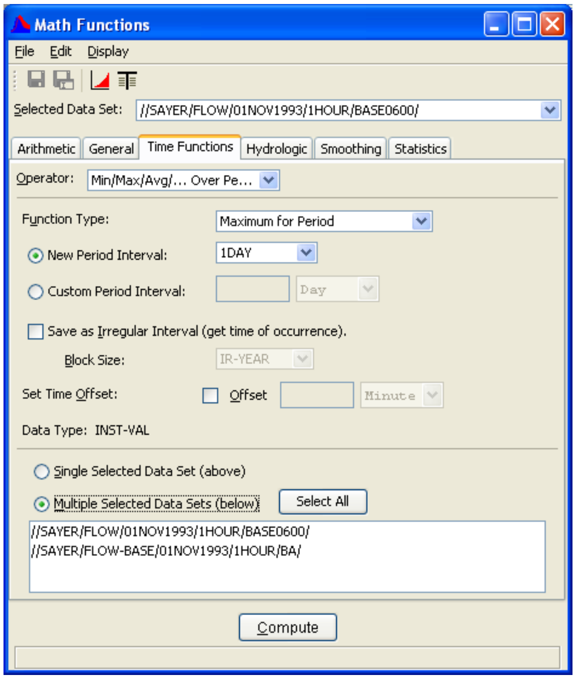
To generate a paired data set by pairing values in two time series data sets:

1. Choose the **General** tab of the Math Functions screen and select the **Generate Data Pairs** operator.
2. From the **Selected Data Set** list, select the primary time series data set. Time series values from this data set will comprise the x-values of the resultant paired data set.
3. From the **Data Set** list, select the second time series data set. Time series values from this data set will comprise the y-values of the resultant paired data set.
4. Check the **Sorted** box if the data pairs are to be sorted by ascending x-values.
5. Click **Compute**.

If time points in the two time series data sets do not match exactly, an error message will be posted and the function operation will not performed.

7.5 Time Functions

The **Time Functions** tab (Figure 7.9) contains the following functions: **Min/Max/Avg… Over Period**, **Copy in Time**, **Shift in Time**, **Change Time Interval**, **Irregular to Regular**, **Regular to Irregular**, **To Irregular using Pattern**, and **Extract Time Series**.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Min/Max/Avg/… Over Pe…

Function Type: Maximum for Period

● New Period Interval: 1DAY

○ Custom Period Interval: Day

□ Save as Irregular Interval (get time of occurrence).

Block Size: IR-YEAR

Set Time Offset: □ Offset Minute

Data Type: INST-VAL

○ Single Selected Data Set (above)

● Multiple Selected Data Sets (below)

//SAYER/FLOW/01NOV1993/1HOUR/BASE0600/
//SAYER/FLOW-BASE/01NOV1993/1HOUR/BA/

Figure 7.9 Math Functions - Time Conversion Tab

7.5.1 Min/Max/Avg/… Over Period

The **Min/Max/Avg/… Over Period** function generates a time series data set using a variety of functions from an existing irregular or regular interval time series data set. The functions are:

Interpolate at end of Period
Maximum for Period
Minimum for Period
Average over Period
Accumulation over Period
Volume for Period
Integration over Period
Number of Valid Data over Period

where "period" is a user selected time interval to evaluate the data over. This function is often used to compute information such as the maximum or minimum annual flow, total annual precipitation, and maximum and minimum daily temperatures. (See remarks following regarding computing values for water years.)

The resulting data can be put in an irregular-interval data set. By doing so, the date and time of each occurrence will be saved with the data. For example, if maximum annual flows are computed, the date and time of that maximum will be provided with an irregular-interval block. You should choose an irregular-interval block size, so that between 50 and 1,000 values are stored in a block. Daily data results should use a block size of “IR-YEAR”. Annual results should use a block size of “IR-DECADE”.

Besides choosing a standard interval, you can also select a **Custom Period Interval**. This provides you the capability to compute information over a period that is different from a standard storage interval, such as two-day averages, and maximum or accumulation over 2, 5 or 10 years. To compute using a custom interval, select **Custom Period Interval** and then enter an integer interval value and select the time increment for that value from the list further to the right. The time increments include: **Minute**, **Hour**, **Day**, **Week**, **Month**, and **Year.** If you choose a custom period interval, the results must be stored in irregular-interval data sets.

The data type of the original time series data governs how values are interpolated. Data type "INST-VAL" (or "INST-CUM") considers the value to change linearly over the interval from the previous data value to the current data value. Data type "PER-AVER" considers the value to be constant at the current data value over the interval. Data type "PER-CUM" considers the value to increase from 0.0 (at the start of the interval) up to the current value over the interval. Interpolation of the three data types is illustrated in Figure 7.10.

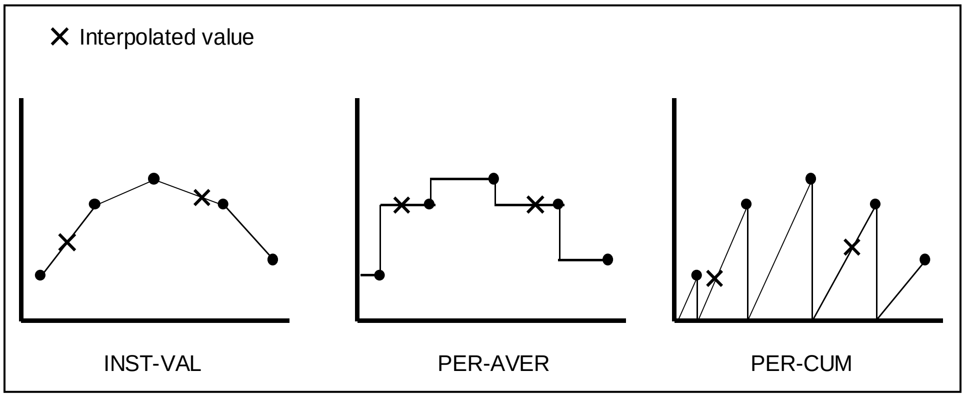


Figure 7.10 Interpolation of "INST-VAL", "PER-AVER" and "PER-CUM" Data

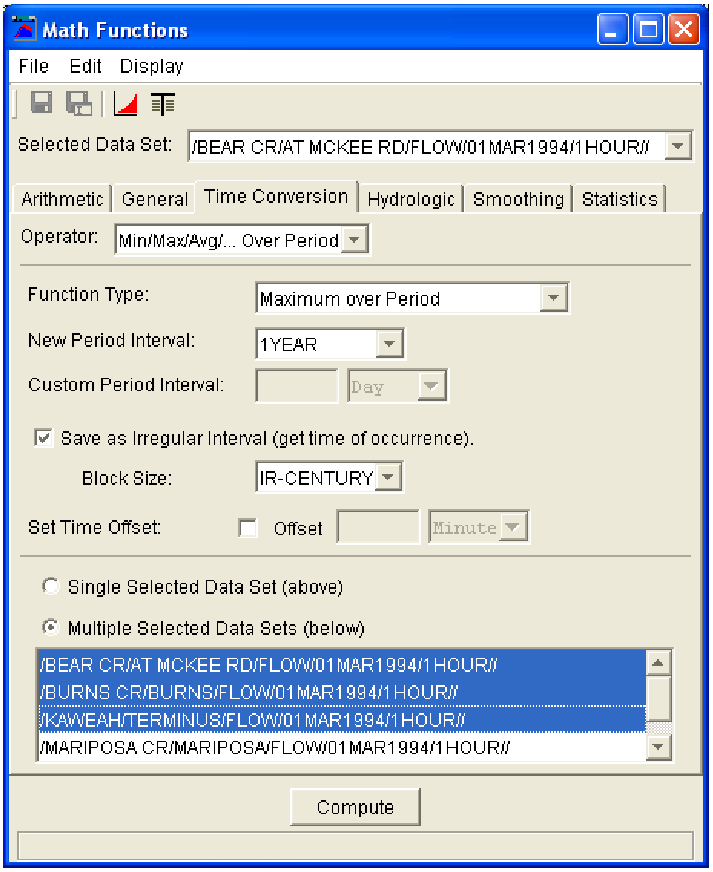
How interpolation is performed for a specific data type influences the computation of new time series values for the selected function. For example, if the data type is "INST-VAL", the function **Maximum for Period** is evaluated by:

- Finding the maximum value of the data points from the original time series that are inclusive in the new time period.
- Linearly interpolate values at beginning and ending of the new time interval, and determine if these values represent the maximum over the interval.

Referring to the plots in Figure 7.10, the **Average over Period** function is applied to a time series by integrating the area under the curve between interpolated points and dividing the result by the interval time.

Times in the new time series may be shifted (offset) from the regular interval time by a user specified offset. As an example, the offset could be used to shift times in regular hourly interval data from the top of the hour to 6 minutes past the hour.

Figure 7.11 shows the appearance of the Math Functions screen for the **Min/Max/Avg/… Over Period** function.



Selected Data Set: /BEAR CR/AT MCKEE RD/FLOW/01MAR1994/1HOUR//

Operator: Min/Max/Avg/... Over Period

Function Type: Maximum over Period

New Period Interval: 1YEAR

Custom Period Interval: Day

☑ Save as Irregular Interval (get time of occurrence).

Block Size: IR-CENTURY

Set Time Offset: □ Offset Minute

○ Single Selected Data Set (above)

● Multiple Selected Data Sets (below)

/BEAR CR/AT MCKEE RD/FLOW/01MAR1994/1HOUR//
/BURNS CR/BURNS/FLOW/01MAR1994/1HOUR//
/KAWEAH/TERMINUS/FLOW/01MAR1994/1HOUR//
/MARIPOSA CR/MARIPOSA/FLOW/01MAR1994/1HOUR//

Figure 7.11 Screen for Min/Max/Avg/… Over Period Function

To compute the minimum, maximum, average, etc. of time series data sets:

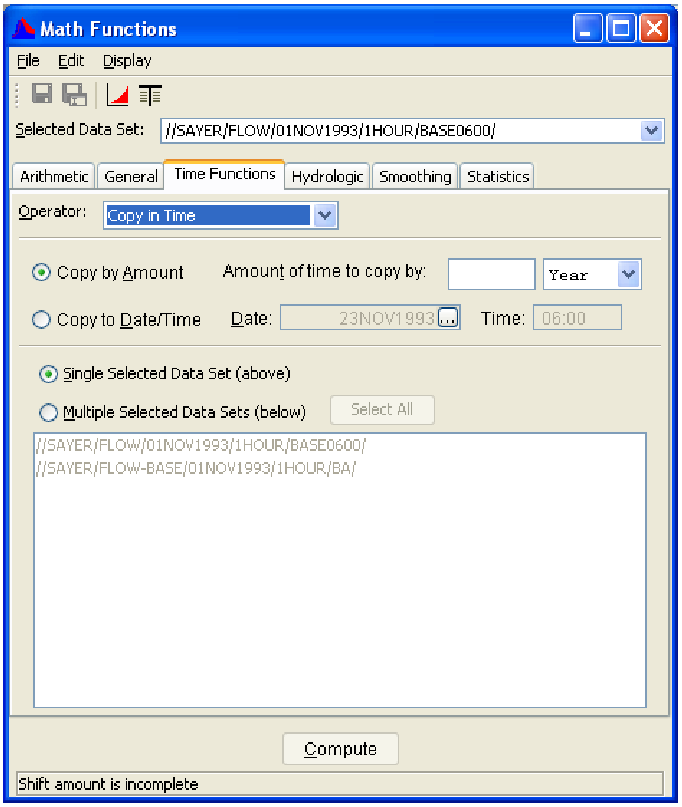
1. Choose the **Time Functions** tab of the **Math Functions** screen and select the **Min/Max/Avg/… Over Period** operator.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. From the **Function Type** list, select the computation to apply to the data.
4. From the **New Period Interval** list or the **Custom Period Interval** list, select the interval period to compute values for.
5. If you select a **Custom** period, enter the integer period interval and select the **minute/hour/day/week/month/year** time increment for that value from the list further to the right.
6. Optionally, you may put the data in an irregular interval time series data set, which will retain the dates and times of each occurrence. You should choose an irregular-interval **Block Size**, so that between 50 and 1,000 values are stored in a block. Daily data results should use a block size of "IR-YEAR". Annual results should use a block size of "IR-DECADE".
7. Optionally, you may specify a **Time Offset** to offset the new points in time from the standard interval time. For example, if you select a daily interval for the Time Interval, you may use the option to offset time points to be at 6 a.m. instead of at midnight. If the **Set Time Offset** box is checked, enter an integer number in the box to the right and select the **minute/hour/day/week/month/year** time increment from the list further to the right. If the **Set Time Offset** box is unchecked, times in the resultant data sets will fall on the regular interval time (typical setting).
8. Click **Compute**.

7.5.2 Copy in Time

The **Copy in Time** function copies the data set and sets a new start time with a specified increment from the original or to start at a specified date and time. Both the original and the copied data values are not changed, except where the two data sets overlap. The difference between the **Copy in Time** function and the **Shift in Time** function is that the shift function deletes the original data before storing the shifted data, whereas the copy function does not. Data sets may be regular or irregular interval time series data.

To copy time series data sets to another time:

1. Choose the **Time Functions** tab of the Math Functions screen and select the **Copy in Time** operator, as shown in Figure 7.12.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Copy in Time

● Copy by Amount Amount of time to copy by: Year

○ Copy to Date/Time Date: 23NOV1993 Time: 06:00

● Single Selected Data Set (above)

○ Multiple Selected Data Sets (below)

//SAYER/FLOW/01NOV1993/1HOUR/BASE0600/
//SAYER/FLOW-BASE/01NOV1993/1HOUR/BA/

Shift amount is incomplete

Figure 7.12 Copy in Time

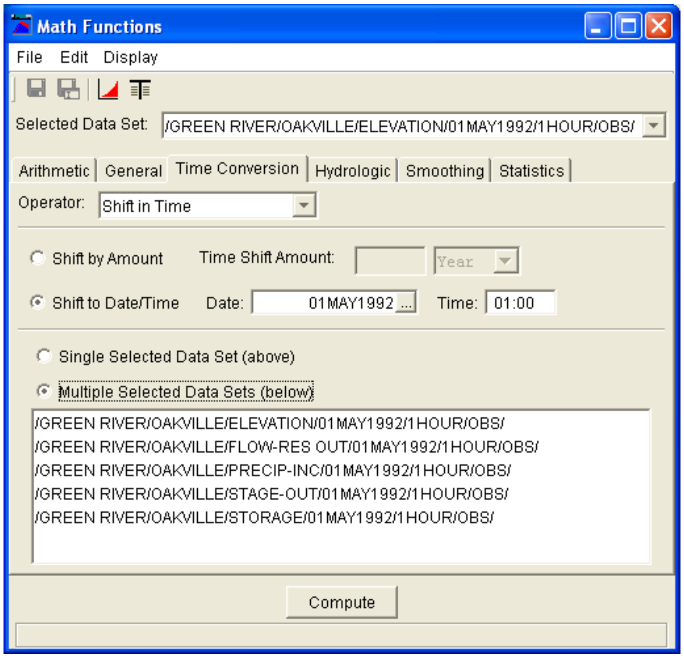
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. In the **Copy by Amount** box, enter the amount of the time to increment the data sets by (integer values only) and then select **Minute**, **Hour**, **Day, Week, Month** or **Year**. To increment data backwards in time, use a negative amount for the time shift.
4. Alternatively, you may select the **Copy to Date/Time** radio button and specify the date and time to copy the first value in each data set to.

7.5.3 Shift in Time

The **Shift in Time** function shifts the times in time series data sets by a specified increment or to start at a specified date and time. The original data values are removed from the file when the shifted data is stored. To accomplish this, the data sets must be saved immediately following the operation. For example, if you shift data by one year from 1965 to 1966, data will exist for 1966 and not 1965. To retain the original data, use the **Copy in Time** function. Data sets may be regular or irregular interval time series data.

To shift times in time series data sets:

1. Choose the **Time Functions** tab of the Math Functions screen and select the **Shift in Time** operator, as shown in Figure 7.13.



Selected Data Set: /GREEN RIVER/OAKVILLE/ELEVATION/01MAY1992/1HOUR/OBS/

Operator: Shift in Time

○ Shift by Amount Time Shift Amount: Year

● Shift to Date/Time Date: 01MAY1992 Time: 01:00

○ Single Selected Data Set (above)

● Multiple Selected Data Sets (below)

/GREEN RIVER/OAKVILLE/ELEVATION/01MAY1992/1HOUR/OBS/
/GREEN RIVER/OAKVILLE/FLOW-RES OUT/01MAY1992/1HOUR/OBS/
/GREEN RIVER/OAKVILLE/PRECIP-INC/01MAY1992/1HOUR/OBS/
/GREEN RIVER/OAKVILLE/STAGE-OUT/01MAY1992/1HOUR/OBS/
/GREEN RIVER/OAKVILLE/STORAGE/01MAY1992/1HOUR/OBS/

Figure 7.13 Screen for Shift in Time Function

2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. In the **Shift by Amount** box, enter the amount of the time to shift the data sets by (integer values only) and then select **Minute**, **Hour**, **Day, Week, Month** or **Year**. To shift data backwards in time, use a negative amount for the time shift.
4. Alternatively, you may select the **Shift to Date/Time** radio button and specify the date and time to shift the first value in each data set to.

7.5.4 Change Time Interval

The **Change Time Interval** function interpolates or extracts values from existing regular interval time series data sets to new regular interval time series data sets with a different time interval.

Whether the time series data type is "INST-VAL", "INST-CUM", "PER-AVER", or "PER-CUM" controls how interpolation is performed. Interpolated values are derived from "INST-VAL" or "INST-CUM" data using linear interpolation. Values are derived from "PER-AVER" data by computing the period average value over the new time interval. Values are derived from "PER-CUM" data by computing the period cumulative value over the new time interval.

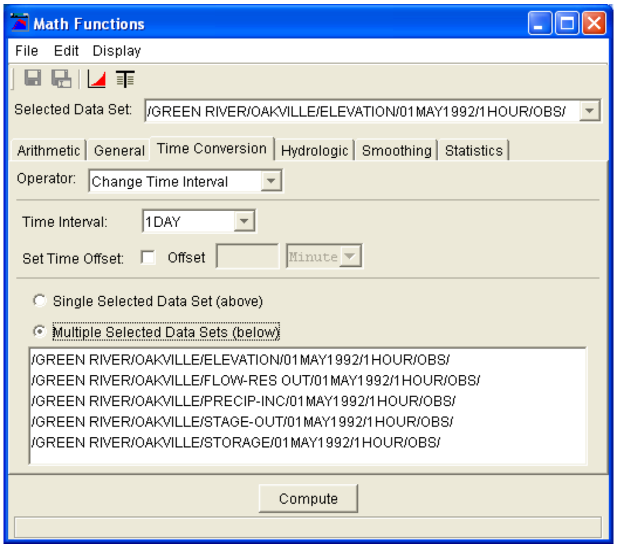
For example, if an original data set is hourly data and the new regular interval data set is to have a six-hour time interval:

- The value for "INST-VAL" or "INST-CUM" type data is computed from the linear interpolation of hourly points bracketing the new six-hour time point.
- The value for "PER-AVER" type data is computed from the period average value over the six-hour interval.
- The value for "PER-CUM" type data is computed from the accumulated value over the six-hour interval.

The treatment of missing value data is also dependent upon data type. Interpolated "INST-VAL" or "INST-CUM" points must be bracketed or coincident with valid (not missing) values in the original time series; otherwise the interpolated values are set as missing. Interpolated “"PER-AVER" or "PER-CUM" data must contain all valid values over the interpolation interval; otherwise the interpolated value is set as missing.

To change the time interval in time series data sets:

1. Choose the **Time Functions** tab of the Math Functions screen and select the **Change Time Interval** operator, as shown in Figure 7.14.



Selected Data Set: /GREEN RIVER/OAKVILLE/ELEVATION/01MAY1992/1HOUR/OBS/

Operator: Change Time Interval

Time Interval: 1DAY

Set Time Offset: ☐ Offset Minute

○ Single Selected Data Set (above)

● Multiple Selected Data Sets (below)

/GREEN RIVER/OAKVILLE/ELEVATION/01MAY1992/1HOUR/OBS/
/GREEN RIVER/OAKVILLE/FLOW-RES OUT/01MAY1992/1HOUR/OBS/
/GREEN RIVER/OAKVILLE/PRECIP-INC/01MAY1992/1HOUR/OBS/
/GREEN RIVER/OAKVILLE/STAGE-OUT/01MAY1992/1HOUR/OBS/
/GREEN RIVER/OAKVILLE/STORAGE/01MAY1992/1HOUR/OBS/

Figure 7.14 Change Time Interval Function

2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. From the **Time Interval** list, select the new time interval for the data sets.
4. Optionally, you may specify a **Time Offset** to offset the new points in time from the standard interval time. For example, if you select a daily interval for the Time Interval, you may use the option to offset time points to be at 6 a.m. instead of at midnight. If the **Set Time Offset** box is checked, enter an integer number in the box to the right and select the **minute/hour/day/week/month/year** time increment from the list further to the right. If the **Set Time Offset** box is unchecked, times in the resultant data sets will fall on the regular interval time (typical setting).
5. Click the **Compute** button to perform the interpolation.

7.5.5 Irregular to Regular

The **Irregular to Regular** function interpolates or extracts values from existing irregular interval time series data sets to create new regular interval time series data sets, or "snaps" (moves) irregular interval data to the regular times.

For interpolating, whether the time series data type is "INST-VAL", "INST-CUM", "PER-AVER", or "PER-CUM" controls how the interpolation is performed. Interpolated values are derived from "INST-VAL" or "INST-CUM" data using linear interpolation. Values are derived from "PER-AVER" data by computing the period average value over the new regular time interval. Values are derived from "PER-CUM" data by computing the period cumulative value over the new regular time interval.

For example, if the original data set has data that typically occurs several times in a six-hour period and the new regular interval data set is to have a six-hour time interval:

- The value for "INST-VAL" or "INST-CUM" type data is computed from the linear interpolation of points bracketing the new six-hour time point.
- The value for "PER-AVER" type data is computed from the period average value over the six-hour interval.
- The value for "PER-CUM" type data is computed from the accumulated value over the six-hour interval.

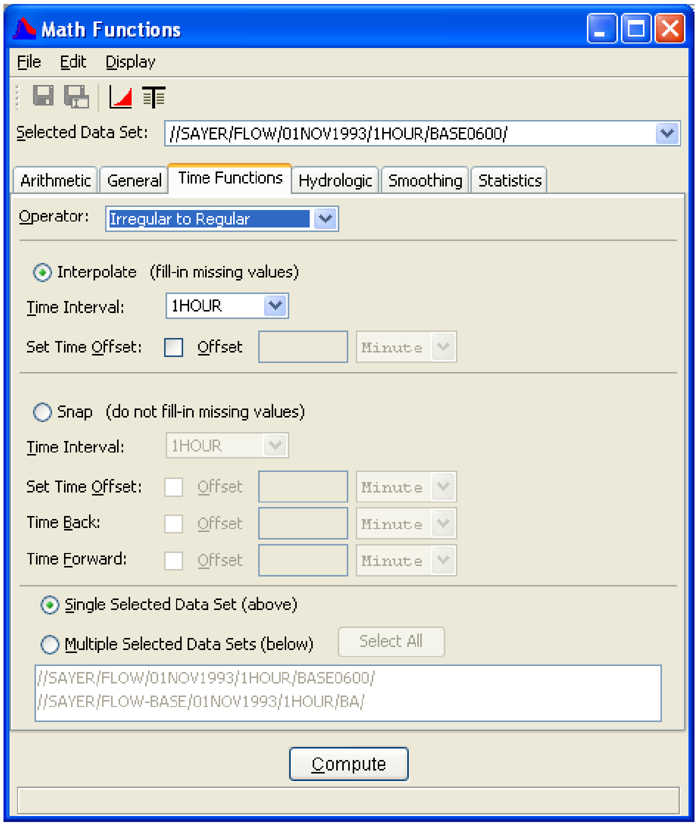
The **Snap** function derives new regular interval time series by "snapping" data to a user specified regular interval time if the time of the original data falls within the time window tolerance set by the user. A snap just changes the time of the data values and does no interpolation. For example, a time series record from a gauge recorder collects readings six minutes past the hour. The Snap Irregular to Regular function is often used to “snap” or shift the time points to the top of the hour.

Times in resultant time series may be shifted (offset) from the regular interval time by a user specified offset. As an example, the offset could be used to shift times in regular hourly interval data from the top of the hour to 6 minutes past the hour.

By default values in the resultant regular interval time series data sets are set to missing unless matched to times in the original time series data set (within the time window tolerance).

To convert irregular interval time series data to regular interval data by interpolation:

1. Choose the **Time Functions** tab of the Math Functions screen and select the **Irregular to Regular** operator, as shown in Figure 7.15. Click **Interpolate**.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Irregular to Regular

● Interpolate (fill-in missing values)

Time Interval: 1HOUR

Set Time Offset: □ Offset Minute

○ Snap (do not fill-in missing values)

Time Interval: 1HOUR

Set Time Offset: □ Offset Minute

Time Back: □ Offset Minute

Time Forward: □ Offset Minute

● Single Selected Data Set (above)

○ Multiple Selected Data Sets (below)

//SAYER/FLOW/01NOV1993/1HOUR/BASE0600/
//SAYER/FLOW-BASE/01NOV1993/1HOUR/BA/

Figure 7.15 Irregular Data to Regular Data Function

2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. From the **Time Interval** list, select the time interval for the data sets.
4. Optionally, you may specify a **Time Offset** to offset the new points in time from the standard interval time. For example, if you select a daily interval for the Time Interval, you may use the option to offset time points to be at 6 a.m. instead of at midnight. If the **Set Time Offset** box is checked, enter an integer number in the box to the right and select the **minute/hour/day/week/month/year** time increment from the list further to the right. If the **Set Time Offset** box is unchecked, times in the resultant data sets will fall on the regular interval time (typical setting).
5. Click the **Compute** button to perform the interpolation.

To convert irregular interval time series data to regular interval data by snapping:

1. Choose the **Time Functions** tab of the Math Functions screen and select the **Irregular to Regular** operator. Click **Snap**.
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. From the **Time Interval** list, select the regular interval time period for the resultant time series data sets.
4. Optionally, you may specify a **Time Offset** to offset the new points in time from the standard interval time. For example, if you select a daily interval for the Time Interval, you may use the option to offset time points to be at 6 a.m. instead of at midnight. If the **Set Time Offset** box is checked, enter an integer number in the box to the right and select the **minute/hour/day/week month/year** time increment from the list further to the right. If the **Set Time Offset** box is unchecked, times in the resultant data sets will fall on the regular interval time (typical setting).
5. Use the **Time Back** and **Time Forward** offsets to define a time window around the regular interval time. If a time point from the selected time series data set falls within the time window, the data value is applied to the new regular interval time point. These offsets are set in the same manner as the **Time Offset** above.
6. Click **Compute**.

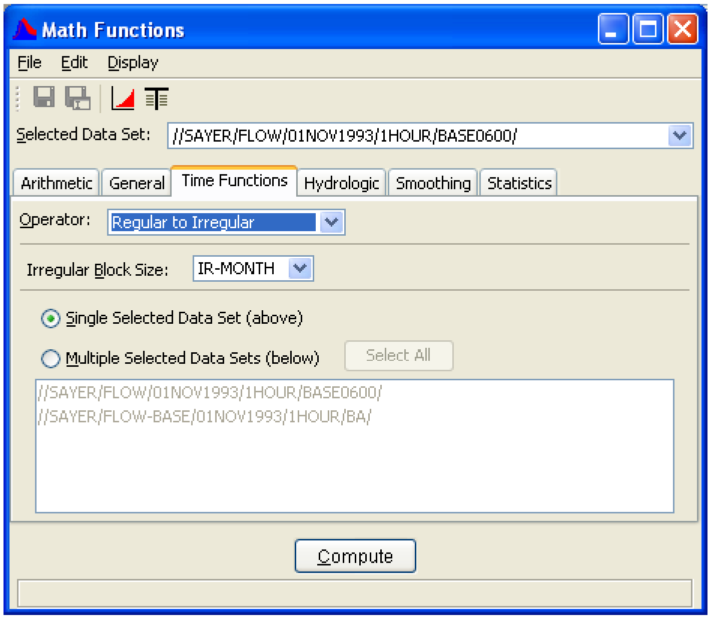
7.5.6 Regular to Irregular

The **Regular to Irregular** function derives new irregular interval time series data sets from existing regular interval time series data sets. No conversion, time shifting, or other operations are applied to the data. Embedded missing data values are retained in the irregular interval data sets.

The **Irregular Block Size** dictates how the data is stored in the HEC-DSS file. You should choose an irregular-interval **block size**, so that between 100 and 2,000 values are stored in a block. Data that occurs approximately on the average of once an hour should use a block size of "IR-MONTH". Data that occurs approximately on the average of once a day should use a block size of "IR-YEAR". Annual values should use a block size of "IR-DECADE".

To convert regular interval time series data to irregular interval data:

1. Choose the **Time Functions** tab of the Math Functions screen and select the **Regular to Irregular** operator, as shown in Figure 7.16.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Regular to Irregular

Irregular Block Size: IR-MONTH

● Single Selected Data Set (above)

○ Multiple Selected Data Sets (below)

//SAYER/FLOW/01NOV1993/1HOUR/BASE0600/
//SAYER/FLOW-BASE/01NOV1993/1HOUR/BA/

Figure 7.16 Regular Data to Irregular Data Function

2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. Select the **Irregular Block Size** for the data sets. Choose an irregular-interval **block size**, so that between 100 and 2,000 values are stored in a block. Data that occurs approximately on the

average of once an hour should use a block size of "IR-MONTH". Data that occurs approximately on the average of once a day should use a block size of "IR-YEAR". Annual values should use a block size of "IR-DECADE".

4. Click **Compute**.

7.5.7 To Irregular using Pattern

The **To Irregular using Pattern** function generates a new time series data set from an existing irregular or regular interval time series data set. The times for the new time series are defined by the times of a second time series data set. Values for the new time series are computed from the original time series data using one of seven available functions. The functions are:

Interpolate at end of Period
Maximum for Period
Minimum for Period
Average over Period
Accumulation over Period
Volume for Period
Integration over Period
Number of Valid Data over Period

where "period" is the time between data points in the pattern data set.

The data type of the original time series data governs how values are interpolated. Data type "INST-VAL" (or "INST-CUM") considers the value to change linearly over the interval from the previous data value to the current data value. Data type "PER-AVER" considers the value to be constant at the current data value over the interval. Data type "PER-CUM" considers the value to increase from 0.0 (at the start of the interval) up to the current value over the interval. Interpolation of the three data types is illustrated in Figure 7.17.

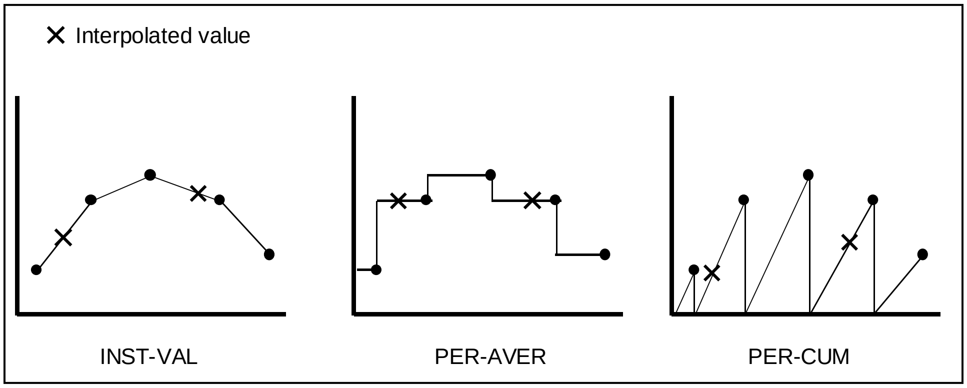


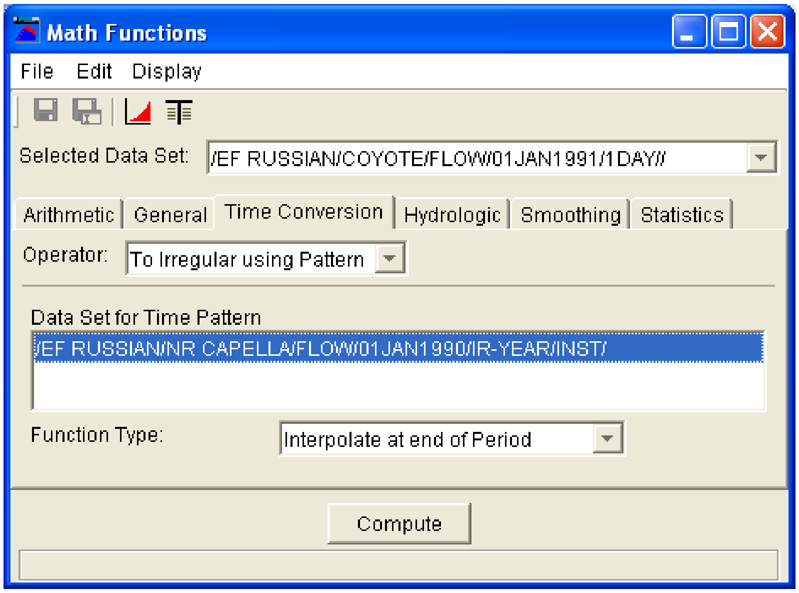
Figure 7.17 Interpolation of "INST-VAL", "PER-AVER" and "PER-CUM" Data

How interpolation is performed for a specific data type influences the computation of new values for the selected function. For example, if the data type is "INST-VAL", the function **Maximum for Period** is evaluated by:

- Finding the maximum value of the data points from the original time series that are inclusive in the new time interval.
- Linearly interpolate values at beginning and ending of the new time interval, and determine if these values represent the maximum over the interval.

Referring to the plots in Figure 7.17, the **Average over Period** function is applied to a time series by integrating the area under the curve between interpolated points and dividing the result by the interval time.

The appearance of the Math Functions screen for the Transform to Irregular interval function is shown in Figure 7.18.



Selected Data Set: /EF RUSSIAN/COYOTE/FLOW/01JAN1991/1DAY//

Operator: To Irregular using Pattern

Data Set for Time Pattern
/EF RUSSIAN/NR CAPELLA/FLOW/01JAN1990/IR-YEAR/INST/

Function Type: Interpolate at end of Period

Figure 7.18 Screen for Transforming To Irregular using Pattern Function

To transform to an irregular interval time series data set:

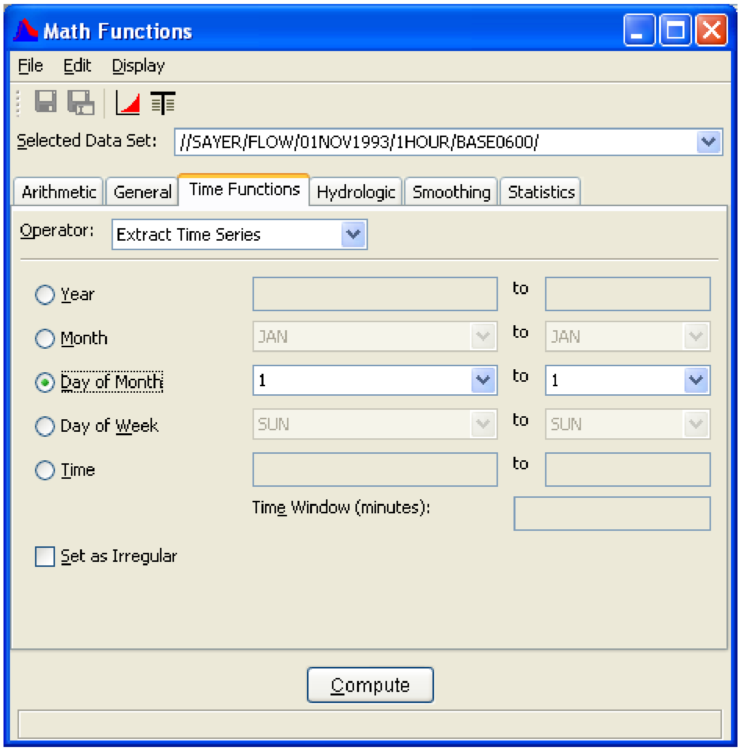
1. Choose the **Time Functions** tab of the Math Functions screen and select the **To Irregular using Pattern** operator.
2. From the **Selected Data Se**t list, select a time series data set.
3. From the **Data Set for Time Pattern** list, select a time series data set to provide the time pattern for the resultant time series.
4. From the **Function Type** list, select the method for computing new time series values.
5. Click **Compute**.

7.5.8 Extract Time Series

The **Extract Time Series** function selects/extracts data points from a regular or irregular interval time series data set based on user defined time specifications. For example, the function may be used to extract values observed every day at noon from hourly interval data.

Data may be extracted on the basis of one of the following time specifications: year, month of the year (January, February, etc.), day of month, day of week (Sunday, Monday, etc.), or time of day.

The appearance of the Math Functions screen for the Extract Time Series function is shown in Figure 7.19.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Extract Time Series

○ Year ______ to ______
○ Month JAN to JAN
● Day of Month 1 to 1
○ Day of Week SUN to SUN
○ Time ______ to ______
Time Window (minutes): ______

□ Set as Irregular

Figure 7.19 Screen for Extract Time Series Function

To derive a new time series data set from the selected times of another time series data set:

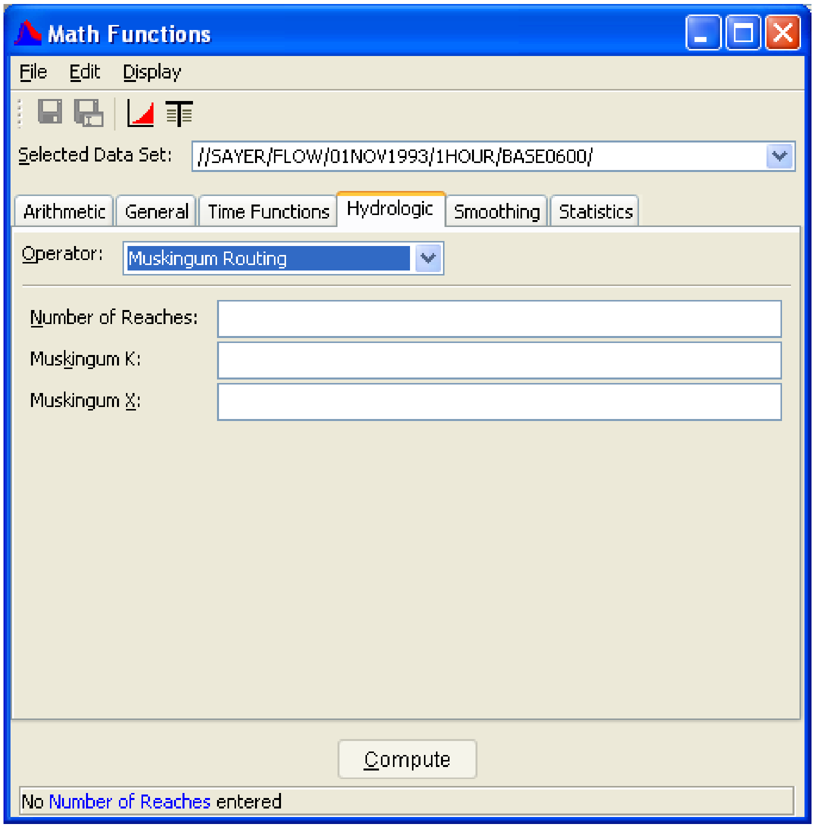
1. Choose the **Time Functions** tab of the Math Functions screen and select the **To Irregular using Pattern** operator.
2. From the **Selected Data Se**t list, select a time series data set to apply the function.
3. Choose one of the time levels, **Year**, **Month**, **Day of Month**, **Day of Week**, or **Time** of day and complete the boxes to the right. Entering or selecting a value in only the left-most box sets a time specification for data extraction to an individual year, month, day,

or time of day. Fill in both boxes to specify a time range for data extraction.

4. Click **Year** to extract data for an individual year or range of years. To extract data for an individual year, fill the left-most box in with the desired year. To extract data for a range of years fill the left box with the beginning year and the right box with the ending year.
5. Click **Month** to extract data for an individual month of the year or range of months. Use the lists to set the month boxes. The list contains the months of the year, JAN to DEC. Set only the left-most box to extract data for a single month of the year. Set both boxes to extract data for a range of months. The range of months can be set to encompass the end of the year; that is, "OCT to FEB" is a valid setting.
6. Click **Day of Month** to extract data for an individual day or range of day of month days. The days are set using the two lists to the right. As above, the lists can be used to specify a single day of month, using the left-most pull down, or a range of days, using both lists. The day of month values range from 1 to 31, and "LastDay", the last day of the month, which may vary by month. The range of days can be set to encompass the end of the month. For example, for "27 to 4", data will be extracted from the 27th of one month to the 4th of the next month.
7. Click **Day of Week** to extract data for an individual or range of day of the weekdays. Select day from the two lists to the right. As above, you may use the lists to specify one day, using the left-most list, or a range of days, using both lists. The day of week values range from SUN to SAT. The range of days can be set to encompass the end of the week; that is, "SAT to MON" is a valid setting.
8. Click **Time** to extract data by the time of day. An individual time is specified by completing the left-most box only. Complete both boxes to specify a time of day range for data extraction. The time specification employs the standard four-digit military style 24-hour format (e.g. "2300" or "0310 to 0600"). The time range can encompass the end of day; that is, "2200 to 0330" is a valid setting. The optional **Time Window** box is applied to the time of day extraction only. The Time Window is an integer value in minutes, used to extend the beginning and ending of the time of day period for data extraction. For example, with a time of day extraction time of "0300" and a Time Window of "10 minutes", data will be extracted from the selected time series if times falls within in the period 0250 to 0310.
9. Select the **Set as Irregular** checkbox (recommended) to ensure the resultant time series data set is identified as irregular interval time series data. If the box is not checked, the function will attempt to determine if the extracted data set can be classified as regular interval time series data.

7.6 Hydrologic Functions

The **Hydrologic** tab (Figure 7.20) contains the following functions: **Muskingum Routing**, **Straddle Stagger Routing**, **Modified Puls Routing**, **Rating Table**, **Reverse Rating Table**, **Two Variable Rating Table**, **Decaying Basin Wetness**, **Shift Adjustment**, **Period Constants**, **Multiple Linear Regression**, **Apply Multiple Linear Regression**, **Conic Interpolation**, **Polynomial**, **Polynomial with Integral**, and **Flow Accumulator Gage Processor**.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Muskingum Routing

Number of Reaches:

Muskingum K:

Muskingum X:

No Number of Reaches entered

Figure 7.20 Math Functions - Hydrologic Tab

7.6.1 Muskingum Routing

The **Muskingum Routing** function routes a regular interval time series data set by the Muskingum hydrologic routing method.

1. Choose the **Hydrologic** tab of the Math Functions screen and select the **Muskingum Routing** operator, as shown in Figure 7.20.
2. From the **Selected Data Set** list, select the time series data set for routing.
3. In the **Number of Reaches** box, specify the number of routing subreaches.

4. In the **Muskingum K** box, enter the Muskingum "K" routing parameter (travel time, in hours).
5. In the **Muskingum X** box, enter the Muskingum "x" routing parameter. The Muskingum "x" value cannot be less than 0.0 (maximum attenuation) or greater than 0.5 (no attenuation).
6. Click **Compute** to route the selected time series data set.

An error message will be displayed if a compute is attempted using an invalid Muskingum "x" value, and no routing will be performed.

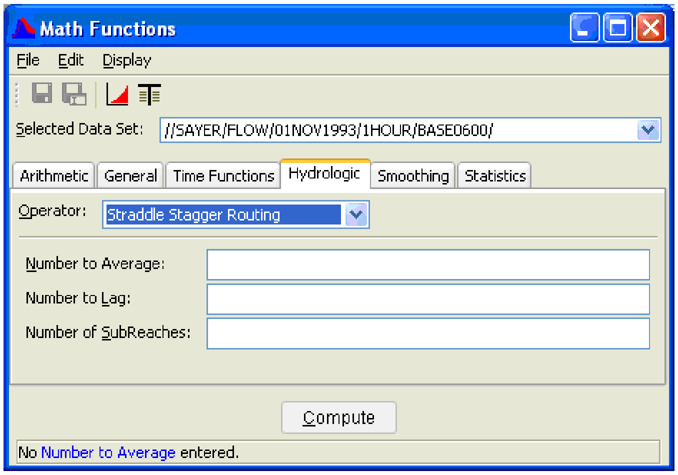
The set of Muskingum routing parameters entered may potentially produce numerical instabilities in the routed time series. The parameters are first checked by the function for possible instabilities before the routing proceeds. If instabilities are possible, a warning message box will appear presenting the details of the instabilities. Click the **Yes** button in the message box to proceed with the compute, or click **No** to cancel the operation.

7.6.2 Straddle Stagger Routing

The **Straddle Stagger Routing** function routes a regular interval time series data set by the Straddle Stagger hydrologic routing method.

To route a time series data set by the Straddle Stagger hydrologic routing method:

1. Choose the **Hydrologic** tab of the Math Functions screen and select the **Straddle Stagger Routing** operator, as shown in Figure 7.21.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Straddle Stagger Routing

Number to Average:

Number to Lag:

Number of SubReaches:

No Number to Average entered.

Figure 7.21 Straddle Stagger Routing Method

2. From the **Selected Data Set** list, select the time series data set for routing.

3. In the **Number to Average** box, enter the number of ordinates to average over (Straddle).
4. In the **Number to Lag** box, enter the number of ordinates to lag (Stagger).
5. In the **Number of SubReaches** box, enter the number of routing subreaches.
6. Click **Compute** to route the selected time series data set.

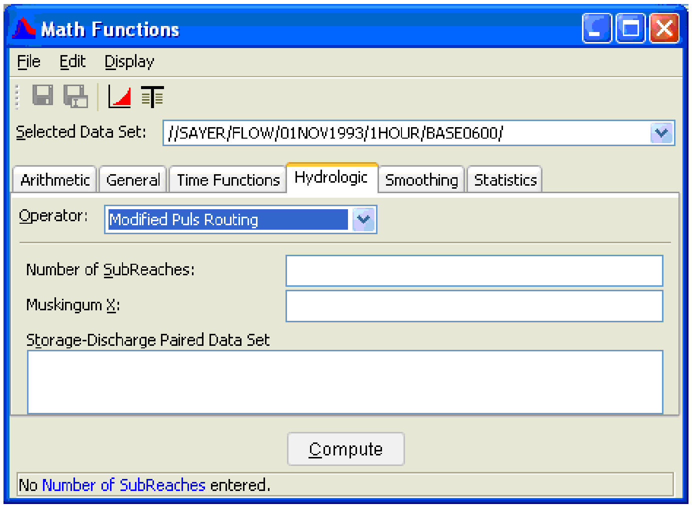
7.6.3 Modified Puls Routing

The **Modified Puls Routing** function routes a regular interval time series data set by the Modified Puls hydrologic routing method.

With the Modified Puls method, outflow from a routing reach is a unique function of storage. The storage-discharge relationship is entered into the function as a paired data set, where the x-values are storage and the y-values are discharge.

To route a time series data set by the Modified Puls hydrologic routing method:

1. Choose the **Hydrologic** tab of the Math Functions screen and select the **Modified Puls Routing** operator, as shown in Figure 7.22.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Modified Puls Routing

Number of SubReaches:

Muskingum X:

Storage-Discharge Paired Data Set

No Number of SubReaches entered.

Figure 7.22 Modified Puls Routing Method

2. From the **Selected Data Set** list, select the time series data set for routing.
3. From the **Storage-Discharge Paired Data Set** list, select the paired data set containing the storage-discharge table. Only paired data sets will appear in this list, and only one data set at a time can be selected.

4. In the **Number of SubReaches** box, enter the number of routing subreaches.
5. In the **Muskingum X** box, enter the Muskingum "x" routing parameter. The Muskingum "x" value cannot be less than 0.0 or greater than 0.5. Enter 0.0 to route by the Modified Puls method, or a value greater than 0.0 to apply the Working R&D method.
6. Click **Compute** to route the selected time series data set.

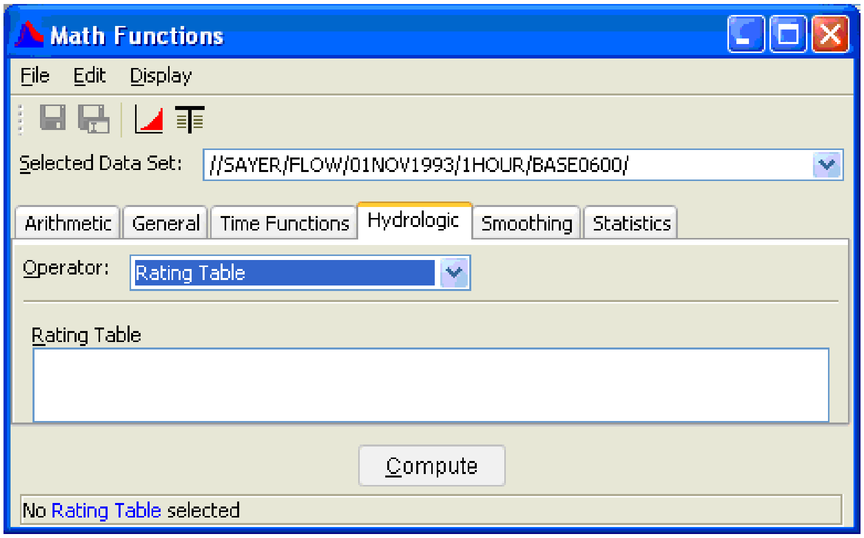
7.6.4 Rating Table

The **Rating Table** function transforms/interpolates values in a time series data set using the rating table x-y values stored in a paired data set. For example, the Rating Table function can be used to transform stage values to flow values using a stage-flow rating table.

The units and parameter type of the resultant time series data set are duplicated from the y-units and y-parameter type of the rating table paired data set. For example, if a stage-flow rating table has a y-data parameter type of "FLOW" and y-units of "cfs", the resultant time series data set will have the data parameter type “FLOW” and the units "cfs".

To derive a new time series data set from the rating table interpolation based on the selected time series data set:

1. Choose the **Hydrologic** tab of the Math Functions screen and select the **Rating Table** operator, as shown in Figure 7.23.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Rating Table

No Rating Table selected

Figure 7.23 Rating Table Method

2. From the **Selected Data Set** list, select a time series data set for rating table interpolation. If the selected data set is not a time series, the remaining input boxes on the screen will be unavailable.

3. From the **Rating Table** list, select the paired data set containing the rating table information. Only paired data sets will appear in this list, and you may select only one data set at a time.
4. Click **Compute** to perform the rating table interpolation of the selected time series data set.

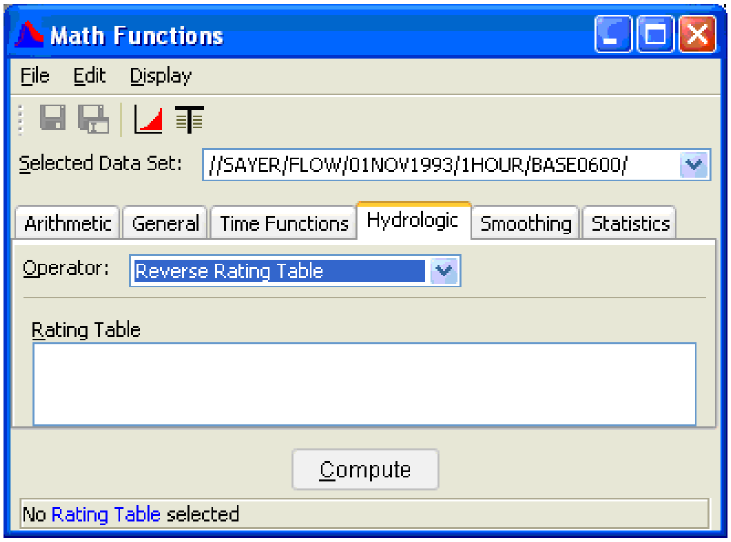
7.6.5 Reverse Rating Table

The **Reverse Rating Table** function transforms/interpolates values in a time series data set using the reverse of the rating table stored in a paired data set. For example, the **Reverse Rating Table** function can be used to transform flow values to stage values using a stage-flow rating table.

The units and parameter type of the resultant time series data set are duplicated from the x-units and x-parameter type of the rating table paired data set. For example, if a stage-flow rating table has an x-data parameter type of "STAGE" and x-units of "ft", the resultant time series data set will have the data parameter type "STAGE" and the units "ft".

To derive a new time series data set from the reverse rating table interpolation of the selected time series data set:

1. Choose the **Hydrologic** tab of the Math Functions screen and select the **Reverse Rating Table** operator, as shown in Figure 7.24.


Math Functions
File Edit Display
Arithmetic General Time Functions Hydrologic Smoothing Statistics
Rating Table
Compute

Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Reverse Rating Table

No Rating Table selected

Figure 7.24 Reverse Rating Table Method

2. From the **Selected Data Set** list, select a time series data set for reverse rating table interpolation. If the selected data set is not a time series, the remaining input boxes on the screen will be unavailable.

3. From the **Rating Table** list, select the paired data set containing the rating table information. Only paired data sets will appear in this list, and you may select only one data set at a time.
4. Click **Compute** to perform the reverse rating table interpolation of the selected time series data set.

7.6.6 Two Variable Rating Table

The **Two Variable Rating Table** function computes new time series values from the input of two other independent time series data sets. The functional relationship is specified by a table of values contained in a paired data set having multiple sets of x-y curves.

As an example, reservoir release is a function of both the gate opening height and reservoir elevation (Figure 7.25).

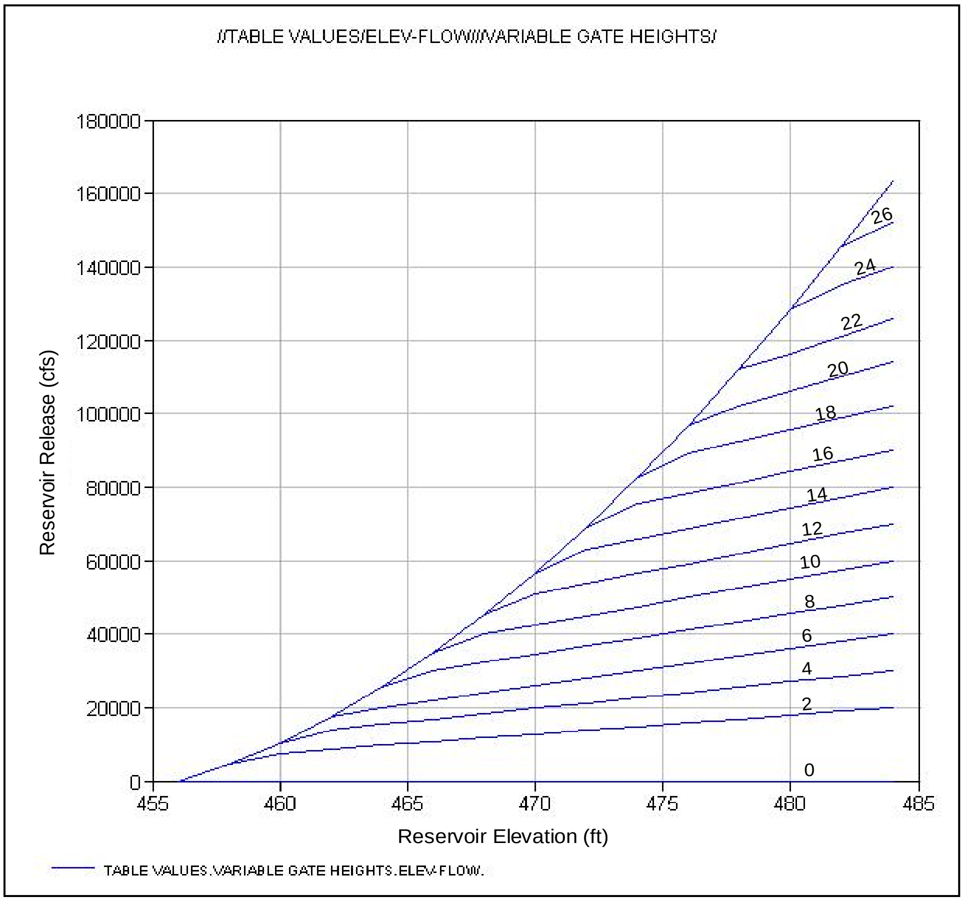


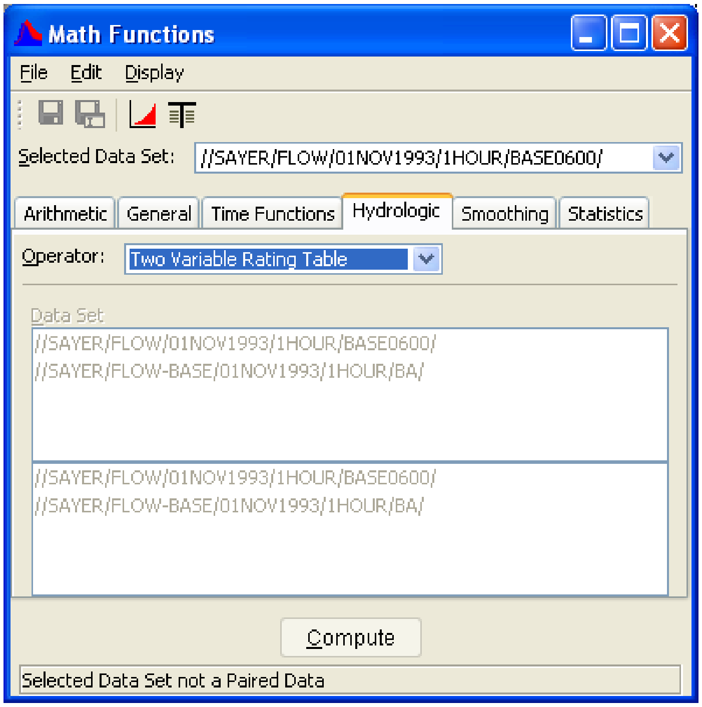
Figure 7.25 Example of two variable rating table paired data, reservoir release as a function of reservoir elevation and gate opening height (curve labels).

For each gate opening height, there is a reservoir elevation-reservoir release curve, where reservoir elevation is the independent variable (x-values) and reservoir release is the dependent variable (y-values) of a paired data set. Each paired data curve has a curve label. In this case, the curve label is assigned the gate opening height. Using the paired data set shown in Figure 7.25 (page 7-45), you may employ the Two Variable Rating Table function to interpolate time series values of reservoir elevation and gate opening height to develop a time series of reservoir release.

Times for the two input time series data sets must match. Curve labels must be set for curves in the rating table paired data set and must be interpretable as numeric values.

By default, the units and parameter type of the resultant time series data set are duplicated from the y-units and y-parameter type of the rating table paired data set.

The appearance of the Math Functions screen for the Two Variable Rating Table function is shown in Figure 7.26.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Two Variable Rating Table

Data Set
//SAYER/FLOW/01NOV1993/1HOUR/BASE0600/
//SAYER/FLOW-BASE/01NOV1993/1HOUR/BA/

//SAYER/FLOW/01NOV1993/1HOUR/BASE0600/
//SAYER/FLOW-BASE/01NOV1993/1HOUR/BA/

Selected Data Set not a Paired Data

Figure 7.26 Screen for Two Variable Rating Table Function

To derive a new time series data set from the two variable rating table interpolation of two other time series data sets:

1. Choose the **Hydrologic** tab of the Math Functions screen and select the **Two Variable Rating Table** operator.
2. From the **Selected Data Set** list, select the paired data set containing the rating table. If the selected data set is not a paired data set, the remaining input boxes on the screen will be unavailable.

3. From the two lists shown below the **Data Set** title, select the two input time series data sets.
    - Use the top list to select the time series data set matching the **x-values parameter** of the rating table. In Figure 7.26 (page 7-46), this is a time series data set of reservoir elevation.
    - Use the bottom list to select the time series data set matching the **curve labels parameter**. In Figure 7.26 (page 7-46) this is a time series data set of gate opening height. You may select only one data set from a list.
4. Once the data sets are selected, click **Compute** to apply the Two Variable Rating Table function**.**

7.6.7 Decaying Basin Wetness

The **Decaying Basin Wetness** function computes a time series of decaying basin wetness parameters from a regular interval time series data set of incremental precipitation by:

$$\text{TSPar}(t) = \text{Rate} * \text{TSPar}(t-1) + \text{TSPrecip}(t)$$

where Rate is the decay rate and $0 < \text{Rate} < 1$.

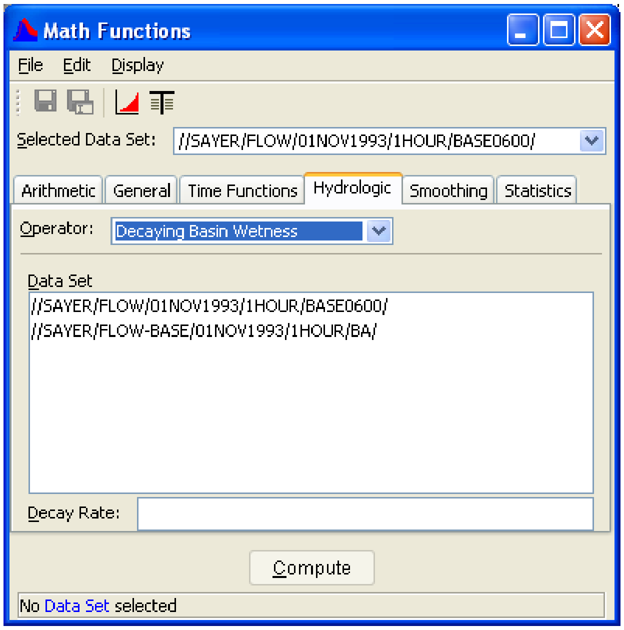
The first value of the resultant time series, TSPar(1), is set to the first value in the precipitation time series, TSPrecip (1). Missing values in the precipitation time series are zero for the above computation.

To compute a time series of decaying basin wetness parameters:

1. Choose the **Hydrologic** tab of the Math Functions screen and select the **Decaying Basin Wetness** operator, as shown in Figure 7.27.
2. From the **Selected Data Set** list, select the time series data set of incremental precipitation.
3. Enter the **Decay Rate**. This value must be greater than 0.0 and less than 1.0.
4. Click **Compute**.

7.6.8 Shift Adjustment

The **Shift Adjustment** function linearly interpolates values in the primary time series data set to the times defined by a second time series data set. If times in the new time series precede the first data point in the primary time series, the value for these times is set to 0.0. If times in the new time series occur after the last data point in the primary time series, the value



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Decaying Basin Wetness

Data Set
//SAYER/FLOW/01NOV1993/1HOUR/BASE0600/
//SAYER/FLOW-BASE/01NOV1993/1HOUR/BA/

Decay Rate:

No Data Set selected

Figure 7.27 Decaying Basin Wetness Method

for these times is set to the value of the last point in the primary time series. Interpolation of values with the Shift Adjustment function is shown in Figure 7.28.

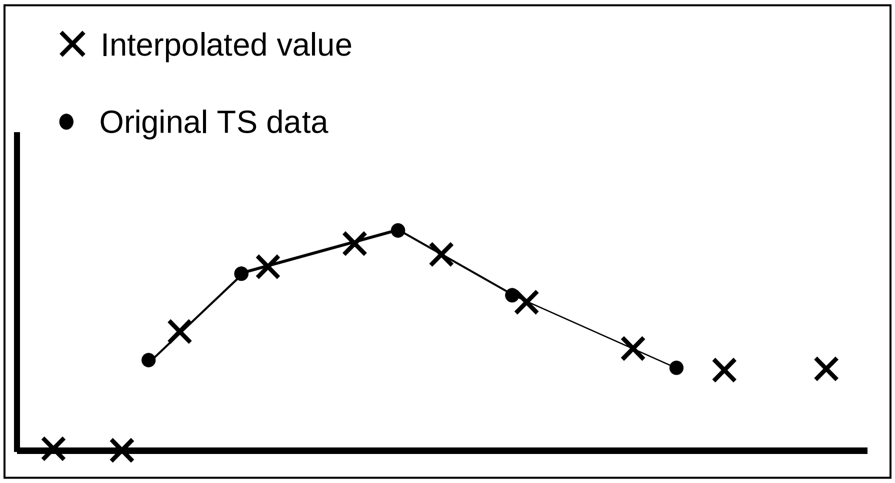
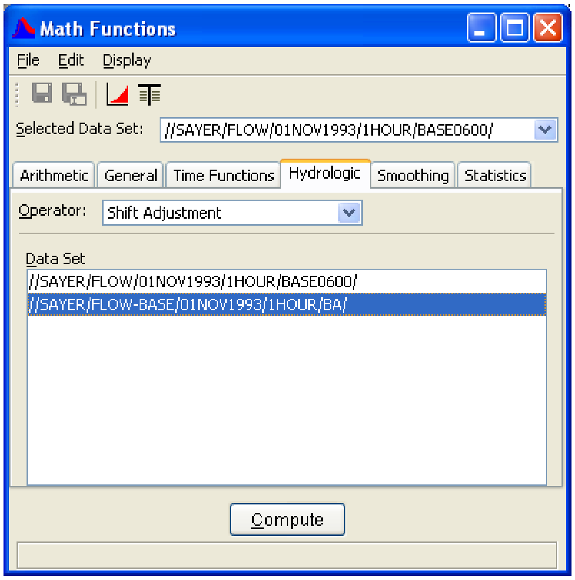


Figure 7.28 Interpolation of Time Series Values using Shift Adjustment Function

Both time series data sets may be regular or irregular interval. Interpolated points must be bracketed or coincident with valid (not missing) values in the original time series; otherwise the values are set as missing.

To generate a new time series of shift adjustments:

1. Choose the **Hydrologic** tab of the Math Functions screen and select the **Shift Adjustment** operator, as shown in Figure 7.29 (page 7-49).



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Shift Adjustment

Data Set
//SAYER/FLOW/01NOV1993/1HOUR/BASE0600/
//SAYER/FLOW-BASE/01NOV1993/1HOUR/BA/

Figure 7.29 Shift Adjustment Method

2. From the **Selected Data Set** list, select the *primary* time series data set. This data set has the *values* for the interpolation.
3. From the **Data Set** for time pattern list, which is located in the bottom part of the window, select the *second* time series data set. This data set has the *times* for the interpolation.
4. Click **Compute**.

7.6.9 Period Constants

The **Period Constants** function applies *values* in the *primary* time series data set to the *times* defined by a *second* time series data set. Both time series data sets may be regular or irregular interval. Values in the new time series are set according to:

$$ts1(j) \leq tsnew(i) < ts1(j+1)\,, \qquad TSNEW(i) = TS1(j)$$

where ts1 is the time in the primary time series, TS1 is the value in the primary time series, tsnew is the time in the new time series, TSNEW is the value in the new time series.

If times in the new time series precede the first data point in the primary time series, the value for these times is set to missing. If times in the new time series occur after the last data point in the primary time series, the value for these times is set to the value of the last point in the primary time series. Interpolation of values with the Period Constants function is shown in Figure 7.30 (page 7-50).

To generate a new time series of period constants:

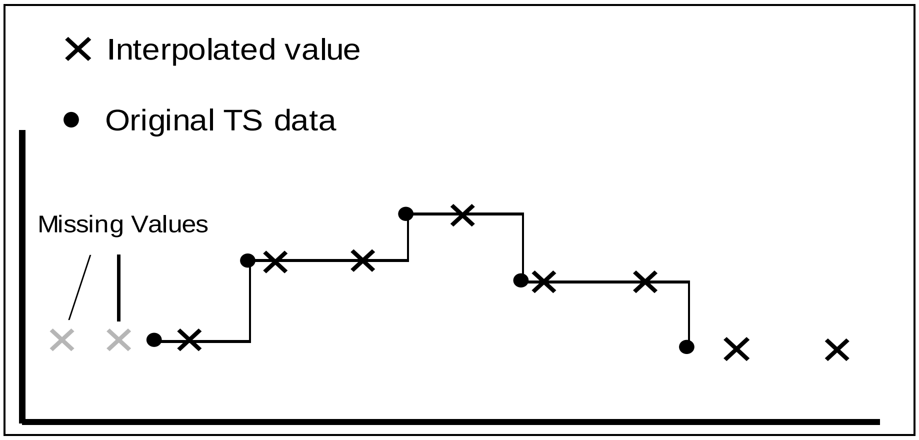


Figure 7.30 Interpolation of Time Series Values using Period Constants Function

1. Choose the **Hydrologic** tab of the Math Functions screen and select the **Period Constants** operator.
2. From the **Selected Data Set** list, select the *primary* time series data set. This data set has the *values* for the interpolation.
3. From the **Data Set** for time pattern list, which is located in the bottom part of the window, select the *second* time series data set. This data set has the *times* for the interpolation.
4. Click **Compute**.

7.6.10 Multiple Linear Regression

The **Multiple Linear Regression** function computes the multiple linear regression coefficients between a primary time series data set and a collection of independent time series data sets, and stores the regression coefficients in a new paired data set. This paired data set may be used with the Apply Multiple Linear Regression function to derive a new estimated time series (see Section 7.6.11)

For the general linear regression equation, a dependent variable, Y, may be computed from a set independent variables, Xn:

$$Y = B0 + B1*X1 + B2*X2 + B3*X3$$

where Bn are the linear regression coefficients.

For time series data, an estimate of the primary time series values may be computed from a set of independent time series data sets using regression coefficients such that:

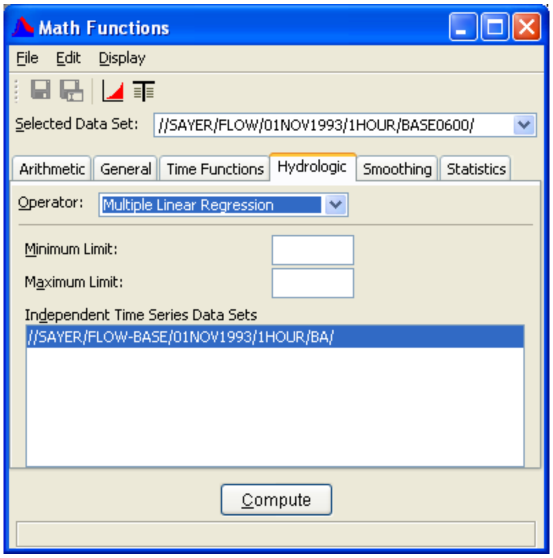
$$TsEstimate(t) = B0 + B1*TS1(t) + B2*TS2(t) + \dots + Bn*TSn(t)$$

where Bn are the set of regression coefficients and TSn are the time series data sets.

All the time series data sets must be regular interval and have the same time interval. The function filters the data to determine the time period common to all data sets and uses only those points in the regression analysis. For any given time, if a value is missing in any time series, no data for that time is processed. Optional minimum and maximum limits can be set to exclude values in the primary time series which fall outside a specified range.

To compute the set of multiple linear regression coefficients:

1. Choose the **Hydrologic** tab of the Math Functions screen and select the **Multiple Linear Regression** operator, as shown in Figure 7.31.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Multiple Linear Regression

Minimum Limit:

Maximum Limit:

Independent Time Series Data Sets

//SAYER/FLOW-BASE/01NOV1993/1HOUR/BA/

Figure 7.31 Multiple Linear Regression Method

2. From the **Selected Data Set** list, select primary time series data set. If the selected data set is not a time series data set, the remaining input boxes on the screen are unavailable.
3. From the **Independent Time Series Data Sets** list, select one or more of the independent time series data sets (by clicking, control-click or shift-click).
4. Click **Compute** to derive new a paired data set containing the linear regression coefficients.

7.6.11 Apply Multiple Linear Regression

The **Apply Multiple Linear Regression** function applies the multiple linear regression coefficients computed with the Multiple Linear Regression function (see Section 7.6.10). The coefficients, stored in a paired data set, are applied to a collection of independent time series data sets to derive a new estimated time series data set.

For time series data, an estimate of the primary time series values may be computed from a set of independent time series data sets using regression coefficients such that:

$$Y = B0 + B1*X1 + B2*X2 + B3*X3$$

where Bn are the linear regression coefficients.

For time series data, an estimate of the original time series values may be computed from a set of independent time series data using regression coefficients such that:

$$TsEstimate(t) = B0 + B1*TS1(t) + B2*TS2(t) + \ldots + Bn*TSn(t)$$

where Bn are the set of regression coefficients and TSn are the time series data sets.

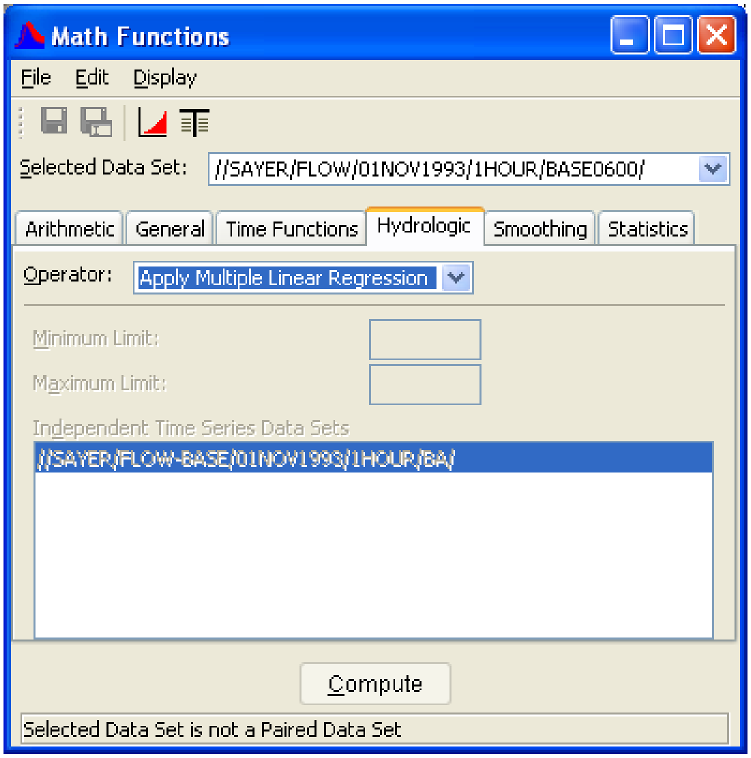
The number regression coefficients in the paired data set must be one more than the number of independent time series data sets. The collection of selected time series data sets should be in the same order as when the regression coefficients were computed. The displayed list of time series is sorted alphabetically in the dialog. The user should be aware if the names of the time series data sets used for coefficient generation are substantially different from the data names used in this function.

All the time series data sets must be regular interval and have the same time interval. The function filters the data to determine the time period common to all time series data sets and uses only those points in the regression analysis. For any given time, if a value is missing in any time series, the value in the resultant time series is set to missing. You can also set optional minimum and maximum value limits. Computed values in the resultant time series, which fall outside the specified range, are set to missing.

Names, parameter type and unit labels for the resultant time series data set are taken from the first time series data set. The "F part” (Version) in the new data set is set to "COMPUTED.”

To apply a set of multiple linear regression coefficients to derive a new time series data set:

1. Choose the **Hydrologic** tab of the Math Functions screen and select the **Apply Multiple Linear Regression** operator, as shown in Figure 7.32 (page 7-53).



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Apply Multiple Linear Regression

Independent Time Series Data Sets

//SAYER/FLOW-BASE/01NOV1993/1HOUR/BA/

Selected Data Set is not a Paired Data Set

Figure 7.32 Multiple Linear Regression Method

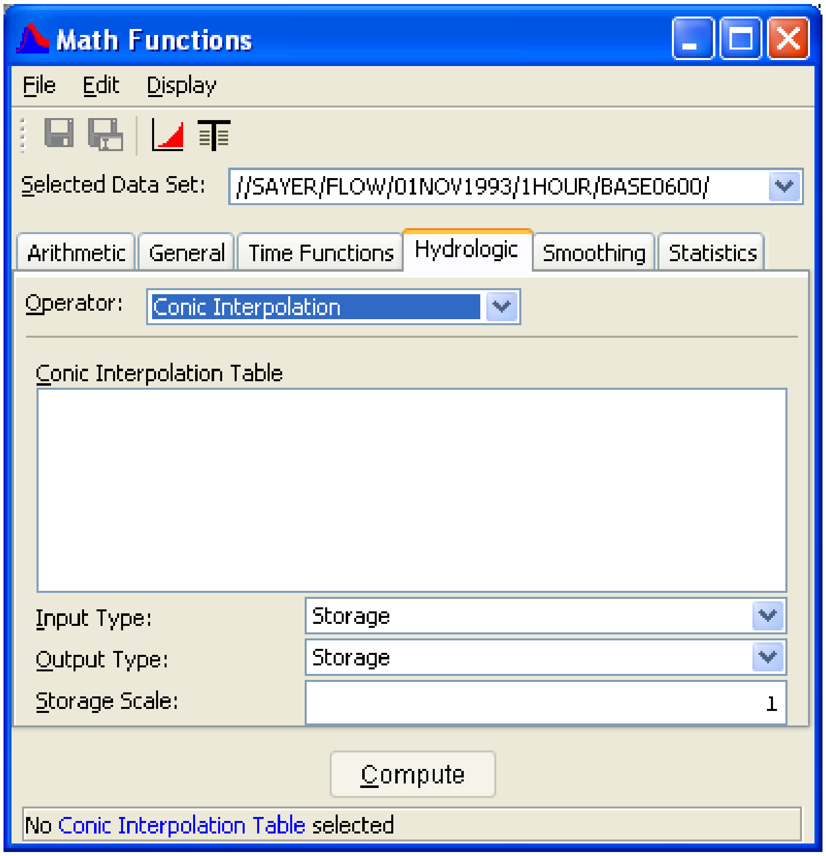
2. From the **Selected Data Set** list, select the paired data set containing the regression coefficients. If the selected data set is not a paired data set, the remaining input boxes on the screen are unavailable.
3. From the **Independent Time Series Data Sets** list, select one or more of the independent time series data sets. (You can select a single data set by clicking on it; select multiple contiguous data sets using **Shift+Click**, or select multiple, non-contiguous data sets using **Ctrl+Click**.)
4. Click **Compute**.

7.6.12 Conic Interpolation

The **Conic Interpolation** function transforms values in a time series data set by conic interpolation using an elevation-area table stored in a paired data set. The first value pair in the paired data set contains the initial conic depth and the initial storage volume at the first elevation (given in the next value pair). If the initial conic depth is missing, one is computed by the function. Values in the elevation-area table are stored in ascending order.

The Conic Interpolation function can interpolate a time series of elevation to derive a time series of storage or area. The function can interpolate a time series of storage to derive a time series of elevation or area. If the output time series is elevation, the output units are assigned from the x-units of the paired data set. If the output time series is area, the output units are assigned from the y-units of the paired data set. If the output time series is storage, the output units are undefined.

The appearance of the Math Functions screen for the Conic Interpolation function is shown in Figure 7.33. The Conic Interpolation function is accessed from the **Hydrologic** tab of the Math Functions screen.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Conic Interpolation

Input Type: Storage

Output Type: Storage

Storage Scale: 1

No Conic Interpolation Table selected

Figure 7.33 Screen for Conic Interpolation Function

To perform the conic interpolation of a of elevation or storage:

1. Choose the **Hydrologic** tab of the Math Functions screen and select the **Conic Interpolation** operator.
2. From the **Selected Data Set** list, select a time series data set. If the selected data set is not a time series, the remaining input boxes on the screen are unavailable.
3. From the **Conic Interpolation Table** list, select the paired data set containing the conic interpolation table. Only paired data sets will appear in this list, and only one data set at a time can be selected.
4. From the **Input Type** list, select the parameter type of the input time series data (Elevation or Storage).
5. From the **Output Type** list, select the desired output data type (Storage, Elevation or Area).

6. In the **Storage Scale** box, enter factor for the scale of the storage output values. By default this value is 1. The value is used to scale input (by multiplying) and output (by dividing) storage values. For example, if the area in the conic interpolation table is expressed in sq. ft., the storage scale could be set to 43560 to convert the storage output to acre-ft.
7. Click **Compute** to interpolate the selected time series data set.

7.6.13 Polynomial

The **Polynomial** function computes a polynomial transform of a regular or irregular interval time series data using the polynomial coefficients in a paired data set. Missing values in the input time series data remain missing in the resultant time series data set.

A new time series can be computed from an existing time series with the polynomial expression,

$$TS2\ (t) = B1^{*}\ TS1(t) + B2^{*}\ TS1(t)^{2} + ... + Bn^{*}\ TS1(t)^{n}$$

where Bn are the polynomial coefficients for term “n.”

Values for the polynomial coefficients are stored in the x-values of a paired data set. Before the above equation is applied, values in the input time series are adjusted by subtracting off the paired data “datum” value if defined. The x-units and parameter type from the paired data set are applied to the resultant time series data set.

To compute the polynomial transform of a selected time series data set:

1. Choose the **Hydrologic** tab of the Math Functions screen and select the **Polynomial** operator.
2. From the **Selected Data Set** list, select a time series data set. If the selected data set is not a time series, the remaining input boxes on the screen will be unavailable.
3. From the **Polynomial Coefficients** list, select the paired data set containing the polynomial coefficients. Only paired data sets will appear in this list, and you may select only one data set at a time.
4. Click **Compute** to perform the polynomial transform of the selected time series data set.

7.6.14 Polynomial with Integral

The **Polynomial with Integral** function computes a polynomial transform with integral of a regular or irregular interval time series data using the polynomial coefficients in a paired data set. Missing values in the input time series data remain missing in the resultant time series data set.

The polynomial transform coefficients are integrated, with the new time series values computed from an existing time series by the expression,

$$TS2\ (t) = B1 * TS1(t)^{2}/2 + B2*TS1(t)^{3}/3 + \ldots + Bn * TS1(t)^{n+1}/(n+1)$$

where Bn are the polynomial coefficients for term "n".

Values for the polynomial coefficients are stored in the x-values of a paired data set. Before the above equation is applied, values in the input time series are adjusted by subtracting off the paired data “datum” value if defined. The x-units and parameter type from the paired data set are applied to the resultant time series data set.

To compute the polynomial transform with integral of a selected time series data set:

1. Choose the **Hydrologic** tab of the Math Functions screen and select the **Polynomial with Integral** operator.
2. From the **Selected Data Set** list, select a time series data set. If the selected data set is not a time series, the remaining input boxes on the screen will be unavailable.
3. From the **Polynomial Coefficients** list, select the paired data set containing the polynomial coefficients. Only paired data sets will appear in this list, and you may select only one data set at a time.
4. Click **Compute**.

7.6.15 Flow Accumulator Gage Processor

The **Flow Accumulator Gage Processor** function computes the time series period-average flows from a flow accumulator type gage. Accumulator gage data consists of time series data sets of accumulated flow and counts. The two input time series data sets must match times exactly.

A new time series data set is derived from the time series of flow and counts by:

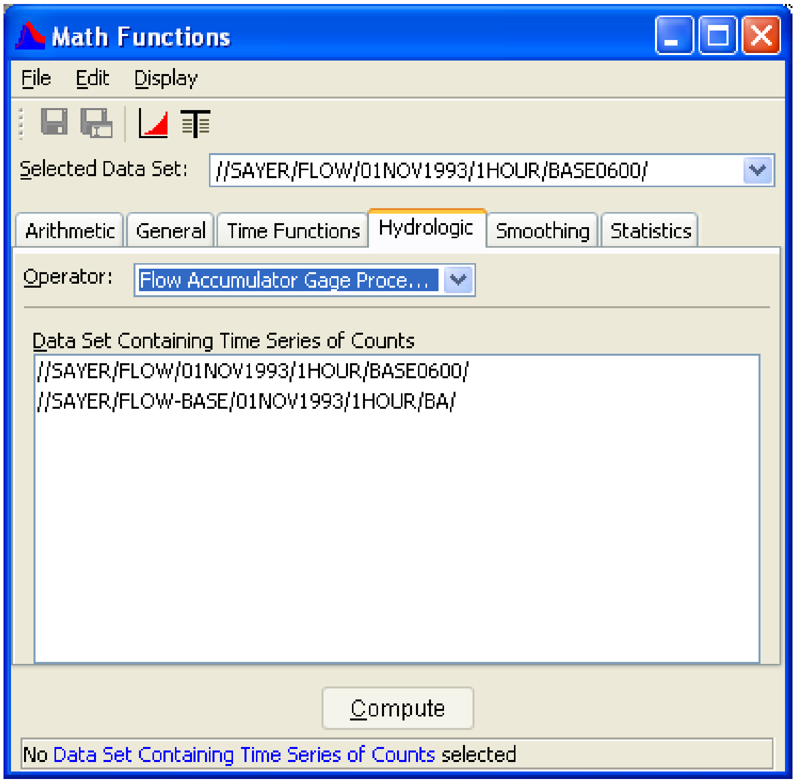
$$TsNew(t) = (TsAccFlow(t) - TsAccFlow(t-1)) / (TsCount(t) - TsCount(t-1))$$

where TsAccFlow is the gage accumulated flow time series and TsCount is the gage time series of counts.

In the above equation, if TsAccFlow(t), TsAccFlow(t-1), TsCount(t) or TsCount(t-1) are missing, TsNew(t) is set to missing. The new time series is assigned the data type "PER-AVER”.

To process the flow accumulator type data:

1. Choose the **Hydrologic** tab of the Math Functions screen and select the **Flow Accumulator Gage Processor** operator, as shown in Figure 7.34.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Flow Accumulator Gage Proce...

Data Set Containing Time Series of Counts
//SAYER/FLOW/01NOV1993/1HOUR/BASE0600/
//SAYER/FLOW-BASE/01NOV1993/1HOUR/BA/

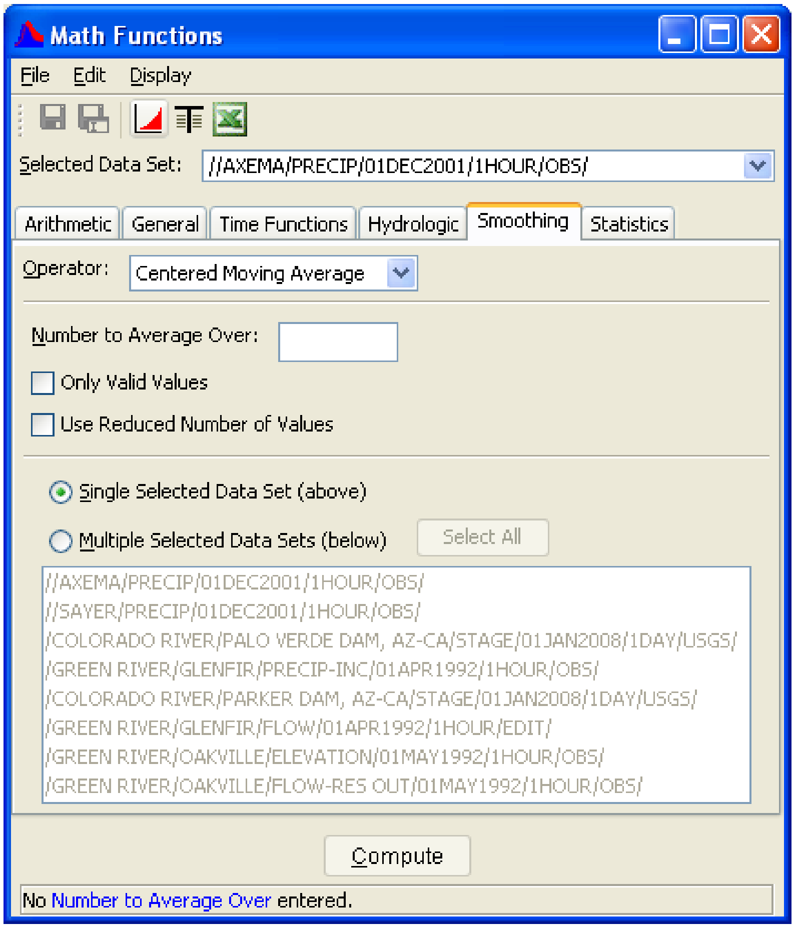
No Data Set Containing Time Series of Counts selected

Figure 7.34 Flow Accumulator Gage Processor Method

2. From the **Selected Data Set** list, select a time series data set.
3. From the **Data Set Containing Time Series of Counts** list, select the time series data set of counts.
4. Click **Compute**.

7.7 Smoothing Functions

The **Smoothing** tab (see Figure 7.35, page 7-58) contains the following functions: **Centered Moving Average**, **Olympic Smoothing Average**, and **Forward Moving Average**. The proceeding segments for this chapter section contain detailed information on each of these options.



Selected Data Set: //AXEMA/PRECIP/01DEC2001/1HOUR/OBS/

Operator: Centered Moving Average

Number to Average Over:

□ Only Valid Values

□ Use Reduced Number of Values

● Single Selected Data Set (above)

○ Multiple Selected Data Sets (below)

```
//AXEMA/PRECIP/01DEC2001/1HOUR/OBS/
//SAYER/PRECIP/01DEC2001/1HOUR/OBS/
/COLORADO RIVER/PALO VERDE DAM, AZ-CA/STAGE/01JAN2008/1DAY/USGS/
/GREEN RIVER/GLENFIR/PRECIP-INC/01APR1992/1HOUR/OBS/
/COLORADO RIVER/PARKER DAM, AZ-CA/STAGE/01JAN2008/1DAY/USGS/
/GREEN RIVER/GLENFIR/FLOW/01APR1992/1HOUR/EDIT/
/GREEN RIVER/OAKVILLE/ELEVATION/01MAY1992/1HOUR/OBS/
/GREEN RIVER/OAKVILLE/FLOW-RES OUT/01MAY1992/1HOUR/OBS/
```

No Number to Average Over entered.

Figure 7.35 Math Functions - Smoothing Tab

7.7.1 Centered Moving Average

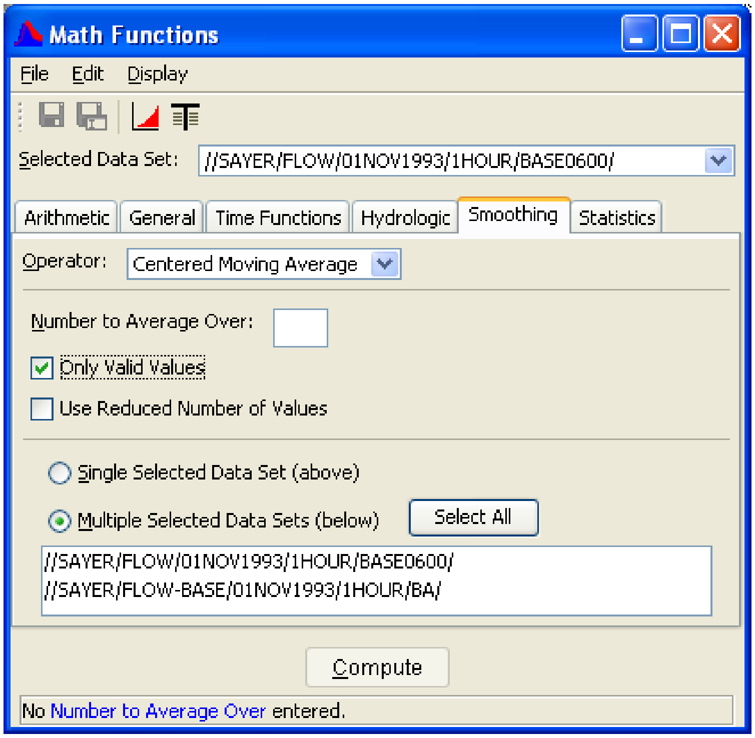
The **Centered Moving Average** function computes a centered moving average of "NAVG" values for regular or irregular interval time series data. The number of values to average over ("NAVG") must be an odd integer greater than two.

The **Only Valid Values** option pertains to when the averaging interval contains missing values. If checked, the option sets the value in the resultant data set to missing. If unselected, the option computes a smoothed value from the valid values in the interval.

The **Use Reduced Number of Values** option pertains to the first and last NAVG/2 values in the resultant data. If selected, the option computes smoothed values at the beginning and ending of the data set from a reduced number of values of the original data set. If unselected, the option sets the first and last NAVG/2 values in the resultant data to missing.

To compute the centered moving average for time series data:

1. Choose the **Smoothing** tab of the Math Functions screen and select the **Centered Moving Average** operator, as shown in Figure 7.36.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Centered Moving Average

Number to Average Over:

☑ Only Valid Values

□ Use Reduced Number of Values

○ Single Selected Data Set (above)

● Multiple Selected Data Sets (below)

//SAYER/FLOW/01NOV1993/1HOUR/BASE0600/
//SAYER/FLOW-BASE/01NOV1993/1HOUR/BA/

No Number to Average Over entered.

Figure 7.36 Centered Moving Average

2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. In the **Number to Average Over** box, enter a value to set the number of values for the moving average. This number must be an odd number greater than two (2).
4. Select the **Only Valid Values** box to compute a smoothed value only if all the values in the averaging interval are valid (no missing values). If there are one or more missing values, the value in the resultant time series data is then set to missing. If the box is not checked, the value in the resultant time series data is computed using the remaining valid values in the averaging interval.
5. Select the **Use Reduced Number of Values** box to compute a moving average from a reduced number of time series values near the beginning and ending of a time series. At these locations, the moving average interval becomes truncated and there are not “Number to Average Over” values for averaging. If unchecked, values in the resultant time series data at these locations will be set to missing.
6. Click **Compute** to perform a centered moving average smoothing of the selected data.

7.7.2 Olympic Smoothing Average

The **Olympic Smoothing Average** function uses the same smoothing scheme as the Centered Moving Average function (see Section 7.7.1), except the minimum and maximum values in the averaging interval are excluded from the computation. The input time series may be regular or irregular interval. The number of values to average over ("NAVG") must be an odd integer greater than two.

The **Only Valid Values** option pertains to when the averaging interval contains missing values. If checked, the option sets the value in the resultant data to missing. If unselected, the option computes a smoothed value from the valid values in the interval.

The **Use Reduced Number of Values** option pertains to the first and last NAVG/2 values in the resultant data. If selected, the option computes smoothed values at the beginning and ending of the data from a reduced number of values of the original data. If unselected, the option sets the first and last NAVG/2 values in the resultant data to missing.

The span of the averaging interval, "NAVG", must be an odd integer greater than two.

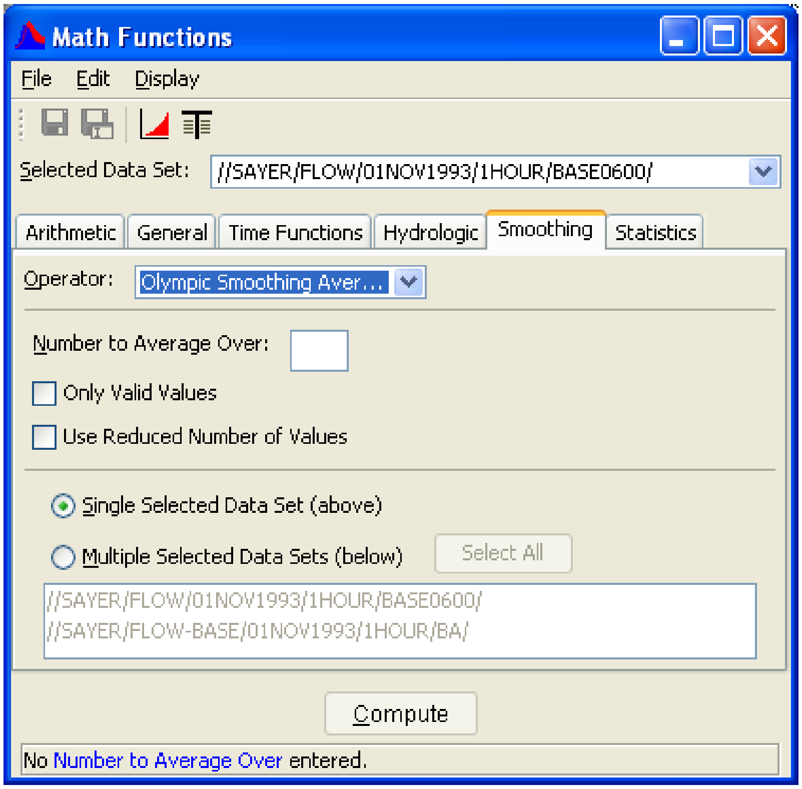
Under two conditions there are not NAVG valid values over the averaging interval: if there are missing values in the averaging interval, or if the smoothed point is within NAVG/2 values of the beginning or ending of the time series.

For the first condition, if the averaging interval contains any missing values, the option is to set the value in the resultant data set to missing, or compute a smoothed value from the remaining valid values in the interval.

For the second condition, the option is to set the first and last NAVG/2 values in the resultant data set to missing, or compute smoothed values at the beginning and ending of the data set from a reduced number of values.

To compute the Olympic smoothing average for a time series data set:

1. Choose the **Smoothing** tab of the Math Functions screen and select the **Olympic Smoothing Average** operator, as shown in Figure 7.37 (page 7-61).
2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Olympic Smoothing Aver...

Number to Average Over:

□ Only Valid Values

□ Use Reduced Number of Values

● Single Selected Data Set (above)

○ Multiple Selected Data Sets (below)

//SAYER/FLOW/01NOV1993/1HOUR/BASE0600/
//SAYER/FLOW-BASE/01NOV1993/1HOUR/BA/

No Number to Average Over entered.

Figure 7.37 Olympic Smoothing Average

3. In the **Number to Average Over** box, enter a value to set the number of values for the averaging interval. This number must be an odd integer greater than two.
4. Select the **Only Valid Values** box to compute a smoothed value only if all the values in the averaging interval are valid (no missing values). If there are one or more missing values, the value in the resultant time series data is then set to missing. If the box is not checked, the value in the resultant time series data is computed using the remaining valid values in the averaging interval.
5. Select the **Use Reduced Number of Values** box to compute a moving average from a reduced number of time series values near the beginning and ending of a time series. At these locations, the moving average interval becomes truncated and there are not "Number to Average Over" values for averaging. If unchecked, values in the resultant time series data at these locations will be set to missing.
6. Click **Compute** to perform an Olympic smoothing of the selected data.

7.7.3 Forward Moving Average

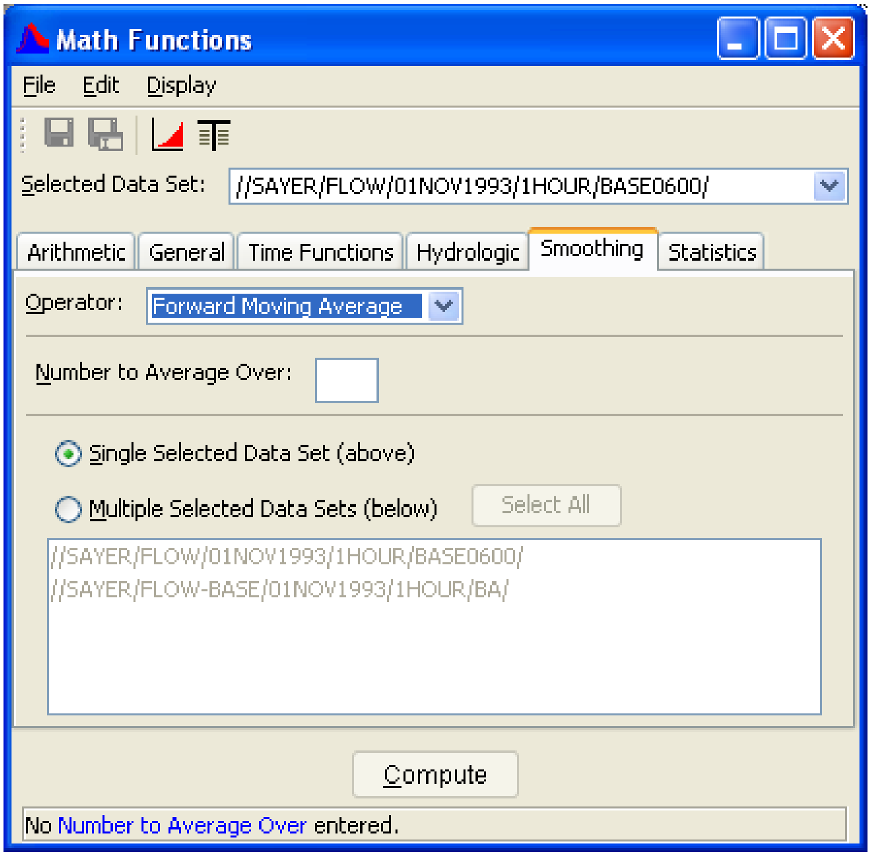
The **Forward Moving Average** function computes a moving average of the last "NAVG" values for regular or irregular interval time series data.

The number of values for averaging, "NAVG", must be greater than two. The first NAVG-1 values in the resultant time series are set to missing.

If the averaging interval contains a missing value, the smoothed value is computed from the remaining valid values in the interval. However, if there are less than two valid values in the interval, the value in the resultant data is set to missing.

To compute the forward moving average for time series data:

1. Choose the **Smoothing** tab of the Math Functions screen and select the **Forward Moving Average** operator, as shown in Figure 7.38.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Operator: Forward Moving Average

Number to Average Over:

● Single Selected Data Set (above)

○ Multiple Selected Data Sets (below)

//SAYER/FLOW/01NOV1993/1HOUR/BASE0600/
//SAYER/FLOW-BASE/01NOV1993/1HOUR/BA/

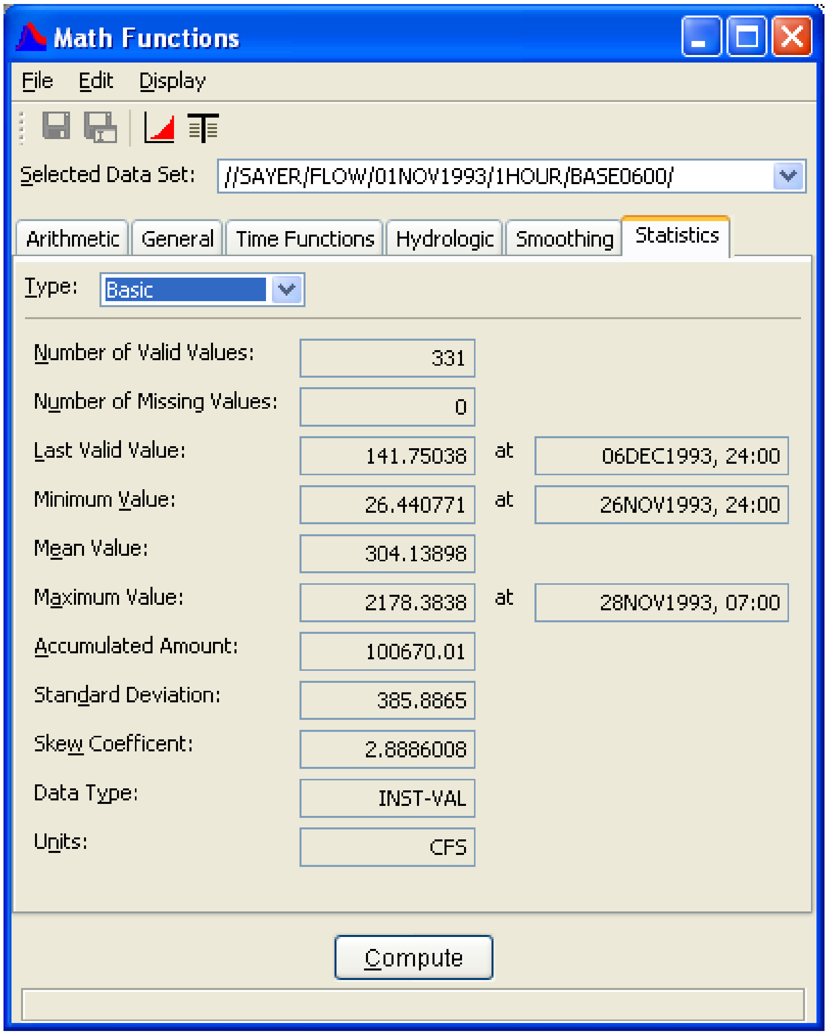
No Number to Average Over entered.

Figure 7.38 Forward Moving Average

2. Select a data set to apply the function from the **Selected Data Set** pull-down list at the top of the screen or multiple data sets from the list in the lower portion of the screen. If you include the data set selected in the top **Selected Data Set**, the operation on that set will only be done once.
3. In the **Number to Average Over** box, enter the number of values in the averaging interval. This number must be greater than two (2).
4. Click **Compute**.

7.8 Statistics Functions

The **Statistics** tab (Figure 7.39) contains the following functions: **Basic** (statistics), **Linear Regression**, **Cyclic Analysis**, **Duration Analysis**, and **Frequency Plot**.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Type: Basic

Number of Valid Values: 331

Number of Missing Values: 0

Last Valid Value: 141.75038 at 06DEC1993, 24:00

Minimum Value: 26.440771 at 26NOV1993, 24:00

Mean Value: 304.13898

Maximum Value: 2178.3838 at 28NOV1993, 07:00

Accumulated Amount: 100670.01

Standard Deviation: 385.8865

Skew Coefficent: 2.8886008

Data Type: INST-VAL

Units: CFS

Figure 7.39 Math Functions - Statistics Tab

7.8.1 Basic

The **Basic** function computes the basic statistical values for a regular or irregular interval time series data set. The statistical and informational values displayed are:

Number of valid values
Number of missing values
Last valid value and date and time

Minimum value and date and time
Mean value
Maximum value and date and time
Accumulated value for the time series
Standard deviation
Skew coefficient
Data type ("INST-VAL", "INST-CUM", "PER-AVER", "PER-CUM")
Data Units ("ft", "cfs", etc.)

To compute the basic statistical parameters for a time series data set:

1. Choose the **Statistics** tab of the Math Functions screen and select the **Basic** type.
2. From the **Selected Data Set** list, select a time series data set.

The statistics are displayed for the time series data set once the data set is selected.

7.8.2 Linear Regression

The **Linear Regression** function computes the linear regression and other correlation coefficients between two time series data sets. Values in the primary time series data set and the second time series data set are matched by time to form data pairs for the correlation analysis. Missing values are ignored. Times for the two time series data sets must match exactly. The data sets may be either regular or irregular interval time series data.

The correlations statistics computed by the function are:

Number of Valid Values
Regression Constant
Regression Coefficient
Determination Coefficient
Standard Error of Regression
Adjusted Determination Coefficient
Adjusted Standard Error of Regression

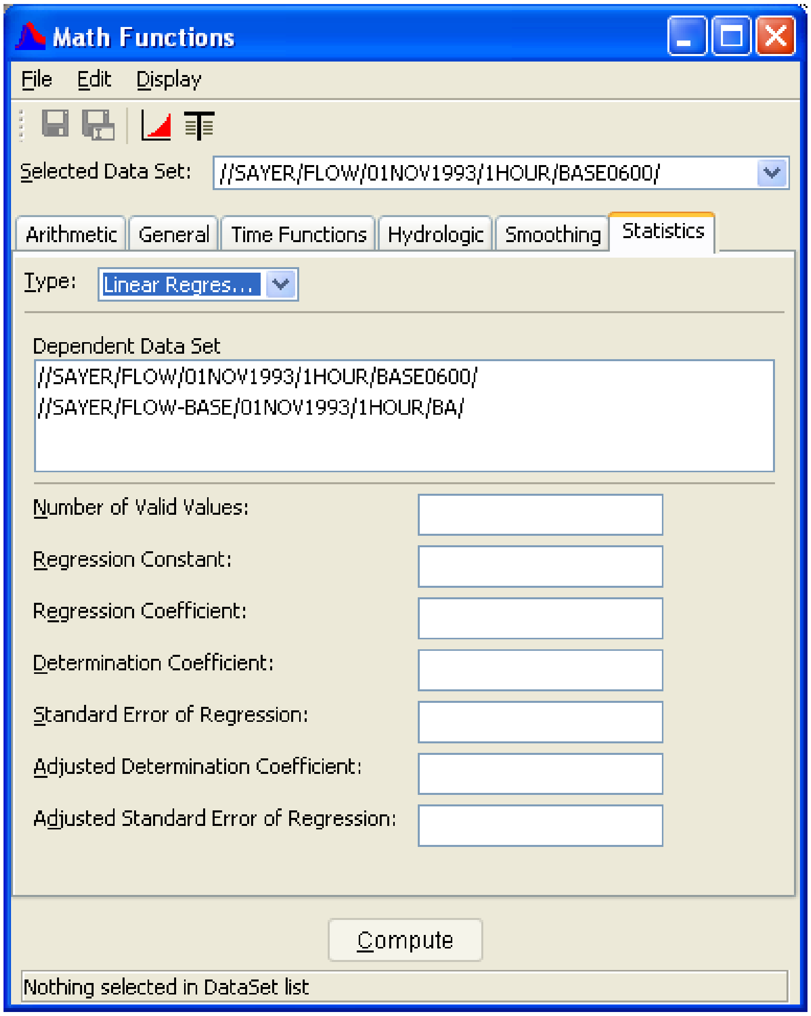
The primary time series data set forms the values of the independent variable (x-values), while values of the second time series data set comprise the dependent variable (y-values). The linear regression coefficients express how values in the second data set can be derived from values in the primary data set:

$$TS2(t) = a + b * TS1(t)$$

where "a" is the regression constant and "b" the regression coefficient.

To compute the linear regression and correlation coefficients between two time series data sets:

1. Choose the **Statistics** tab of the Math Functions screen and select the **Linear Regression** type, as shown in Figure 7.40.



Selected Data Set: //SAYER/FLOW/01NOV1993/1HOUR/BASE0600/

Type: Linear Regres...

Dependent Data Set
//SAYER/FLOW/01NOV1993/1HOUR/BASE0600/
//SAYER/FLOW-BASE/01NOV1993/1HOUR/BA/

Number of Valid Values:
Regression Constant:
Regression Coefficient:
Determination Coefficient:
Standard Error of Regression:
Adjusted Determination Coefficient:
Adjusted Standard Error of Regression:

Nothing selected in DataSet list

Figure 7.40 Linear Regression Statistics

2. From the **Selected Data Set** list, select the primary time series data set for analysis. Time series values from this data set form the independent variable (x-values) of the correlation analysis.
3. From the **Dependent Data Set** list, select the second time series data set for analysis. Time series values from this data set will comprise the dependent variable (y-values) of the correlation analysis.

The correlation statistics are computed automatically once the two data sets are selected.

7.8.3 Cyclic Analysis

The **Cyclic Analysis** function derives a set of cyclic statistics from a regular interval time series data set. The time series data set must have a time interval of "1HOUR", "1DAY" or "1MONTH". The function sorts the time series values into statistical "bins" relevant to the time interval. Values for the 1HOUR interval data are sorted into twenty-four bins representing the hours of the day, 0100 to 2400. The 1DAY interval data is apportioned to 365 bins for the days of the year. The 1MONTH interval data is sorted into twelve bins for the months of the year.

The format of the resultant data sets is as a "pseudo" time series for the year 3000. For example, the cyclic analysis of one month of hourly interval data will produce pseudo time series data sets having twenty-four hourly values for the day January 1, 3000. If the statistical parameter is the “maximum” value, then the twenty-four values represent the maximum value occurring at that hour of the day in the original time series. The cyclic analysis of daily interval data will produce pseudo time series data sets having 365 daily values for the year 3000. The cyclic analysis of monthly interval data will result in pseudo time series data sets having twelve monthly values for the year 3000.

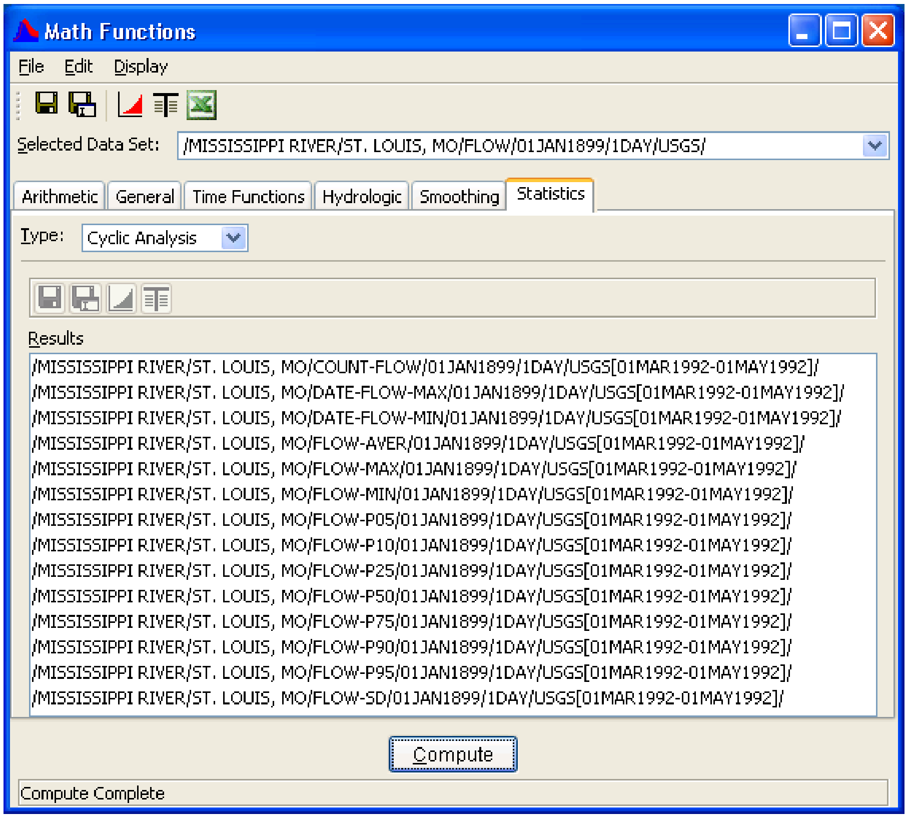
Fourteen pseudo time series data sets are derived by the cyclic analysis function for the following statistical parameters:

Number of values processed for each time interval
Maximum value
Time of maximum value
Minimum value
Time of minimum value
Average value
Probability exceedence percentiles for 5%, 10%, 25%, 50% (median value), 75%, 90%, and 95%
Standard deviation

The appearance of the Math Functions screen for the Cyclic Analysis function is shown in Figure 7.41 (page 7-67).

To compute the cyclic analysis of a time series data set:

1. Choose the **Statistics** tab of the Math Functions screen and select the **Cyclic Analysis** type.
2. From the **Selected Data Set** list, select a time series data set for cyclic analysis.
3. Click **Compute**.



Selected Data Set: /MISSISSIPPI RIVER/ST. LOUIS, MO/FLOW/01JAN1899/1DAY/USGS/

Type: Cyclic Analysis

Results

/MISSISSIPPI RIVER/ST. LOUIS, MO/COUNT-FLOW/01JAN1899/1DAY/USGS[01MAR1992-01MAY1992]/
/MISSISSIPPI RIVER/ST. LOUIS, MO/DATE-FLOW-MAX/01JAN1899/1DAY/USGS[01MAR1992-01MAY1992]/
/MISSISSIPPI RIVER/ST. LOUIS, MO/DATE-FLOW-MIN/01JAN1899/1DAY/USGS[01MAR1992-01MAY1992]/
/MISSISSIPPI RIVER/ST. LOUIS, MO/FLOW-AVER/01JAN1899/1DAY/USGS[01MAR1992-01MAY1992]/
/MISSISSIPPI RIVER/ST. LOUIS, MO/FLOW-MAX/01JAN1899/1DAY/USGS[01MAR1992-01MAY1992]/
/MISSISSIPPI RIVER/ST. LOUIS, MO/FLOW-MIN/01JAN1899/1DAY/USGS[01MAR1992-01MAY1992]/
/MISSISSIPPI RIVER/ST. LOUIS, MO/FLOW-P05/01JAN1899/1DAY/USGS[01MAR1992-01MAY1992]/
/MISSISSIPPI RIVER/ST. LOUIS, MO/FLOW-P10/01JAN1899/1DAY/USGS[01MAR1992-01MAY1992]/
/MISSISSIPPI RIVER/ST. LOUIS, MO/FLOW-P25/01JAN1899/1DAY/USGS[01MAR1992-01MAY1992]/
/MISSISSIPPI RIVER/ST. LOUIS, MO/FLOW-P50/01JAN1899/1DAY/USGS[01MAR1992-01MAY1992]/
/MISSISSIPPI RIVER/ST. LOUIS, MO/FLOW-P75/01JAN1899/1DAY/USGS[01MAR1992-01MAY1992]/
/MISSISSIPPI RIVER/ST. LOUIS, MO/FLOW-P90/01JAN1899/1DAY/USGS[01MAR1992-01MAY1992]/
/MISSISSIPPI RIVER/ST. LOUIS, MO/FLOW-P95/01JAN1899/1DAY/USGS[01MAR1992-01MAY1992]/
/MISSISSIPPI RIVER/ST. LOUIS, MO/FLOW-SD/01JAN1899/1DAY/USGS[01MAR1992-01MAY1992]/

Compute Complete

Figure 7.41 Screen for Cyclic Analysis Function

Once the compute is performed, the resultant 14-pseudo time series data sets appear in **Results** list on the screen (Figure 7.42). One or more data sets in this list may be selected (by clicking, control-click or shift-click) for saving to file, plotting or tabulation by using the **Save** button, **SaveAs** button, the **Plot** button or the **Tabulate** button from the toolbar located immediately above the **Results** list. An example showing the data plotted is shown in Figure 7.42.

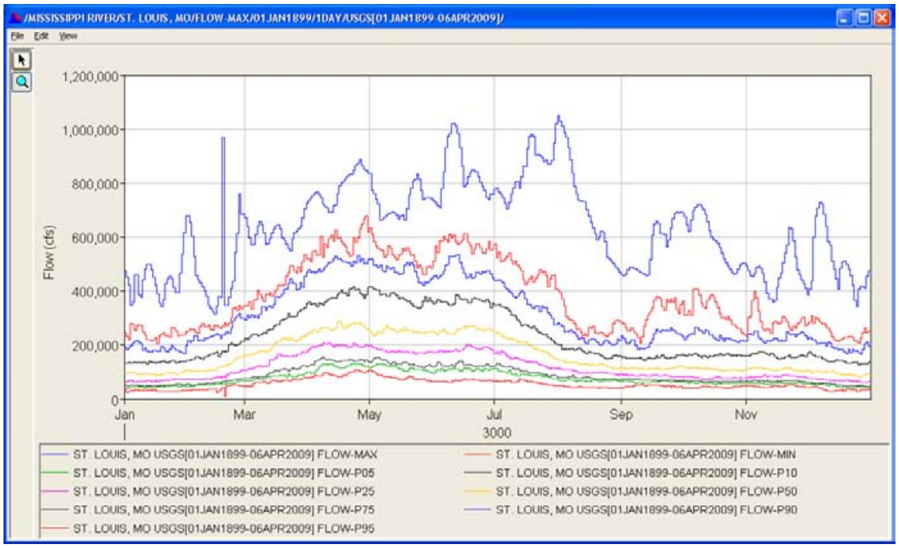


Figure 7.42 Plot of Data Set Results from Cyclic Analysis

7.8.4 Duration Analysis

The **Duration Analysis** function computes the duration curve for a regular interval time series data set and stores the results in a new paired data set. Flood duration curves are useful in assessing the general low flow characteristics of a stream. For example, if the lower end drops rapidly, the stream has low ground-water storage and low or no sustained flow. The overall slope of the curve is an indication of the flow variability in the stream. Refer to EM 1110-2-1415 Chapter 2 (USACE, 1993) for a description of duration analysis. Duration analysis is usually applied to daily flow or elevation or similar data. This function is not Volume Duration Frequency Analysis; see the program HEC-SSP for this capability.

HEC-DSSVue provides two techniques for compute duration curves. The first method develops the curve by ranking all the data and then extracting points along that curve. The second technique segregates values into "bins" and then plots the cumulative amount. This technique was used by the HEC-STATS program and was developed to accommodate the small amount of memory that computers used to have, or when the analysis was done by hand. Because memory limitations have been removed, the first technique is more accurate and is recommended.

The data to be analyzed must have a regular time interval of "1DAY", "1WEEK", "TRI-MONTH", "SEMI-MONTH", "1MON" or "1YEAR". Typically the time series data set for the duration analysis spans multiple years. The time series data values are first sorted by time of the year into periods of "Annual", "Quarterly", "Monthly" or "Other Defined". For an "Annual" period type, all data are assigned to a continuous single duration period. For a “Quarterly” duration period type, the data values are sorted by quarter year: 1 January to 31 March, 1 April to 30 June, 1 July to 30 September, and 01 October to 31 December. For the "Monthly" duration period type, data values are sorted by month. The "Other Defined" duration period type allows arbitrary duration periods to be defined and used.

For the standard duration analysis technique, the data values within a duration period are ranked (ordered by descending value). The ordered values form the y-values for a new paired data set. The x-values represent the percent of time exceeded computed by:

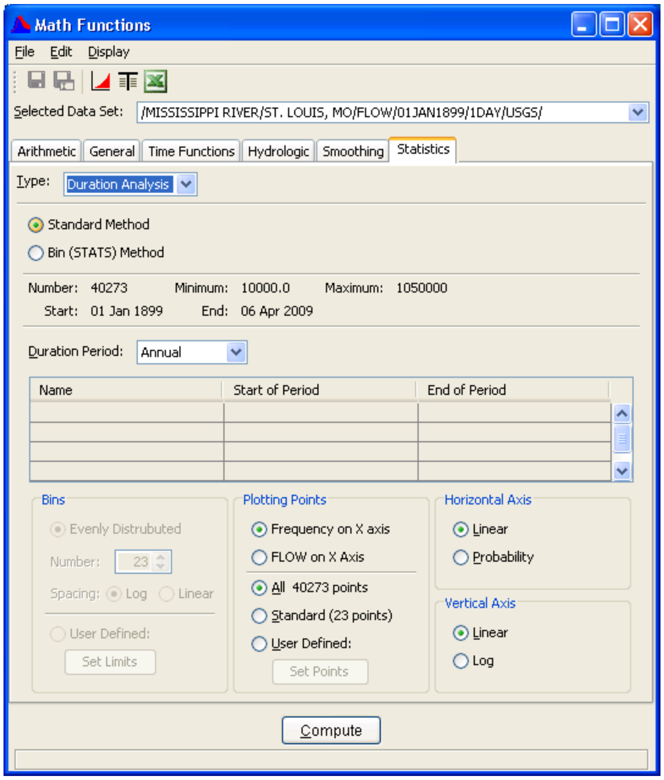
E = $100 * [M /(n+1)]$ (Weibull plotting positions), percent of the time the value is equaled or exceeded
M = the rank position of the value
n = number of values.

A curve is computed for each duration period. Thus for a duration period type "Quarterly", the paired data set has four curves with the curve labels "Jan-Mar", "Apr-Jun", "Jul-Sep" and "Oct-Dec".

The set of x-values generated by the duration analysis are dependent upon the number of values in the time series. Typically all values should be used for plotting the curve. However, this may become unwieldy when presented in a table form, so values may be a standard set of percentages, for example: 1%, 2%, 5%, 10 %, etc. If desired, the Duration Analysis function can sample the analysis results to a standard set of frequency points, or to a set of evenly distributed x-values. Using log spacing provides more clarity of the curve at its ends. User defined points can be specified by entering those points in a table after selecting the User Defined button.

To compute the duration analysis for a regular interval time series:

1. Select the data set from the main HEC-DSSVue screen and then select the **Math Functions** button or Math Functions from the **Tools** menu.
2. Choose the **Statistics** tab of the Math Functions screen and select the **Duration Analysis** type, as shown in Figure 7.43.


Math Functions
File Edit Display
Arithmetic General Time Functions Hydrologic Smoothing Statistics
Compute

Selected Data Set: /MISSISSIPPI RIVER/ST. LOUIS, MO/FLOW/01JAN1899/1DAY/USGS/

Type: Duration Analysis

● Standard Method
○ Bin (STATS) Method

Number: 40273 Minimum: 10000.0 Maximum: 1050000
Start: 01 Jan 1899 End: 06 Apr 2009

Duration Period: Annual

Name	Start of Period	End of Period

Bins
● Evenly Distrubuted
Number: 23
Spacing: ● Log ○ Linear
○ User Defined:
Set Limits

Plotting Points
● Frequency on X axis
○ FLOW on X Axis
● All 40273 points
○ Standard (23 points)
○ User Defined:
Set Points

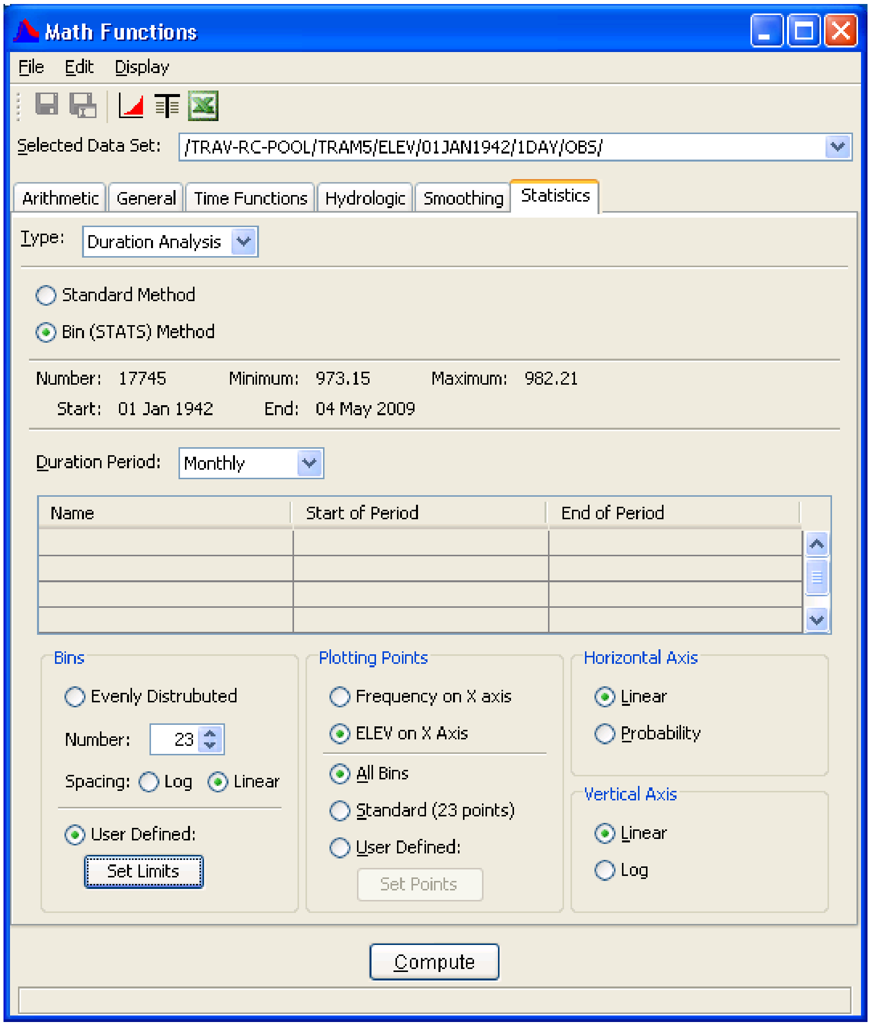
Horizontal Axis
● Linear
○ Probability

Vertical Axis
● Linear
○ Log

Figure 7.43 Duration Analysis Screen for Standard Method

3. Select the time series data set to apply the function from the **Selected Data Set** pull down list.

4. Select either the **Standard Method** or **Bin (STATS) Method** for the analysis technique you wish to use.
5. Set the duration period type using the **Duration Period** pull down list.
6. If the duration period type "Other" is selected, the duration periods are defined in the **Duration Period** table. Dates entered for the "Start of Period" and "End of Period" is of the form, "05May". The duration period "Name" is assigned to the curve label in the paired data set generated by the duration analysis function.
7. If the Bin method is selected, choose the number of bins to use, and if those bins should be spaced log arithmetically or linearly between the minimum and maximum values. Or select **User Defined** bins and enter values into the table by pressing the **Set Limits** button, as shown in Figure 7.44. User defined limits will provide greater flexibility to select the parameter values desired.



Selected Data Set: /TRAV-RC-POOL/TRAM5/ELEV/01JAN1942/1DAY/OBS/

Type: Duration Analysis

○ Standard Method

● Bin (STATS) Method

Number: 17745 Minimum: 973.15 Maximum: 982.21

Start: 01 Jan 1942 End: 04 May 2009

Duration Period: Monthly

Name	Start of Period	End of Period

Bins

○ Evenly Distrubuted

Number: 23

Spacing: ○ Log ● Linear

● User Defined:

Plotting Points

○ Frequency on X axis

● ELEV on X Axis

● All Bins

○ Standard (23 points)

○ User Defined:

Horizontal Axis

● Linear

○ Probability

Vertical Axis

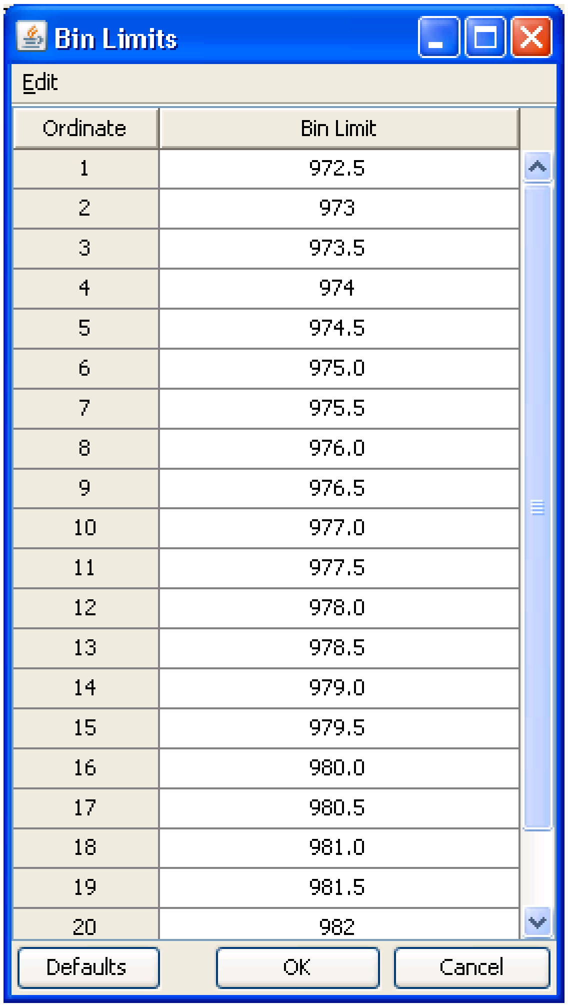
● Linear

○ Log

Figure 7.44 Duration Analysis Screen for Bin Method

8. The values provided in the table are either those computed using the number of bins and spacing method selected before, or those

saved from the previous run. To reset the values to the default ones, press the **Defaults** button. This will use the number of values selected to compute the default values. An example of the **Bin Limits** table is shown in Figure 7.45.



Ordinate	Bin Limit
1	972.5
2	973
3	973.5
4	974
5	974.5
6	975.0
7	975.5
8	976.0
9	976.5
10	977.0
11	977.5
12	978.0
13	978.5
14	979.0
15	979.5
16	980.0
17	980.5
18	981.0
19	981.5
20	982

Figure 7.45 User Defined Bin Limits Table

9. Select whether you want the Frequency to be on the X axis or the Parameter on the X axis. Typically Frequency is given on the X axis. If you display the results in tabular form, you may want to select the parameter for the X axis.
10. The number of duration points retained for plotting and tabulation is controlled by the **Plotting Points** options. Select the **All Bins** radio button to retain all the ranked computed duration points. Select the **Standard** radio button to interpolate the results to the traditional set of 23 log points. You may choose your own number of points by selecting the **User Defined** radio button and entering the desired number of interpolation points in the box to the right. Select either **Log** or **Linear** spacing on where to sample those points.

11. The **Horizontal and Vertical Axis** options control the scales for plotting the duration analysis curves. The **Linear** scale radio button selects a linear plot scale for the both the x and y axes. The **Probability** scale radio button selects a probability scale for the x-axis and a log scale for the y-axis.
12. Click **Compute**. **Note:** any change to the plot configuration requires a re-compute of the data.
13. Select the plot button to view the computed duration curve(s). An example of resulting plots are shown in Figure 7.46 and Figure 7.47.

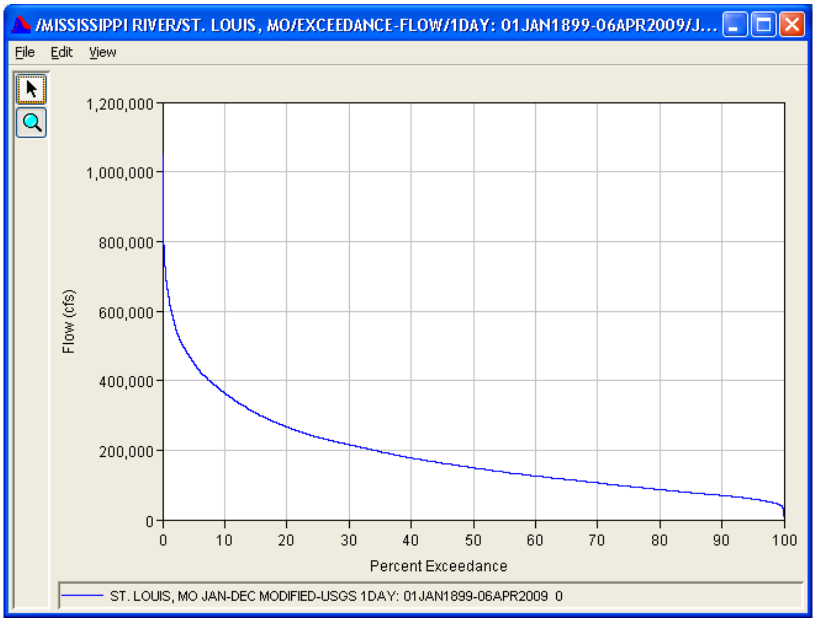

/MISSISSIPPI RIVER/ST. LOUIS, MO/EXCEEDANCE-FLOW/1DAY: 01JAN1899-06APR2009/J...
File
Edit
View
1,200,000
1,000,000
800,000
600,000
400,000
200,000
0
Flow (cfs)
0
10
20
30
40
50
60
70
80
90
100
Percent Exceedance
ST. LOUIS, MO JAN-DEC MODIFIED-USGS 1DAY: 01JAN1899-06APR2009 0

Figure 7.46 Linear Vertical and Horizontal Axis

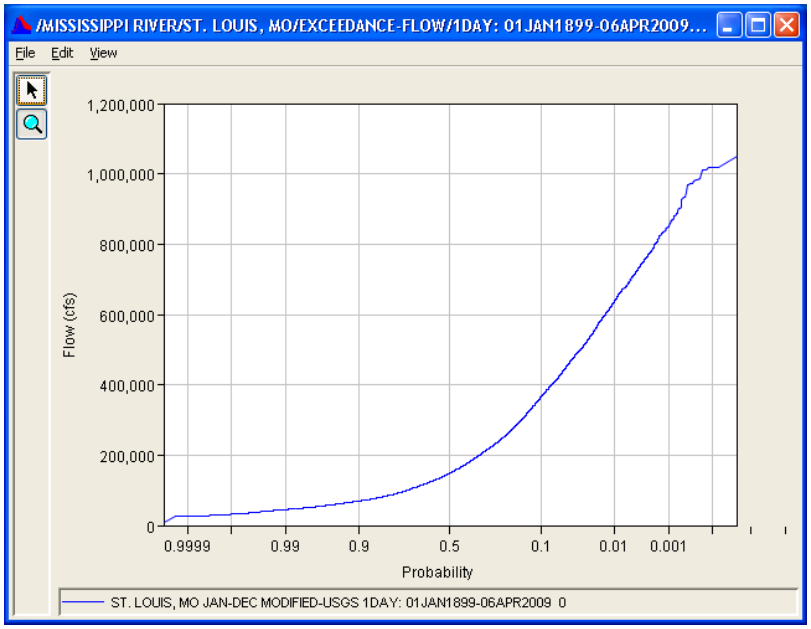
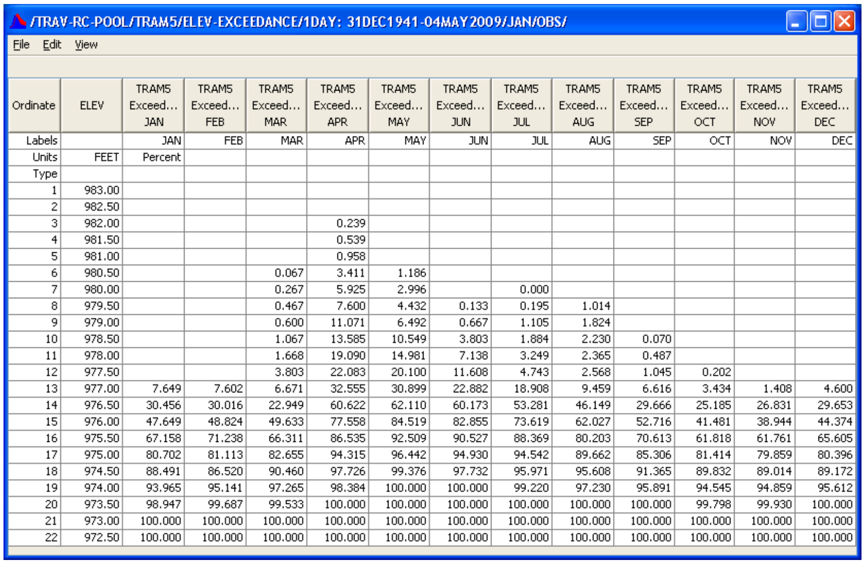

/MISSISSIPPI RIVER/ST. LOUIS, MO/EXCEEDANCE-FLOW/1DAY: 01JAN1899-06APR2009...
File
Edit
View
1,200,000
1,000,000
800,000
600,000
400,000
200,000
0
Flow (cfs)
0.9999
0.99
0.9
0.5
0.1
0.01
0.001
Probability
ST. LOUIS, MO JAN-DEC MODIFIED-USGS 1DAY: 01JAN1899-06APR2009 0

Figure 7.47 Linear Vertical and Log Horizontal Axis

14. You can also select to view the data as tabular or display in Microsoft Excel. The table in Figure 7.48 below was computed by using the bin method for monthly periods and tabulating with Elevation on the X axis.


/TRAV-RC-POOL/TRAM5/ELEV-EXCEEDANCE/1DAY: 31DEC1941-04MAY2009/JAN/OBS/
File
Edit
View

Ordinate	ELEV	TRAM5 Exceed... JAN	TRAM5 Exceed... FEB	TRAM5 Exceed... MAR	TRAM5 Exceed... APR	TRAM5 Exceed... MAY	TRAM5 Exceed... JUN	TRAM5 Exceed... JUL	TRAM5 Exceed... AUG	TRAM5 Exceed... SEP	TRAM5 Exceed... OCT	TRAM5 Exceed... NOV	TRAM5 Exceed... DEC
Labels		JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
Units	FEET	Percent											
Type													
1	983.00												
2	982.50												
3	982.00				0.239								
4	981.50				0.539								
5	981.00				0.958								
6	980.50			0.067	3.411	1.186							
7	980.00			0.267	5.925	2.996		0.000					
8	979.50			0.467	7.600	4.432	0.133	0.195	1.014				
9	979.00			0.600	11.071	6.492	0.667	1.105	1.824				
10	978.50			1.067	13.585	10.549	3.803	1.884	2.230	0.070			
11	978.00			1.668	19.090	14.981	7.138	3.249	2.365	0.487			
12	977.50			3.803	22.083	20.100	11.608	4.743	2.568	1.045	0.202		
13	977.00	7.649	7.602	6.671	32.555	30.899	22.882	18.908	9.459	6.616	3.434	1.408	4.600
14	976.50	30.456	30.016	22.949	60.622	62.110	60.173	53.281	46.149	29.666	25.185	26.831	29.653
15	976.00	47.649	48.824	49.633	77.558	84.519	82.855	73.619	62.027	52.716	41.481	38.944	44.374
16	975.50	67.158	71.238	66.311	86.535	92.509	90.527	88.369	80.203	70.613	61.818	61.761	65.605
17	975.00	80.702	81.113	82.655	94.315	96.442	94.930	94.542	89.662	85.306	81.414	79.859	80.396
18	974.50	88.491	86.520	90.460	97.726	99.376	97.732	95.971	95.608	91.365	89.832	89.014	89.172
19	974.00	93.965	95.141	97.265	98.384	100.000	100.000	99.220	97.230	95.891	94.545	94.859	95.612
20	973.50	98.947	99.687	99.533	100.000	100.000	100.000	100.000	100.000	100.000	99.798	99.930	100.000
21	973.00	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
22	972.50	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000

Figure 7.48 Monthly Exceedences for Elevations

7.8.5 Frequency Plot

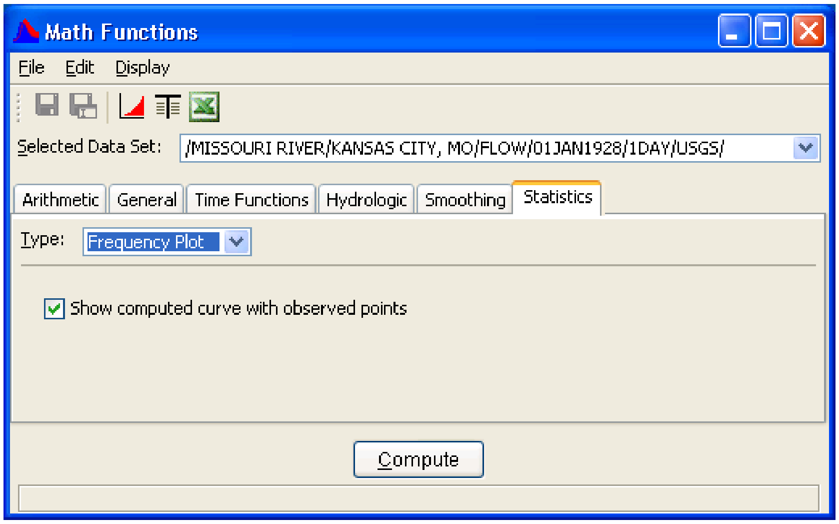
The **Frequency Plot** function computes and displays a frequency plot for a peak annual data from a time series data set. This computation is intended for period of record (years) of primarily stream flow data. The data can be in intervals less than year, as annual peaks are automatically calculated from the data set.

The plot is generated by computing annual peaks from the data set and then ranking the data and plotting. An optional curve may be drawn for your data set using the Pearson type III transformation equation as described in Chapter 12 of EM 1110-2-1415 (USACE, 1993).

For a detailed frequency analysis with additional parameter control, the user is referred to the program HEC-SSP. The HEC-DSSVue computation provides a quick evaluation of data.

To compute a **Frequency Plot** for a time series set:

1. Select the data set from the main HEC-DSSVue screen and then select the **Math Functions** button or Math Functions from the **Tools** menu.
2. Choose the **Statistics** tab of the Math Functions screen and select the **Frequency Plot** type, as shown in Figure 7.49 (page 7-74).



Selected Data Set: /MISSOURI RIVER/KANSAS CITY, MO/FLOW/01JAN1928/1DAY/USGS/

Type: Frequency Plot

☑ Show computed curve with observed points

Figure 7.49 Computing a Frequency Curve

3. Select the time series data set to apply the function from the **Selected Data Set** pull down list.
4. Select the checkbox **Show computed curve with observed points** if you wish to.
5. Press **Compute**.
6. A plot will be displayed showing the frequency plot and computed curve (if selected), which is shown in Figure 7.50.

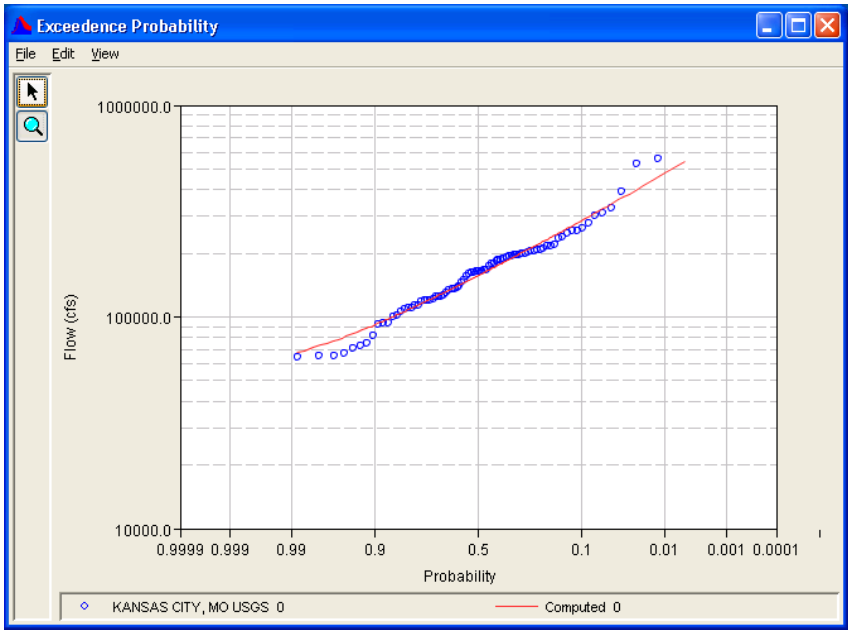


Figure 7.50 Frequency Plot and Computed Curve for Daily Flow Data

CHAPTER 8

Scripting

Scripting provides a way to control the operation of HEC-DSSVue in a non-interactive way. The user can build and save scripts to be executed later - possibly on different data sets.

This chapter provides an introduction to scripting, describing the components of the user interface, scripting language and application program interface (API), and offering examples illustrating how to use the API.

8.1 Executing Scripts

HEC-DSSVue allows the execution of scripts in interactive and batch modes. Scripts are executed interactively by starting the HEC-DSSVue program and selecting the desired script from the Toolbar or the Script Selector from the **Utilities** menu. Scripts are executed in batch mode by starting the HEC-DSSVue program with a script file name as a parameter (e.g. *HEC-DSSVue.exe c:\test\myScript.py*). Scripts can be launched from the HEC-DSSVue program from the **Script Menu**, or from the **Toolbar** or from the **Script Selector**.

Interactive scripts are not passed any parameters upon script execution. In a script executed interactively the variable *sys.argv* is a list of length one, with the only element set to the empty string (e.g., *sys.argv = [" "]*).

Scripts executed in batch mode may take parameters from the command line (e.g., *HEC-DSSVue.exe c:\test\myScript.py a b c*). In a script executed in batch mode the variable *sys.argv* is a list whose length is one greater than the number of parameters passed on the command line, with the first element set to the file name of the executing script and the remaining elements set to the parameters (e.g., *sys.argv = ["c:\\test\\myScript.py", "a", "b", "c"]*).

8.1.1 Script Menu

The **Script Menu** (see Figure 8.1, page 8-2) can display a list of available scripts and execute a script by selecting it from the menu. The **Script**

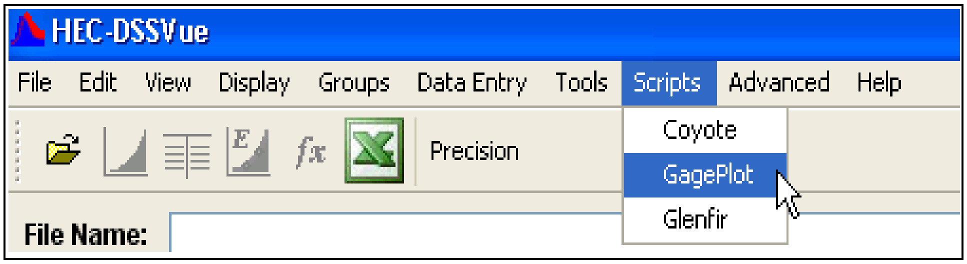
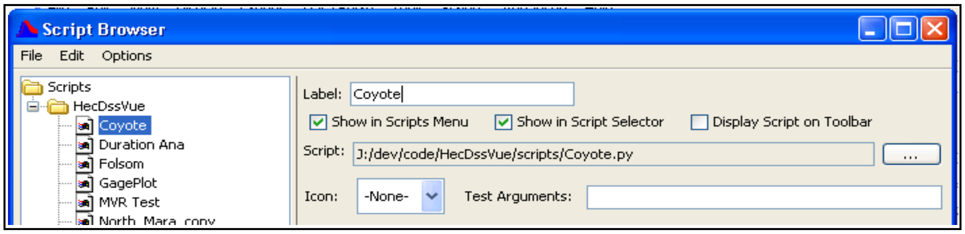

HEC-DSSVue
File Edit View Display Groups Data Entry Tools Scripts Advanced Help
Precision
Coyote
GagePlot
Glenfir
File Name:

Figure 8.1 Running Scripts from the Script Menu

Menu, shown in Figure 8.1, is not made visible on the main HEC-DSSVue menu bar until scripts are set to be displayed on the menu bar from the **Script Editor**, available from the **Tools** menu. To have a script displayed in the **Script Menu**, simply edit the script in the **Script Editor**, select the **Show in Scripts Menu** checkbox at the top of the editor (Figure 8.2).



Label: Coyote

☑ Show in Scripts Menu ☑ Show in Script Selector □ Display Script on Toolbar

Script: J:/dev/code/HecDssVue/scripts/Coyote.py

Icon: -None- Test Arguments:

Figure 8.2 Show in Scripts Menu Checkbox in Script Editor

8.1.2 Main Toolbar

The main toolbar can display buttons for all the available scripts which have the **Display Script on Toolbar** box checked in the Script Editor (Figure 8.2). Buttons are displayed in alphabetical order. After selecting the check box, the menu bar will look like it is shown in Figure 8.3 below.

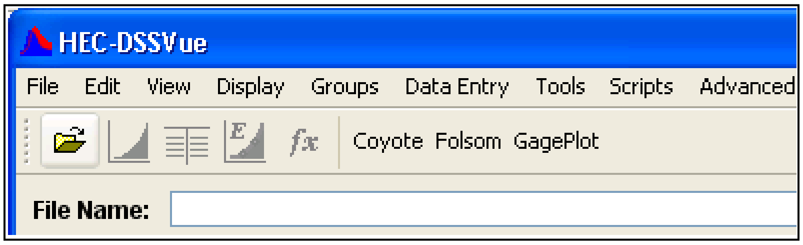

HEC-DSSVue
File Edit View Display Groups Data Entry Tools Scripts Advanced
Coyote Folsom GagePlot
File Name:

Figure 8.3 Script Menu Bar

8.1.3 Script Selector

The **Script Selector** (see Figure 8.4, page 8-3) displays buttons for all the available scripts which have the **Display Script on Toolbar** box checked in the **Script Editor** (Figure 8.2). Buttons are displayed in alphabetical order.

To access the **Script Selector**, select the **Script Selector** command from the **Utilities** menu of HEC-DSSVue. Once the **Script Selector** is open, it will remain open until you close it.

When you press a button, the Jython script engine will execute the associated script.

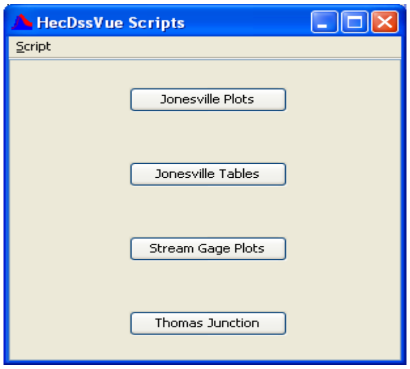
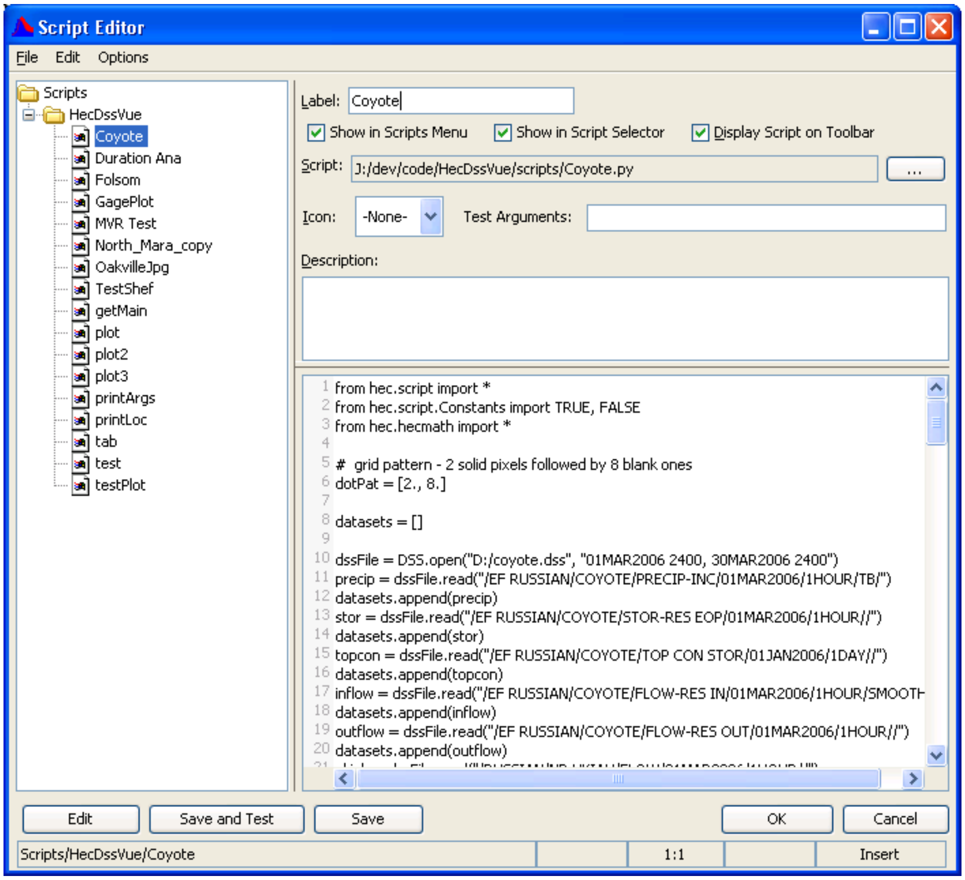


Figure 8.4 Script Selector

8.2 Script Editor

The **Script Editor**, shown in Figure 8.5, allows you to add, delete, and modify scripts.



Label: Coyote

☑ Show in Scripts Menu ☑ Show in Script Selector ☑ Display Script on Toolbar

Script: J:/dev/code/HecDssVue/scripts/Coyote.py

Icon: -None- Test Arguments:

Description:

```
from hec.script import *
from hec.script.Constants import TRUE, FALSE
from hec.hecmath import *

#  grid pattern - 2 solid pixels followed by 8 blank ones
dotPat = [2., 8.]

datasets = []

dssFile = DSS.open("D:/coyote.dss", "01MAR2006 2400, 30MAR2006 2400")
precip = dssFile.read("/EF RUSSIAN/COYOTE/PRECIP-INC/01MAR2006/1HOUR/TB/")
datasets.append(precip)
stor = dssFile.read("/EF RUSSIAN/COYOTE/STOR-RES EOP/01MAR2006/1HOUR//")
datasets.append(stor)
topcon = dssFile.read("/EF RUSSIAN/COYOTE/TOP CON STOR/01JAN2006/1DAY//")
datasets.append(topcon)
inflow = dssFile.read("/EF RUSSIAN/COYOTE/FLOW-RES IN/01MAR2006/1HOUR/SMOOTH
datasets.append(inflow)
outflow = dssFile.read("/EF RUSSIAN/COYOTE/FLOW-RES OUT/01MAR2006/1HOUR//")
datasets.append(outflow)
```

Scripts/HecDssVue/Coyote 1:1 Insert

Figure 8.5 Script Editor

You can access the **Script Editor** from HEC-DSSVue's **Utilities** menu by clicking **Script Editor**. Alternatively, you can get to the **Script Editor** from the shortcut menu of the **Script Selector** (Figure 8.4). In the **Script Selector**, right click on a button to access the shortcut menu, and then select **Edit**. The **Script Browser** will open with that script selected and ready for editing.

Components of the **Script Editor** include the **Menu Bar**, the **Editor Panel**, and the **Tree Hierarchy**. The following sections describe these components.

8.2.1 Menu Bar

The **Menu Bar** (Figure 8.6) contains three primary menu items, **File**, **Edit**, and **Options**.

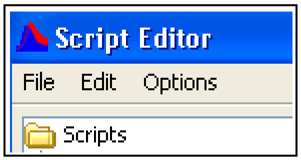


Figure 8.6 Menu Bar

File Menu Commands

New Creates a new script stored at the currently selected position. Available only when a folder node is the selected node in the scripts tree.

Open Edits the currently displayed script. Double clicking on the script also edits the currently displayed script. Available only when a script node is the selected node in the scripts tree.

Import Imports a file into the script browser. If the import is successful the browser is placed in edit mode. Available only when a folder node is the selected node in the scripts tree.

Save Saves the current script. Available only when a script is being edited.

Save As Saves the current script, allowing the user to change the label. Note that the name of the script file does not change. Available only when a script is being edited.

Delete Deletes the currently opened script. Prompts user for confirmation. Available only when a script node is the selected node in the scripts tree.

Test Executes the currently selected script.

Close Closes the **Script Editor**.

Edit Menu Commands

Cut Script Cuts the script at the currently selected tree node to the system clipboard. Available only when a script node is selected in the tree view.

Copy Script Copies the script at the currently selected tree node to the system clipboard. Available only when a script node is selected in the tree view.

Paste Script Pastes the script in the system clipboard to the currently selected tree node. Available only when a folder node is selected in the tree view.

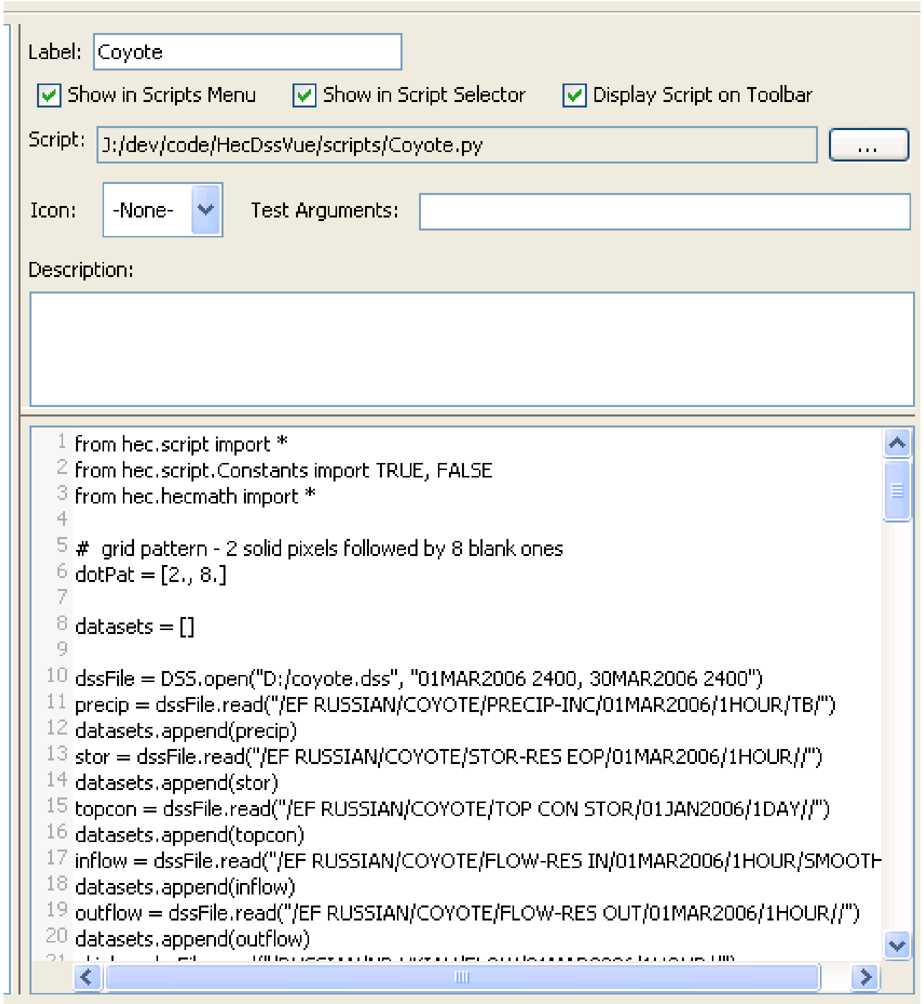
Option Menu Commands

Set Font Opens a dialog box that allows setting of the font used in the script text area. Fixed-space fonts such as "Courier New" and "Lucida Console" are recommended over proportional fonts.

Set Tab Size Opens a dialog box that allows setting the number of spaces that will be displayed in the script text area for each tab character.

8.2.2 Editor Panel

You can select and edit scripts in the **Editor Panel** (Figure 8.7) of the **Script Editor**.



Label: Coyote

☑ Show in Scripts Menu ☑ Show in Script Selector ☑ Display Script on Toolbar

Script: J:/dev/code/HecDssVue/scripts/Coyote.py

Icon: -None- Test Arguments:

Description:

```
from hec.script import *
from hec.script.Constants import TRUE, FALSE
from hec.hecmath import *

#  grid pattern - 2 solid pixels followed by 8 blank ones
dotPat = [2., 8.]

datasets = []

dssFile = DSS.open("D:/coyote.dss", "01MAR2006 2400, 30MAR2006 2400")
precip = dssFile.read("/EF RUSSIAN/COYOTE/PRECIP-INC/01MAR2006/1HOUR/TB/")
datasets.append(precip)
stor = dssFile.read("/EF RUSSIAN/COYOTE/STOR-RES EOP/01MAR2006/1HOUR//")
datasets.append(stor)
topcon = dssFile.read("/EF RUSSIAN/COYOTE/TOP CON STOR/01JAN2006/1DAY//")
datasets.append(topcon)
inflow = dssFile.read("/EF RUSSIAN/COYOTE/FLOW-RES IN/01MAR2006/1HOUR/SMOOTH
datasets.append(inflow)
outflow = dssFile.read("/EF RUSSIAN/COYOTE/FLOW-RES OUT/01MAR2006/1HOUR//")
datasets.append(outflow)
```

Figure 8.7 Editor Panel

The **Label** field allows you to specify the label displayed on a script's button in the **Script Selector**.

Script displays the name of the file in which the script is stored.

Selecting **Show in Scripts Menu** causes the script todisplay in the **Scripts** menu.

Selecting **Show in Script Selector** causes the script to be displayed on the **Script Selector**.

Selecting **Display Script on Toolbar** causes the script to be displayed on the **Toolbar**.

The **Icon** field allows you to choose the Icon to display for the script's button. If you do not select an icon, the script name displays in the script's button.

The **Description** field allows you to add a description of the script. The first line of your description serves as a tooltip for the corresponding button on the **Script Selector** and **Toolbar**.

The **Script Text** field contains the script text itself and serves as an editing window for creating new scripts.

The script text field has a context menu that can be accessed by right-clicking in the script text field.

Script Text Field Context Menu Commands

Cut	Copies the currently-selected script text to the system clipboard and removes it from the script.
Copy	Copies the currently-selected script text to the system clipboard and leaves it in the current script.
Paste	Copies text in the system clipboard into the script at the current cursor location.
Select All	Selects all the text in the script.
Find	Opens a dialog that allows the user to search for specific text in the script. If text is currently selected in the script, the dialog is initialized with this text.
Find Next	Locates the next text in the script that matches the conditions of the most recently executed find command.
Find Previous	Locates the previous text in the script that matches the conditions of the most recently executed find command.
Goto Line	Opens a dialog that allows the user to cause the cursor to jump to the beginning of a specified line in the script text.

8.2.3 Tree Hierarchy

The **Tree Hierarchy**, as shown in Figure 8.8, uses a Windows Explorer-style tree structure to allow you to navigate folders in your directory structure and access scripts. Scripts are stored in a "*scripts*" directory under the directory where HEC-DSSVue was installed.

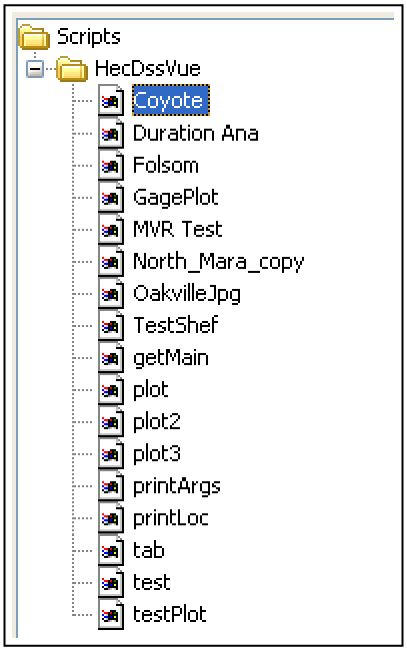


Figure 8.8 Tree Hierarchy

The **Tree Hierarchy** also has a context menu that displays **Cut Script**, **Copy Script**, and **Edit Script** commands for script nodes and **New Script**, **Import Script**, and **Paste Script** for folder nodes. These commands cause the same actions as the **File** and **Edit** menu commands discussed above.

To access the context menu, point to a node in the "tree" and right-click with your mouse.

8.3 Scripting Basics

Scripting in HEC-DSSVue is accomplished using Jython, an implementation of the Python programming language designed specifically for integration with the Java programming language. More information about Jython can be found at the official Jython website – www.jython.org.

Python (of which Jython is an implementation) is an interpreted language with simple syntax and high-level data types. This section is not a comprehensive Python tutorial, but rather a simple primer to allow the creation of simple scripts. This primer does not cover defining classes in Python.

The official Python website - www.python.org - has links to online Python tutorials as well as programming books.

8.3.1 Outputting Text

Text information can be displayed in the console window using the print statement which has the syntax:

print [*item*[, *item*…]]

The *item*s are comma-separated and do not need to be of the same type. The print statement will insert a space between every specified item in the output.

Example 1: Outputting Text

```
print “Testing myFunction, i =”, i, “, x =”, x
```

8.3.2 Data Types and Variables

Python has Boolean, integer, long integer, floating-point, imaginary number, and sequence and dictionary data types. Sequences are divided into mutable (or changeable) sequences called lists, immutable sequences called **tuples**. Tuples may be enclosed in parentheses or may be "naked". Strings are special tuples of characters that have their own syntax. Dictionaries are like sequences but are indexed by non-numeric values. In addition, Python also has a special type called **None**, which is used to indicate the absence of any value. The only valid values for the Boolean type are True and False. In certain circumstances these values also evaluate to the integers 1 and 0, respectively.

Python variable names consist of an upper- or lower-case letter or the “_” (underscore) character followed by an unlimited number of upper- or lower-case characters, digits or underscore characters.

Like many languages, Python uses the single equals sign, "=", to assign values to variables. However, Python allows assignment to multiple variables from a single sequence, as long as the number of variables on the left of the equal sign is the same as the length of the sequence on the right side. If the sequence on the right side is a dictionary, only the dictionary keys are assigned to the variables on the left.

There are situations regarding the HEC-DSSVue API where it is necessary or desirable to set a time-series value to "missing" or to test whether a time-series value is missing. The *hec.script* module supplies a constant to use in these situations.

The currently-defined constants in the *hec.script* module are:

Constant	Type	Represents
Constants.TRUE	integer	True
Constants.FALSE	integer	False
Constants.UNDEFINED	floating-point	missing data value

Example 2: Variable Types

```
# set some integer values
i = 0
j = 1
k = -10998
m = True

# set a long integer
n = 79228162514264337593543950336L

# set some floating-point values
x = 9.375
y = 6.023e23
z = -7.2e-3
t = Constants.UNDEFINED

# set some strings
string_1 = “abc”
string_2 = ‘xyz’
string_3 = “he said \“I won’t!\””
string_4 = ‘he said “I will not!”‘
string_5 = “””this is a
              multi-line string”””

# set a tuple – tuples are contained within ()
tuple_1 = (1, 2, “abc”, x, None)

# set a list – lists are contained within []
list_1 = [1, 2, “abc”, x, tuple_1]

# set a dictionary, using key : value syntax
# dictionaries are contained within {}
dict_1 = {“color” : “red”, “size” : 10, “list” : [1, 5, 8]}

# multiple assignment
a, b, c = string_1
a, b, c = “abc”
a, b, c, d, e = tuple_1
a, b, c, d, e = (1, 2, “abc”, x, None)
a, b, c, d, e = 1, 2, “abc”, x, None     # “Naked” tuple assignment
a, b, c, d, e = list_1
a, b, c, d, e = [1, 2, “abc”, x, tuple_1]
a, b, c = dict_1
a, b, c = {“color” : “red”, “size” : 10, “list” : [1, 5, 8]}
```

Indexing into sequence types is done using [*i*] where *i* starts at 0 for the first element. Subsets of sequence types (called slices) are accessed using [*i*:*j*:*k*] where *i* is the first element in the subset, *j* is the element **after** the last element in the subset, and *k* is the step size. If negative numbers are used to specify and index or slice, the index is applied to the **end** of the sequence, where [-1] specifies the last element, [-2] the next-to last and so on. If *k* is omitted in slice syntax it defaults to 1. If *i* is omitted in slice syntax it defaults to 0. If *j* is omitted in slice syntax it defaults to the length of the sequence, so list_1 [0:len(list_1)] is the same as list_1[:]. Indexing into dictionaries is done using [*x*] where *x* is the key.

The number of elements in a sequence type or dictionary is returned by the len() function.

Example 3: Sequence Indexing and Slicing

```
string_4[3]             # 4th element
string_4[3:5]           # 4th & 5th elements
list_1[0::2]            # even elements
list_1[1::2]            # odd elements
list_1[-1]              # last element
list_1[2:-1]            # 3rd through next-to-last element
list_1[2:len(list_1)]   # 3rd through last element (also list_1[2:])
dict_1[“size”]          # value associated with “size” key
i = len(list_1)         # length of list_1
```

8.3.3 Operators

Each of the following operators can be used in the form a = b operator c. Each can also be used as an assignment operator in the form a operator= b (e.g., *a += 1, x **= 2*).

+	arithmetic addition
-	negation or arithmetic subtraction
*	arithmetic multiplication
/	arithmetic division
//	integer arithmetic division
**	arithmetic power
%	arithmetic modulo
&	bit-wise and
|	bit-wise or
~	bit-wise not
^	bit-wise xor (exclusive or)
<<	bit-wise left shift
>>	bit-wise right shift

Each of the following operators returns True or False and can be used in conditional expressions as discussed in Section 8.3.6.

>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
!=	not equal to
==	equal to

8.3.4 Comments

Python uses the "#" (hash) character to indicate comments. Everything from the "#" character to the end of the line is ignored by the interpreter. Comments may not be placed after a program line continuation ("\") character on the same input line.

8.3.5 Program Lines

Unless otherwise indicated, every input line corresponds to one logical program statement. Two or more statements can be combined on line input line by inserting the ";" (semicolon) character between adjacent statements. A single statement may be continued across multiple input lines by ending each line with the "\" (back slash) character. Comments may not be placed after a program line continuation ("\") character on the same input line.

Example 4: Input vs. Program Lines

```
# multiple statements per line
r = 1; pi = 3.1415927; a = pi * r ** 2

# multiple lines per statement
a = \
  pi * \
  r ** 2
```

Input lines are grouped according to their function. Input lines forming the body of a conditional, loop, exception handler, or function or class definition must be grouped together. Input lines not in any of the construct comprise their own group. In Python, grouping of input lines is indicated by indentation. All lines of a group must be indented the same number of spaces. A horizontal tab character counts as eight spaces on most systems. In some Python documentation, a group of input lines is called a *suite*.

Example 5: Input Line Grouping

```
# this is the main script group
dist = x2 – x1
if dist > 100.:
    # this is the “if” conditional group
    y = dist / 2.
    z = y ** 2.
else :
    # this is the “else” conditional group
    y = dist.
    z = y ** 2. / 1.5
# back to main script group
q = y * z
```

8.3.6 Conditional Expressions

Conditional expressions have the form:

```
if [not] condition :
  if-group
[elif [not] condition :
  elif-group]
[else :
  else-group]
```

The ":" (colon) character must be placed after each condition. The condition in each test is an expression built from one or more simple conditions using the form:

```
simple-condition ( and | or ) [not] simple-condition
```

Parentheses can be used to group conditions. The simple-condition in each expression is either an expression using one of the conditional operators mentioned before or is of the form:

```
item [not] in sequence
```

If the statement group to be processed upon a condition is a single statement, that statement may follow the condition on the same line (after the colon character).

Example 6: Conditional Expressions

```
if (x < y or y >= z) and string_1.index(“debug”) != -1 :
   # do something
   …
elif z not in value_list or (x < z * 2.5) :
   # do something different
   …
else :
   # do something else
```

Example 7: Simple Conditional Expressions

```
if x1 < x2 : xMax = x2
else       : xMax = x1
```

8.3.7 Looping

Python supports conditional looping and iterative looping. For each type, the body of the loop (the loop-group) can contain **break** and/or **continue** statements.

The **break** statement immediately halts execution of the loop-group and transfers control to the statement immediately following the loop.

The **continue** statement skips the remainder of the current iteration of the loop-group and continues with the next iteration of the loop-group

Conditional Looping

Python supports conditional looping with the **while** statement, which has the form:

```
while condition :
    loop-group
```

Conditional looping executes the body of the loop (the loop-group) as long as the condition evaluates to true.

Example 8: Conditional Looping

```
# print the first 10 characters
string_1 = “this is a test string”
i = 0
while i < 10 :
    print string_1[i]
    i += 1
```

Iterative Looping

Python supports iterative looping with the **for** statement, which has the form:

```
for item in sequence :
    loop-group
[else :
    else-group]
```

Iterative looping executes the body of the loop (the loop-group) once for each element in *sequence*, first setting *item* to be that element. If the iteration proceeds to completion without being interrupted by a break statement the else-group will be executed, if specified.

Example 9: Iterative Looping

```
# print the first 10 characters
string_1 = “this is a test string”
for i in range(10) :
    print string_1[i]

# print all the characters
string_1 = “this is a test string”
for i in range(len(string_1)) :
    print string_1[i]

# print all the characters (more Python-y)
string_1 = “this is a test string”
for c in string_1 :
    print c
```

The **range([*start*,] *stop*[, *increment*])** helper function generates a sequence of numbers from *start* (default = 0) to *stop*, incrementing by *increment* (default = 1). Thus range(4) generates the sequence (0, 1, 2, 3).

8.3.8 Defining and Using Functions

In Python, functions are defined with the syntax:

```
def functionName([arguments]) :
    function-body
```

Function names follow the same naming convention as variable names specified in Section 8.3.2. The arguments are specified as a comma-delimited list of variable names that will be used within the function-body. These variables will be positionally assigned the values used in the function call. More complex methods of specifying function arguments are specified in Python tutorials and references listed at the official Python website (www.python.org).

A function must be defined in a Python program before it can be called. Therefore, function definitions must occur earlier in the program than calls to those functions.

A function may optionally return a value of any type or a naked tuple of values.

Example 10: Defining and Using Functions

```
def printString(stringToPrint) :
  “Prints a tag plus the supplied string”
  tag = “function printString : “
  print tag + stringToPrint

def addString(string_1, string_2) :
  “Concatenates 2 strings and returns the result”
  concatenatedString = string_1 + string_2
  return concatenatedString

testString = “this is a test”
printString(testString)
wholeString = addString(“part1:”, “part2”)
printString(wholeString)
printString(addString(“this is “, “another test”))
```

8.3.9 Modules, Functions and Methods

A function is a procedure which takes zero or more parameters, performs some action, optionally modifies one or more of the parameters and optionally returns a value or sequence of values.

A class is the definition of a type of object. All objects of that type (class) have the same definition and thus have the same attributes and behavior. Classes may define functions that apply to individual objects of that type. These functions are called methods.

An object is an instance of a class, which behaves in the way defined by the class, and has any methods defined by the class.

Python provides many functions and classes by default. In our examples, we have used functions **len()** and **range()** which Python provides by default. We have also used the classes list and string, which Python also provides by default. We didn't use any methods of the class list, but we used the string method **index()** in Example 7 (*string_1.index("debug") != -1*). It is important to note that the object method **index()** doesn't apply to the string class in general, but to the specific string object string_1.

There are other functions and classes which Python does not provide by default. These functions and classes are grouped into modules according to their common purpose. Examples of modules are "os" for operating system functions and "socket" for socket-based network functions. Before any of the functions or classes in a module can be accessed, the module must be imported with the import statement, which has the syntax:

```
from module import *
```

Other methods of using the import statement are specified in Python tutorials and references listed at the official Python website (www.python.org). In the Jython implementation, Java packages can be imported as if they were Python modules, and the Java package *java.lang* is imported by default.

Example 11: Using a Function from an Imported Module

```
# use the getcwd() function in the os module to get
# the current working directory

from os import *
cwd = getcwd()
```

A module *does not* have to be imported in order to work with objects of a class defined in that module *if* that object was returned by a function or method already accessible. For example, the Python module "string" does not have to be imported to call methods of string objects, but *does* have to be imported to access string functions.

8.3.10 Handling Exceptions

Certain errors within a Python program can cause Python to raise an exception. An exception that is not handled by the program will cause the program to display error information in the console window and halt the program.

Python provides structured exception handling with the following constructs:

```
try :
  try-group
except :
  except-group
[else :
  else-group]

try :
  try-group
finally :
  finally-group
```

In the try-except-else construct, if an exception is raised during execution of the try-group, the control immediately transfers to the first line of the except-group. If no exception is raised during execution of the try-group, the control transfers to the first line of the else-group, if present. If there is no exception raised and no else-group is specified, the control transfers to the first line after the except-group.

The two constructs cannot be combined into a try-except-finally construct, but the same effect can be obtained by making a try-except-else construct the try-group of a try-finally construct.

Example 12: Exception Handling

```
try :
  try :
    string_1.find(substring) # may raise an exception
  except :
    print substring + “ is not in “ + string_1
    # do some stuff that might raise another exception
    …
  else :
    print substring + “ is in “ + string_1
    # do some stuff that might raise another exception
    …
finally :
  print “No matter what, we get here!”
```

More exception handling information, including filtering on specific types of exceptions, exception handler chains, and raising exceptions, is provided in Python tutorials and references listed at the official Python website (www.python.org). In the Jython implementation, instances or subclasses of *java.lang.Throwable* can also be handled as exceptions. Detailed information can be found at the Java API website (http://java.sun.com/javase/6/docs/api/).

8.4 Displaying Messages

It is often useful to display messages to inform the user that something has occurred, to have the user answer a Yes/No question, or offer debugging information to help determine why a script isn't working as expected. Text information can be displayed in the console window as described in Section 8.3.1.

8.4.1 Displaying Message Dialogs

The **MessageBox** class in the *hec.script* module, and the **MessageDialog** class in the **MessageDialog** module have several functions used to display messages in message box dialogs. The dialogs can be one of four different types: Error, Warning, Informational or Plain. The difference between the **MessageBox** functions and their **MessageDialog** counterparts is the **MessageBox** versions are modal, while the **MessageDialog** versions are modeless (see Section 8.5.4).

MessageBox and **MessageDialog** functions with multiple buttons return a string containing the text of the button that was selected to dismiss the message box.

Note: Do not use the **MessageBox** or **MessageDialog** functions in a script that is to run unattended since these functions cause scripts to pause for user interaction. **MessageDialog** functions should only be used in batch mode scripts (see Section 8.5.3).

MessageBox and **MesssageDialog** functions are described in Table 8.1.

Example 13: Display Error Dialog with MessageBox class

```
from hec.script import *

MessageBox.showError("An Error Occurred”, "Error")
```

Example 14: Display OK/Cancel Dialog with MessageDialog class

```
from hec.script import *
import MessageDialog
   ok=MessageDialog.showOkCancel(“Continue with Operation”, “Confirm”)
```

Table 8.1 MessageBox and MessageDialog Functions

Function	Returns	Comments
showError(string message, string title)	None	Display an error dialog to you with the message and title
showWarning(string message, string title)	None	Display a warning dialog to you with the message and title
showInformation(string message, string title)	None	Display a Informational dialog to you with the message and title
showPlain(string message, string title)	None	Display a plain dialog to you with the message and title
showYesNo(string message, string title)	string	Display a Yes/No dialog to you with the message and title
showYesNoCancel(string message, string title)	string	Display a Yes/No/Cancel dialog to you with the message and title
showOkCancel(string message, string title)	string	Display a Ok/Cancel dialog to you with the message and title

8.5 Headless Operation

Often it is desirable to run scripts in the "background" so that you do not see the interactions or tables or graphics, but instead have the appropriate functions executed or *.png* files created. Most math scripts do not write to the screen, so this is not an issue. If you call higher level functions, there is the potential that some dialogs may appear, especially if there are errors. The **ListSelection** class tries to figure out how you are running a script and will only print messages if you are running it in the background, but show message dialogs if your are running in the foreground. This means that you can get different message dialog behavior if you run a script from HEC-DSSVue that accesses **ListSelection** than if you run it in a batch mode.

8.5.1 Headless on Windows

On the PC, you can set the location of plots and tables off-screen so that they are not visible to the user, create the *.png* or *.jpg* file, and then close the plot or table. For example:

```
plot = Plot.newPlot()
plot.setSize(600, 500)
plot.setLocation(-10000, -10000)
```

```
…
plot.showPlot()
…
plot.saveToPng("C:/temp/myPlot.png")
plot.close()
```

8.5.2 Headless on UNIX

On UNIX, you must have an X-Window server on which to generate a plot or table. This is done using a Virtual X-Window server, such as Xvfb. The programs treat this as a normal X-Window display. Like the PC, you run the scripts as normal, although you don’t need to set at location:

```
plot = Plot.newPlot()
plot.setSize(600, 500)
…
plot.showPlot()
…
plot.saveToPng("/tmp/myPlot.png")
plot.close()
```

For more information, refer to:
http://www.x.org/archive/X11R6.8.2/doc/Xvfb.1.html

On UNIX it is generally advantageous to use the same script to run in an interactive mode as well as a batch mode. This can be accomplished by checking to see if the DISPLAY environment variable is set. If it isn't, then check that Xvfb is running and set the DISPLAY variable to ":1.0". For example:

```
#!/bin/ksh
# *******************************************************************
if [ "$DISPLAY" = "" ]
then
   XvfbRunning=`ps -ef | grep Xvfb | grep -v grep | wc -l | sed -e "s/ //g"`
   if [ $XvfbRunning -gt 0 ]
   then
      DISPLAY=":1.0"
      export DISPLAY
      printf "\nWARNING : Using virtual display – “
      printf "no graphics will be visible.\n\n"
   else
      printf "\nWARNING : Virtual display server is not running."
      printf "\n          Set DISPLAY variable or start Xvfb.\n\n"
      return -1
   fi
fi
# *******************************************************************
# Execute the program here….
```

8.5.3 Interactive and Batch Mode Scripts

A script is said to be *interactive* if it is executed from the HEC-DSSVue menu bar, tool bar, **Script Selector**, or **Script Editor**. With any of these methods, the user interacts with the HEC-DSSVue GUI to lauch a script. Conversely, a script is said to be in *batch mode* if it is executed from a command prompt or shortcut as a parameter to the HEC-DSSVue program.

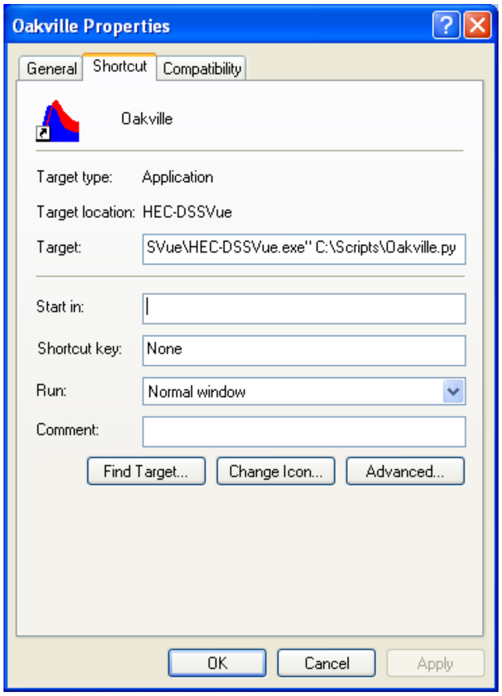
To run a script in batch mode, you specify the file name of the script as a parameter to HEC-DSSVue, following the program name; for example:

```
HEC-DSSVue.exe C:\HecDSSVueDev\HecDssVue\scripts\Oakville.py
```

In Windows, this can be passed through a *.bat* file and then you can run that .bat file; e.g., *RunOakville.bat*:

```
HEC-DSSVue.exe C:\HecDSSVueDev\HecDssVue\scripts\Oakville.py
```

On Windows, a shortcut can be utilized instead of a batch file. To create a shortcut for a script, copy the HEC-DSSVue shortcut and pass the script file name in following the *Target* in the Shortcuts Properties dialog box (Figure 8.9). Right click on the original HEC-DSSVue shortcut on the desktop and click **Copy**. Right click on an empty area on the desktop and click **Paste**. Right-click on the copy and Rename to an appropriate name (.e.g., your script name), right click again and click **Properties**. Enter the script name following the program name in the *Target* area, as shown in Figure 8.9.



Oakville

Target type: Application

Target location: HEC-DSSVue

Target: SVue\HEC-DSSVue.exe" C:\Scripts\Oakville.py

Start in:

Shortcut key: None

Run: Normal window

Comment:

Figure 8.9 Shortcut Properties Dialog Box

In UNIX, a file is made executable by changing its mode to be executable, e.g., *RunOakville*:

```
HEC-DSSVue /usr1/HecDssVue/scripts/Oakville.py
chmod 555 RunOakville
```

If a batch mode script requires parameters, they must be appended to the command line (or target field in the shortcut) after the name of the script:

```
HEC-DSSVue.exe C:\HecDSSVueDev\HecDssVue\scripts\xyz.py a b c
```

A script may determine whether it is executing in interactive or batch mode by calling the *isInteractive()* **ListSelection** method in Table 8.3 (see page 8-23).

One characteristic of batch mode scripts is that they any GUI components (plot, tables, etc…) displayed during their execution are destroyed when the script completes. While this behavior is acceptable for generating files containing tables and plot images, it is often desirable for a batch mode script to display GUI components that persist until the user dismisses them. A batch mode script may always utilize one of the **MessageDialog** functions listed in Table 8.1. However, if a batch mode script uses one of the window types listed in Table 8.2, it can call the *stayOpen()* method of the window to gain persistence without the need for an extra dialog.

Table 8.2 Window Types with *stayOpen()* Method

Window
ListSelection
Plot (G2dDialog)
Table (HecDataTable)
CompareFiles
ExcelFrame
Math
SelectRecords
UndeleteRecords

The **stayOpen()** method causes the window to wait and stay open until the window is closed. Like the **MessageDialog** functions, the *stayOpen()* **cannot be used in an interactive script**. It will cause the program to freeze. Plots will stay open automatically when in an interactive mode. An example use for *Oakville.py* is:

```
#  Display the plot and wait for them to close
plot.stayOpen()    # Do not use in an interactive script
```

8.5.4 Modality in Scripts

Modality refers to whether an application window allows interaction with other windows of the same application. A **modal** window prevents interaction with other application windows as long as it is displayed. Once the modal window is dismissed, interaction with other application windows resumes. Conversely, **modeless** windows allow interaction with other application windows while they are displayed. Most HEC-DSSVue windows are modeless, although error dialogs and dialogs created with the **MessageBox** functions described inTable 8.1 are modal. **MessageDialog** functions described in Table 8.1 that are modeless, can not be executed in interactive scripts unless they are called in a separate thread from the graphics display. Writing multi-threaded scripts is beyond the scope of this manual.

8.6 Accessing the Main Program Window

Situations arise that necessitate accessing the main HEC-DSSVue program window. Examples of such situations are:

- Having a script determine whether it is running in an interactive mode or without a graphical display.
- Having a script gather information about interactively-selected pathnames and time windows for automated processing.
- Having a script launch the interactive graphical editor.

The **ListSelection** class in the *hec.dssgui* module facilitates these activities. An import statement of the form "from *hec.dssgui* import ListSelection" is necessary to use the **ListSelection** class. (For CWMS-VUE, the import statement is "from *hec.cwmsVue* import CwmsListSelection").

If the script is executing in batch mode then a main program window will not exist. If the script needs to have the user utilize the graphical editor, the script will need to create a main program window for the duration of the graphical edit session.

8.6.1 ListSelection Class

The static functions of the **ListSelection** class are listed in Table 8.3.

Table 8.3 ListSelection Static Functions

Function	Returns	Description
createMainWindow()	ListSelection	Returns a new ListSelection object. This should not be called unless the script is not running in interactive mode.
getMainWindow()	ListSelection	Returns the main program window (ListSelection object) of the script. Returns None if the script is not running in interactive mode.
isInteractive()	Boolean	Returns if the program is being run interactively or in a batch mode.

At the end of a batch script that calls *createMainWindow()*, you should close DSS files and exit the **ListSelection** call using the *finish()* method. This function properly shuts down the program without altering any properties. For example:

```
mainWindow = ListSelection.createMainWindow()
…
mainWindow.finish()
```

Don’t call *finish()* when running interactively (unless you want the program to terminate). **ListSelection** objects can be used to gather information about pathnames and time windows that the user selected interactively before launching the script. **ListSelection** objects can also be used to launch the interactive graphical editor. The **ListSelection** methods are listed in Table 8.4.

Table 8.4 ListSelection Methods

Method	Returns	Description
addToolBarButton(JButton b)	None	Adds a button to the main toolbar.
addToSelection(String path)	None	Adds a pathname to the selection list.
clear()	None	Clears selected pathnames.
close()	None	Close the current DSS file.
copyPathnamesToClipboard()	None	Copies selected pathnames to the clipboard
copyRecords(Boolean)	None	Copies selected records to the second DSS file opened if 0, or displays an open dialog
delete()	None	Deletes the selected records
duplicateRecords()	None	Duplicates the selected records
exitProgram()	None	Exits the program, saving settings
exportShef(String filename)	None	Exports the selected records to SHEF in the file given
exportShef(String filename, Vector timeSeriesContainers)	None	Exports the provided data to SHEF in the file given
exportToFile()	None	Exports images, etc. by opening a Open dialog
fileCheck()	None	Runs a file check on the opened DSS file
finish()	None	Exits without saving settings
getCustomMenu()	JMenu	Returns the custom menu
getDataManager()	CombinedData Manager	Returns the combined data manager for the current file
getDisplayMenu()	Jmenu	Returns the Display menu
getDSSFilename()	string	Returns the current DSS file name
getEndTime()	HecTime	Returns the current end time of the ListSelection as an HecTime object.[1]
getEditMenu()	Jmenu	Returns the Edit menu
getExport Menu()	Jmenu	Returns the Export menu
getFileMenu()	Jmenu	Returns the File menu
getImportMenu()	Jmenu	Returns the Import menu
getNumberSelectedPathnames()	integer	Returns the number of selected pathnames
getScriptMenu()	Jmenu	Returns the script menu
getSecondDataManager()	CombinedData Manager	Returns the combined data manager for the second file

Method	Returns	Description
getSelectedDataContainers()	List	Reads data for the selected pathnames and returns it in a list of data containers
getSelectionList()	list of DataReference objects	Returns a list of DataReference objects that represent the interactive pathname and time window selections.
getSelectedPaths()	Vector	Returns a Vector of Strings of pathnames in the open DSS file.
getStartTime()	HecTime	Returns the current start time of the ListSelection as an HecTime object.[1]
getTimeWindow(HecTime startTime, HecTime endTime)	Boolean	Returns whether the start time and end time provided are defined
getToolBar()	JtoolBar	Returns the tool bar
getViewMenu()	Jmenu	Returns the View menu
graphicalEdit()	None	Launches the graphical editor for the interactively-selected pathnames and time windows.
graphicalEdit(TimeSeriesContainer tsc)	None	Launches the graphical editor for the specified TimeSeriesContainer.
graphicalEdit(list tscList)	None	Launches the graphical editor for all TimeSeriesContainers in the list.
importGenericFiles()	None	Launches a file open dialog to import files
importImages()	None	Launches a file open dialog to import image files
importShef()	None	Launches a file open dialog to import SHEF files
importShef(string filename)	None	Imports the SHEF file
isDssFileAccessible()	Boolean	Returns if the DSS file set earlier is accessible currently
isRemote()	Boolean	Returns whether the DSS file opened is remote and accessed using DSSManager
isVisible()	Boolean	Returns whether the dialog is currently visible on the screen
math()	None	Reads the selected data and then opens the math dialog
mergeFiles(string toDssFilename)	None	Copies all the data from the opened DSS file into the one specified
openDSSFile (string DSSFileName)	Boolean	Opens the DSS file name of the ListSelection and returns whether it was opened or not
pairedDataEntry()	None	Opens the paried data entry dialog
plot()	None	Reads the selected data and then plots it
read(string pathname)	DataContainer	Reads the recorded specified by the pathname
read(string pathname, HecTime start, HecTime end)	DataContainer	Reads the recorded specified by the pathname and time window

Method	Returns	Description
read(string pathname, String timeWindow)	DataContainer	Reads the recorded specified by the pathname and time window
refreshCatalog()	None	Refreshes the catalog for the opened file
registerExportPlugin(JmenuItem menuItem)	None	Adds the menu item to the export menu
registerImportPlugin(ImportPlugin importPlugin, JmenuItem menuItem, string dropAndDragExtension)	None	Adds the menu item to the import menu and associates the plugin and drag and drop extension to that item

registerPlugin(integer menuNumber, JmenuItem menuItem)	None	Adds the menu item to main menus. Menu numbers are the following:
		FILE = 0;
		EDIT = 1;
		VIEW = 2;
		DISPLAY = 3;
		GROUPS = 4;
		DATA_ENTRY = 5;
		TOOLS = 6;
		CUSTOM = 7;
		SCRIPTS = 8; ADVANCED = 9;
		HELP = 10;

renameRecords()	None	Brings up the rename dialog for the selected pathnames
runFile()	None	Reads the selected data and runs the programs associated with them (e.g., *.pdf*, *.mp3*)
runFile(Vector pathnames)	None	Reads the selected pathnames and runs the programs associated with them (e.g., *.pdf*, *.mp3*)
save(DataContainer dataContainer)	Boolean	Saves the data container in the opened DSS file. Returns if success or not
saveAs(DataContainer dataContainer)	Boolean	Opens the Save As dialog and saves the data container in the opened DSS file. Returns if success or not
selectAll()	Boolean	Adds all pathnames to the selection list
setCustomMenu(string menuText)	None	Sets the text displayed on the menu bar
setDssLogFile(string logFile)	None	Writes DSS messages to the file provided
setScriptMenu(string menuText)	None	Sets the text displayed on the menu bar
setSelectedPathnames(Vector selectedPathnames)	None	Sets the selected list of pathnames
setTimeWindow(string timeWindow)	None	Sets the current time window of the ListSelection using a string value to define the start time and end time
setTimeWindow(string start, string end)	None	Sets the current time window of the ListSelection using a string value to define the start time and end time
setTimeWindow(HecTime start, HecTime end)	None	Sets the current time window of the ListSelection using an Hec Time object to define the start time and end time

Method	Returns	Description
setVisible(Boolean visible)	None	Sets whether the dialog is visible on the screen.
squeeze()	None	Squeezes the open DSS file
tabulate()	Jframe	Tabulates the selected data
tabularEdit()	Jframe	Displays the tabular editor for the selected data
textEntry()	None	Displays the text entry dialog
textFileEntry()	None	Displays an open dialog to read a text file into text format
timeSeriesDataEntry()	None	Displays the time series data entry dialog
undeleteAll()	None	Undeletes all deleted records in the DSS file
undeleteSelected()	None	Undeletes the selected records
updateCatalog()	None	Creates a new catalog, if needed (refreshCatalog forces a new catalog)
updateMessageField(string message)	None	Puts the message in the message field

[1] The current start and end times of a **ListSelectio**n object are not necessarily the same as the start and end times of any **DataReference** objects in the **ListSelection**. Rather they represent the start and end times that will be applied to subsequent selections.

8.6.2 DataReference Class

The *getSelectionList()* **ListSelection** method returns a Vector of DataReference objects that represent the interactive selections. Each individual selection may have a different DSS filename, pathname, and time window than other selections. The methods of the **DataReference** class are listed in Table 8.5.

Table 8.5 DataReference Methods

Method	Returns	Description
getFilename()	string	Returns the name of the DSS file from which this selection was made.
getNominalPathname()	string	Returns the nominal pathname for this selection. If a condensed set, this will not be a real pathname, but show the date span in the D part
getPathname()	string	Returns the pathname for this selection
getPathnameList()	Vector	Returns a list of the pathnames for this selection. This will be only more than one if a condensed reference
getTimeWindow(HecTime startTime, HecTime endTime)	Boolean	Sets the startTime and endTime parameters with the start time and end time of the selection, respectively. Returns True if a time window is defined for the selection. Returns False otherwise.
hasTimeWindow()	Boolean	Returns True if a time window is defined for the selection. Returns False otherwise.

8.7 Reading and Writing to HEC-DSS Files

Retrieving and storing data from a HEC-DSS file is done through the class **HecDss** in the *hec.heclib.dss* package. The older technique of using **DSSFile** objects returned from *DSS.open()* calls is still supported, but lacks the ability to operate directly with **DataContainer** objects. HecDss inherits from DSSFile in HecMath and retains all the functionality of that class for backwards compatibility. The first call that should be made is the static member *HecDss.open(string filename)*, which will return a **HecDss** object to retrieve and store data, as well as perform math and other functions.

You need to import the classes that you use as well as the hec.script package. To do this, include the follow lines at the beginning of your script:

```
from hec.script import *
from hec.heclib.dss import *
```

8.7.1 HecDss Class

```
from hec.heclib.dss import *

HecDss.open(string filename)
HecDss.open(string filename, string startTime, string
endTime)
HecDss.open(string filename, string timeWindow)
HecDss.open(string filename, Boolean fileMustExist)
```

The **HecDss** class is used to gain access to an HEC-DSS file, as illustrated in Example 15. You must import HecDss from *hec.heclib.dss*.

Example 15: Opening and Releasing a DSS File

```
from hec.heclib.dss import *

theFile = HecDss.open(“MyFile.dss”)
or
theFile = HecDss.open(“C:/temp/sample.dss”, 1)

#  Finished using the file – release it
theFile.done()
```

Using one of the methods that specifies a time window affects the operation of the *get()* and *read()* methods as described in Tables 8.6 and 8.8 of the **Pattern Strings** section.

Note: The preferred manner of releasing access to an HEC-DSS file in most cases is to use the *done()* method and not the *close()* method. This is because HEC-DSS maintains information about file access and can grant access to a file released with *done()* much more quickly than it can to one released with *close()*. HEC-DSS will close files when a script exits or when they have not been accessed for a while. The only time that the *close()* method should be called explicity is to perform some operating system operation on it, such as renaming or deleting the file.

8.7.2 HecDss Retrieve and Store Functions

HecDss objects are used to retrieve and store data sets in a DSS file, as well as several utility functions for DSS files. Data is retrieved and stored using **DataContainers**. **DataContainers** are simple classes that are database independent and are how most **Hec** classes exchange data. **DataContainers** are a base class for several data types, such as time series data, paired data, gridded data, etc. **DataContainers** and their related classes are described following the **HecDss** class. If the data set you are retrieving does not exist, the **DataContainer** will have the *numberValues* set to 0 (zero).

HecDss, and most classes accessed through scripts, use Java Exceptions for error handling. Use a try block to catch and process errors. If data cannot be stored, **HecDss** will throw an exception. Refer to section 8.3.11 on **Handling Exceptions** for more information.

Tip: From the HEC-DSSVue main screen, select a pathname, right click and select **Copy Pathnames to Clipboard**. You can then paste the pathname into your script and avoid having to type it. You cannot use *condensed* pathnames for these functions.

The **HecDss** primary retrieve and storage methods are described in Table 8.6. Additional primary methods are shown in Table 8.7 and secondary methods are shown in Table 8.6. Time Series Data Storage Methods (Regular and Irregular) are shown in Table 8.9.

Pattern Strings

The pattern strings used with the getCatalogedPathnames(…) methods filter the list of returned pathnames in a user-specified manner. Pattern strings can be specified in two modes: pathname mode and pathname part mode. Both modes make use of pathname part filters:

Table 8.6 HecDss Retrieval and Storage Methods

Method	Returns	Description
get(String pathname)	DataContainer	Retrieves data for the single pathname and returns in a **DataContainer**. **DataContainer** will be a **TimeSeriesContainer**, **PairedDataContainter**, etc., depending on the data type. If a default time window has been set via one of the *open()* or *setTimeWindow()* methods, the "D" part of a time series pathname is ignored and the data for the time window is retrieved.
get(string pathname, Boolean getEntireDataSet)	DataContainer	Ignores the "D" part of time series data and retrieves data for the entire date span in the DSS file.
get(DSSIdentifier dssid)	DataContainer	Retrieves data for the pathname and time window given in the dssid
put(DataContainer dataContainer)	None	Store the data in **DataContainer** in DSS. The **DataContaine**r must be a **TimeSeriesContainer** or **PairedDataContainte**r, etc. Throws an exception if the data cannot be stored

Table 8.7 HecDss Additional Primary Methods

Method	Returns	Description
done()	None	Tells HEC-DSS that you are finished using it and releases the file. Use this instead of *close()*.
getPathnameList()	list	Returns a list of the pathnames in the file.
getTimeSeriesExtents(String pathname, HecTime start, HecTime end)	Boolean	Sets the contents of the start and end parameters to the date/time of the first and last piece of data in the entire data span. Returns whether the pathname exists.
isOpened ()	Boolean	Returns if the DSS file is accessible.
open(String dssFileName)	HecDss	Opens the DSS file and returns a **HecDss** object for further access. Creates the file if it does not exist. Throws an exception if the file cannot be created or opened.
open(String dssFileName, Boolean fileMustExist)	HecDss	Opens the DSS file and returns a **HecDss** object for further access. Throws an exception if you indicate the file must exist and it does not.
recordExists(String pathname)	Boolean	Tests to see if the single record exists.

Example 16: Reading from DSS

```
from hec.script import *
from hec.heclib.dss import *
import java

try :
  try :
    myDss = HecDss.open(“C:/temp/sample.dss”)
    flow = myDss.get(“/RUSSIAN/NR UKIAH/FLOW/01MAR2006/1HOUR//”)
    plot = Plot.newPlot(“Russian River at Ukiah”)
    plot.addData(flow)
    plot.showPlot()
  except Exception, e :
    MessageBox.showError(‘ ‘.join(e.args), “Python Error”)
  except java.lang.Exception, e :
    MessageBox.showError(e.getMessage(), “Error”)
finally :
  myDss.done()
```

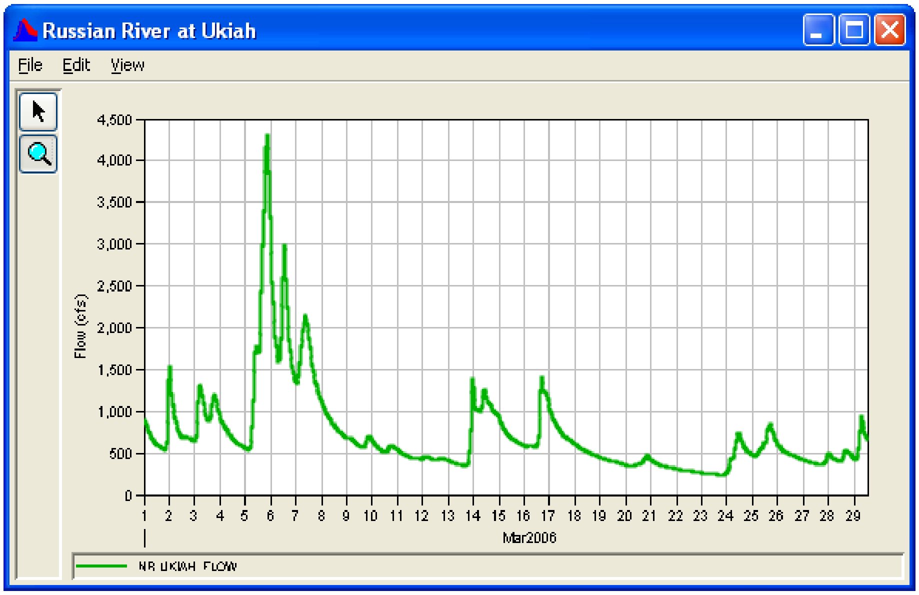


Figure 8.10 Example 16 Results

Table 8.8 HecDss Secondary Methods

Method	Returns	Description
close()	None	Close the DSS file. Only call this when you need the file closed (e.g. to rename it); otherwise call done()
copyRecordsFrom(String toDSSFilename, list of strings pathnameList)	integer Success = 0 Fail < 0	Copy records from the open DSS file to the DSS file name specified in the call.
copyRecordsFrom(String toDSSFilename, Vector pathnameList)	integer Success = 0 Fail < 0	Copy records from the open DSS file to the DSS file name specified in the call.
copyRecordsInto(String fromDSSFilename, list of strings pathnameList)	integer Success = 0 Fail < 0	Copy records from the file specified in the call into the currently open DSS file

Method	Returns	Description
copyRecordsInto(string fromDSSFilename, Vector pathnameList)	integer Success = 0 Fail < 0	Copy records from the file specified in the call into the currently open DSS file
delete(list of strings pathnameList)	integer Success = 0 Fail < 0	Deletes the records corresponding the list of pathnames
delete(Vector pathnameList)	integer Success = 0 Fail < 0	Deletes the records corresponding the list of pathnames
duplicateRecords(list of strings pathnameList, list of strings newPathnameList)	integer Success > 0 Fail < 0	Duplicates the records in the first list giving the names in the second list (same DSS file)
duplicateRecords(Vector pathnameList, Vector newPathnameList)	integer Success > 0 Fail < 0	Duplicates the records in the first list giving the names in the second list (same DSS file)
getCatalogedPathnames()	list of strings	Returns a list of all cataloged pathnames without generating a new catalog.
getCatalogedPathnames(Boolean forceNew)	list of strings	Returns a list of all cataloged pathnames, optionally generating a new catalog first.
getCatalogedPathnames(string pattern)	list of strings	Returns a list of all cataloged pathnames that match the specified pattern without generating a new catalog.
getCatalogedPathnames(string pattern, Boolean forceNew)	list of strings	Returns a list of all cataloged pathnames that match the specified pattern, optionally generating a new catalog first.
getDataManager()	CombinedDataManager	Gets the data manager for this DSS file
getEndTime()	string	Returns the end time set
getFileName()	string	Returns the name of the DSS file
getStartTime()	String	Returns the start time set
isRemote()	Boolean	Returns if the DSS file is being served on a remote machine (client/server mode)
read(string pathname)[1]	HecMath	Returns an **HecMath** object that holds the data set specified by pathname. If a default time window has been set via one of the *open()* or *setTimeWindow()* methods, the "D" part of a time series pathname is ignored and the data for the time window is retrieved.
read(string pathname, string timeWindow) [1]	HecMath	Returns an **HecMath** object that holds the data set specified by pathname with the specified time window.

Method	Returns	Description
read (string pathname, string startTime, string endTime) [1]	HecMath	Returns an HecMath object that holds the data set specified by pathname with the specified time window.
renameRecords (list of strings pathnameList, list of strings newPathnameList)	integer Success > 0 Fail < 0	Renames the records in the first list to those in the second list.
renameRecords (Vector pathnameList, Vector newPathnameList)	integer Success > 0 Fail < 0	Renames the records in the first list to those in the second list.
setIrregularStoreMethod (integer storeMethod)	None	See table following
setRegularStoreMethod (integer storeMethod)	None	See table following
setTimeWindow(string timeWindow)	None	The default time window for this DSSFile.
setTimeWindow(string startTime, string endTime)	None	The default time window for this DSSFile.
setTrimMissing(Boolean trim)	None	Sets whether TimeSeriesMath objects retrieved via calls to read(…) will have missing data trimmed from the beginning and end of the time window. [2]
write(HecMath dataset) [1]	integer	Write the data set to the DSS file. A return value of zero indicates success.
write (TimeSeriesMath timeSeriesData, string storeMethod)	integer	See table following

[1] Currently, the *read(…)* and *write(…)* methods can operate only on time-series data, paired data stream rating data, and text data represented by **TimeSeriesMath, PairedDataMath, StreamRatingMath**, and **TextMath** objects, respectively. Other record types, such as text data and gridded data are not yet supported by these methods.

[2] By default, **TimeSeriesMath** objects retrieved via calls to *read(…)* contain data only between the first non-missing value and the last non-missing value within the specified time window. Calling *setTrimMissing (False)* causes the data retrieved to include all data for the specified time window, including blocks of missing values at the beginning and end of the time window.

- Pathname mode = ***/****Afilter****/****Bfilter****/****Cfilter****/****Dfilter****/****Efilter****/****Ffilter****/***
- Pathname part mode = [**A=***Afilter*] [**B=***Bfilter*] [**C=***Cfilter*] [**D=***Dfilter*] [**E=***Afilter*] [**F=***Ffilter*]

In pathname mode, filters must be supplied for all pathname parts. In pathname part mode, only those parts that will not match everything must be specified.

Filters are comprised of the following components:

- **Normal text characters.** These characters are interpreted as they appear. For example, a pattern string of "B=XYZ" specifies matching every pathname that has a B-part of "XYZ".

Example 17: Reading a complete data set from DSS

```
from hec.script import *
from hec.heclib.dss import *
import java

try :
  try :
    myDss = HecDss.open(“C:/temp/sample.dss”)
    flow = myDss.get(“/MISSISSIPPI/ST. LOUIS/FLOW//1DAY/OBS/”, 1)

    if flow.numberValues == 0 :
      MessageBox.showError(“No Data”, “Error”)
    else :
      plot = Plot.newPlot(“Mississippi”)
      plot.addData(flow)
      plot.showPlot()

  except Exception, e :
    MessageBox.showError(‘ ‘.join(e.args), “Python Error”)
  except java.lang.Exception, e :
    MessageBox.showError(e.getMessage(), “Error”)
finally :
  myDss.done()
```

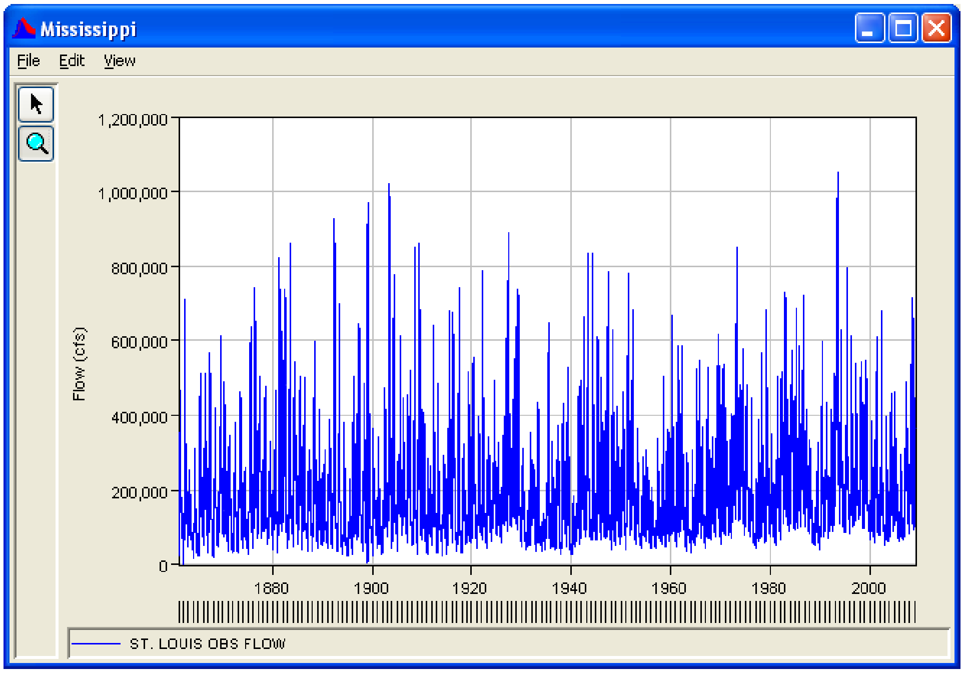


Figure 8.11 Example 17 Results

- **Special characters.** The special characters are comprised of the following list:
  - **'@'** or **'*'** (used interchangeably). This character can be specified as the first and/or last character of a filter and specifies matching a string of zero or more characters. For example, a pattern string of "B=XYZ@" specifies matching every pathname that as a B-part that begins with "XYZ". A pattern string of "B=*XYZ" specifies matching every

Example 18: Writing time series data to DSS

```
from hec.script import *
from hec.heclib.dss import *
from hec.heclib.util import *
from hec.io import *
import java

try :
  try :
    myDss = HecDss.open(“C:/temp/test.dss”)
    tsc = TimeSeriesContainer()
    tsc.fullName = “/BASIN/LOC/FLOW//1HOUR/OBS/”
    start = HecTime(“04Sep1996”, “1330”)
    tsc.interval = 60
    flows = [0.0,2.0,1.0,4.0,3.0,6.0,5.0,8.0,7.0,9.0]
    times = []
    for value in flows :
      times.append(start.value())
      start.add(tsc.interval)
    tsc.times = times
    tsc.values = flows
    tsc.numberValues = len(flows)
    tsc.units = “CFS”
    tsc.type = “PER-AVER”
    myDss.put(tsc)

  except Exception, e :
    MessageBox.showError(‘ ‘.join(e.args), “Python Error”)
  except java.lang.Exception, e :
    MessageBox.showError(e.getMessage(), “Error”)
finally :
  myDss.done()


Output:
  -----DSS---ZOPEN:  New File Opened,  File: C:/temp/test.dss
           Unit:   71;  DSS Version: 6-QD
 -----DSS---ZWRITE:  /BASIN/LOC/FLOW/01SEP1996/1HOUR/OBS/
  -----DSS---ZCLOSE Unit:   71,   File: C:/temp/test.dss
       Pointer Utilization:  0.25
       Number of Records:      1
       File Size:     70.2  Kbytes
       Percent Inactive:   0.0
```

pathname that has a B-part that ends with "XYZ". A pattern string of “B=*XYZ*” specifies matching every pathname that "XYZ" anywhere in the B-part. Note that the **@** or ***** character must be the first and/or last character of the filter (e.g. a pattern string of "B=ABC*XYZ" is invalid).

- **'#'** or **'!'** (used interchangeably). This character must be specified as the first character of a filter and specifies matching of every string *except* the remainder of the filter (e.g. it negates the remainder of the filter). A pattern string of "B=!XYZ*" specifies matching every pathname that does *not* have a B-part that begins with "XYZ".

Table 8.9 Time Series Data Storage Methods (Regular and Irregular)

	Integer	String	Description
Regular	0	REPLACE_ALL	Overwrite every value in the existing data
	1	REPLACE_MISSING_VALUES_ONLY	Don't overwrite non-missing values in the existing data
	2	REPLACE_ALL_CREATE	REPLACE_ALL + if new data has entire records of missing data, create empty records in existing data
	3	REPLACE_ALL_DELETE	REPLACE_ALL + if new data has entire records of missing data, delete records from existing data
	4	REPLACE_WITH_NON_MISSING	Overwrite every value in the existing data only if replacement is non-missing
Irregular	0	MERGE	Result for time window will be combination of existing and specified data
	1	DELETE_INSERT	Result for time window will be specified data only

- **No character**. The absence of any character specifies an empty filter, which matches a blank pathname part. Both of the following pattern strings match every pathname that has a blank A-part:
  - "//*/*/*/*/*/"
  - "A="

Using *getCatalogedPathnames(…)* with a pattern string utilizes the pathname matching in the underlying DSS library. To accomplish more sophisticated pathname filtering, use one of the *getCatalogedPathnames(…)* methods in conjunction with python text parsing and matching utilities.

Time Windows

Dates and times used to specify time windows should not contain blank characters and should include the 4 digit year. An example date and time is "04MAR2003 1400". If a single string is used to specify the time window, the starting date and time must precede the ending date and time, for example *dssFile.setTimeWindow* ("04MAR2003 1400 06APR2004 0900"). A relative time may be used in the single string command, where the letter "T" represents the current time, and the days or hours can be subtracted or added to that. For example *dssFile.setTimeWindow("T-14D T-2H")* would specify the starting time as current time minus fourteen days and the ending time as the current time minus two hours.

Time Windows (specified by starting time and ending time) effect how the *DSSFile.read(…)* method operates for the various data types.

- **Time-series data.** The time window supplied to DSS.open(…) or *dssFile*.setTimeWindow(…) specifies the default time window for all subsequent *dssFile.read(…)* operations. If no time window is supplied to *DSS.open(…)* then the default time window is undefined until *dssFile.setTimeWindow(…)* is called. If the default time window is undefined, then all *dssFile*.read(…) operations involving time-series data *must* specify a time window either implicitly via the D-part of the specified pathname or explicitly via the *startTime* and *endTime* parameters. The order of precedence for time windows is as follows:
  - **Explicit time window.** Specified in *dssFile.read(pathname, startTime, endTime)* or *dssFile.read(pathname, timeWindow)*. The D-part of the pathname is ignored and may be empty.
  - **Default time window.** Specified in *DSS.open(filename, startTime, endTime)* or *DSS.open(filename, timeWindow)* or *dssFile.setTimeWindow(startTime, endTime)* or *dssFile.setTimeWindow(timeWindow)*. The D-part of the pathname is ignored and may be empty. The default time window can be set to undefined by calling *dssFile.setTimeWindow("”, “")*.
  - **Implicit time window.** Specified as the D-part of the pathname supplied to *dssFile.read(pathname)* when the default time window is undefined. The D-part of the pathname must not be empty.
- **Paired data**. Time windows have no effect on reading paired data records.
- **Stream rating data**. Stream rating data are comprised of a time-series of individual rating records for a common location. The reading of stream rating data is not affected by default time windows. The implicit time window for reading stream rating data is the entire time span covered by the rating records. An explicit time window may be supplied by using the *dssFile.read(pathname, startTime, endTime)* method. If an explicit time window is specified, a (possible) subset of the rating records is retrieved that

cover the specified time window. The explicit time window is interpreted as the time window containing all time-series data that is to be rated via the stream rating data. The set of rating records retrieved for an explicit time window meets the following criteria.

- The earliest record retrieved is the latest rating that is on or before the start of the time window. If no such record exists, the earliest record in the rating time-series is retrieved.
- The latest record retrieved is the earliest rating that is on or after the end of the time window. If no such record exists, the latest record in the rating time-series is retrieved.
- All rating records between the earliest and latest retrieved records are also retrieved.

- **Text data**. Time windows have no effect on reading text records.

8.8 DataContainer Class

The **HecDss** *get()* method returns a **DataContainer** object and the put() method takes a **DataContainer** object as a parameter. **DataContainers** are a base for different types of data. The **TimeSeriesContainer** and **PairedDataContainer** classes discussed below are both types of **DataContainer** classes. **DataContainer** objects have no methods that can be called by the user, but all data fields of **DataContainer** objects are directly accessible. **DataContainer** data fields are described in Table 8.10.

You can also exchange **DataContainers** with **HecMath** objects. Both **TimeSeriesContainer** and **PairedDataContainer** classes are extracted from **HecMath** objects using the *getData()* method as documented in Section 8.15.35. **HecMath** objects can be updated from **DataContainer** objects using the *setData()* method as documented in Section 8.15.109.

Table 8.10 DataContainer Data Fields

Field	Type	Description
fullName	string	The full name associated with the data in the data store (DSS pathname, if the DataContainer is associated with a DSS file)
location	string	The location associated with the data (DSS pathname B-part if the DataContainer is associated with a DSS file).
subVersion	string	The sub-version associated with the data.
version	string	The version associated with the data (DSS pathname F-part if the DataContainer is associated with a DSS file).
watershed	string	The watershed associated with the data (DSS pathname A-part if the DataContainer is associated with a DSS file).

Example 19: Extracting a DataContainer from HecMath

```
from hec.script import *
from hec.hecmath import *
theFile = DSS.open(“myFile.dss”)
flowDataSet = theFile.read(“/BASIN/LOC/FLOW/01NOV2002/1HOUR/OBS/”)
theFile.done()
flowData = flowDataSet.getData()
thisWatershed = flowData.watershed
```

8.8.1 TimeSeriesContainer Class

TimeSeriesContainer is a type of DataContainer that contains information about time series data. TimeSeriesContainer objects are returned by the get () method and are required in the **put()** method of HecDss objects. TimeSeriesContainer objects have all the data fields described Section 8.8 (DataContainerClass) in addition to those described in Table 8.11. New TimeSeriesContainer objects can be created by a script if the TimeSeriesContainer class has been imported from the *hec.io* module (e.g. "from hec.io import *" or "from *hec.io* import TimeSeriesContainer"). The new object can be created by calling TimeSeriesContainer () (e.g. "myTSC = TimeSeriesContainer ()").

Table 8.11 TimeSeriesContainer Data Fields

Field	Type	Description
endTime	Integer[1]	The end time of the time window. If the data were retrieved, the end time may be later than the last time in the times list.
interval	Integer	The interval, in minutes, between each set of consecutive times in the times list. For irregular-interval times, the interval field is set to –1.
numberValues	Integer	The length of values and times lists.
parameter	string	The parameter associated with the data.
quality	list of integers[2]	The optional list of quality flags. If this list is present, there must be a quality for each value in the values array. If this list is not present, the quality field is set to None.
startTime	integer[2]	The start time of the time window. If the data were retrieved, the start time may be earlier than the first time in the times list.
subLocation	string	The sub-location associated with the data.
subParameter	string	The sub-parameter associated with the data.
times	list of integers[1]	The list of times. There must be a time for each value in the values list, and times must increase from one index to the next.
timeZoneID	string	The time-zone for times in the times list. If unknown, the timeZoneID field is set to None

Field	Type	Description
timeZoneRawOffset	integer	The offset, in milliseconds, from UTC to the time zone for the times in the times list.
type	string	The type of the data (e.g. "INST-VAL", "INST-CUM", "“PER-AVER", "PER-CUM").
units	string	The units of the data.
values	list of floating-point	The data values, each of which has a corresponding time in the times list and optionally a corresponding quality in the quality list. All lists must be the same length.

[1] Integer times from this field can be converted to string representations by using the *set()* and *dateAndTime()* methods of **HecTime** objects and can be generated by the **HecTime** *value()* method.

[2] Quality values are interpreted according to Table 8.12.

Table 8.12 Data Quality Bits

Bit(s)	Description
1	Set when the datum has been tested (screened).
2	Set when the datum passed all tests.
3	Set when the datum is missing (either originally missing or set to missing by a test).
4	Set when at least one test classified the datum as questionable.
5	Set when at least one test classified the datum as rejected.
6-7	Set by the RANGE test. Interpreted as a 2-bit unsigned integer with values having the following meanings: **0** value of datum < than 1st limit or no range test applied **1** 1st limit <= value of datum < 2nd limit **2** 2nd limit <= value of datum < 3rd limit **3** 3rd limit <= value of datum
8	Set when the datum has been changed from the original value.
9-11	Datum replacement indicator. Interpreted as a 3-bit unsigned integer with values having the following meanings: **0** datum was not replaced (original value) **1** datum was replaced by an automated tool **2** datum was replaced by an interactive tool (e.g., fill operaton) **3** datum was replaced by manual edit in an interactive tool **4** original value was accepted or restored in an interactive tool **5-7** reserved for future use
12-15	Datum replacement value computation method. Interpreted as a 4-bit unsigned integer with values having the following meanings: **0** datum was not replaced (original value) **1** datum value computed by linear interpolation **2** datum value was entered manually **3** datum value was replaced with a missing value **4** datuma value was entered graphically **5-15** reserved for future use
16	set when datum failed an absolute magnitude test
17	set when datum failed a constant value test

Bit(s)	Description
18	set when datum failed a rate of change test
19	set when datum failed a relative magnitude test
20	set when datum failed a duration magnitude test
21	set when datum failed a negative incremental value test
22	reserved for future use
23	set when datum is excluded from testing (e.g. DATCHK GAGEFILE entry)
24	reserved for future use
25	set when datum failed a user-defined test
26	set when datum failed a distribution test
27-31	reserved for future use
32	set when datum is protected from being replaced

Example 20: Using a TimeSeriesContainer Object

```
from hec.script import *
from hec.heclib.dss import *
from hec.io import TimeSeriesContainer
from hec.heclib.util import HecTime

watershed = “GREEN RIVER”
loc = “OAKVILLE”
param = “STAGE”
ver = “OBS”
startTime = “12Oct2003 0100”
values = [12.36, 12.37, 12.42, 12.55, 12.51, 12.47, 12.43, 12.39]
hecTime = HecTime()
tsc = TimeSeriesContainer()
tsc.watershed = watershed
tsc.location = loc
tsc.parameter = param
tsc.version = ver
tsc.fullName = “/%s/%s/%s//1HOUR/%s/” % \
        (watershed, loc, param, ver)
tsc.interval = 60
hecTime.set(startTime)
times = []
for value in values :
        times.append(hecTime.value())
        hecTime.add(tsc.interval)
tsc.values = values
tsc.times = times
tsc.startTime = times[0]
tsc.endTime = times[-1]
tsc.numberValues = len(values)
tsc.units = “FEET”
tsc.type = “INST-VAL”
dssFile = HecDss.open(“myFile.dss”)
dssFile.put(tsc)
dssFile.done()
```

8.8.2 PairedDataContainer Class

PairedDataContainer is a type of **DataContainer** that contains information about paired data. **PairedDataContainer** objects are returned by the *get ()* method. **PairedDataContainer** objects are described in Section 8.8 in addition to those described in Table 8.13. New **PairedDataContainer** objects can be created by a script if the **PairedDataContainer** class has been imported from the *hec.io* module (e.g. "from hec.io import *" or "from hec.io import PairedDataContainer"). The new object can be created by calling **PairedDataContainer ()** (e.g. "myPDC = PairedDataContainer ()").

Table 8.13 PairedContainer Data Fields

Field	Type	Description
date	string	The date associated with the paired-data.
datum	floating-point	The zero-stage elevation of a stream gauge.
labels	list of strings	The list of labels used to identify each of the y-ordinates lists. If there is only 1 y-ordinates list (e.g. numberCurves == 1), the labels field is set to None.
labelsUsed	Boolean	A flag specifying whether labels are used. This field is set to True if there is more than 1 y-ordinates list and False otherwise.
numberCurves	integer	The number of y-ordinates lists.
numberOrdinates	integer	The length of the x-ordinates list and each of the y-ordinates lists.
offset	floating-point	The offset value of a stream rating.
shift	floating-point	The shift value for a stream rating.
transformType	integer	Type of transformation to use (1 = "LINLIN", 2 = "LOGLOG")
xOrdinates	list of floating-point	The x-ordinate values. Each y-ordinate values list must be of the same length as this field.
xparameter	string	The parameter of the x-ordinates.
xtype	string	The type of the x-ordinates ("UNT" for unitary or "LOG" for logarithmic).
xunits	string	The units of the x-ordinates.
yOrdinates	list of lists of floating-point	The y-ordinate values. Each list of y-ordinate values must be of the same length as the list of x-ordinate values. The nth value of each y-ordinates list is associated with the nth value of the x-ordinates list.
yparameter	string	The parameter of the y-ordinates.
ytype	string	The type of the y-ordinates ("UNT" for unitary or "LOG" for logarithmic).
yunits	string	The units of the y-ordinates.

Example 21: Using a PairedDataContainer Object

```
from hec.script import *
from hec.heclib.dss import *
from hec.io import PairedDataContainer

watershed = “GREEN RIVER”
loc = “OAKVILLE”
xParam = “STAGE”
yParam = “FLOW”
date = “12Oct2003”
stages = [0.4, 0.5, 1.0, 2.0, 5.0, 10.0, 12.0]
flows = [0.1, 3, 11, 57, 235, 1150, 3700]
pdc = PairedDataContainer()
pdc.watershed = watershed
pdc.location = loc
pdc.parameter = param
pdc.version = ver
pdc.fullname = “/%s/%s/%s-%s///%s/” % \
    (watershed, loc, xParam, yParam, date)
pdc.xOrdinates = stages
pdc.yOrinates = [flows]
pdc.numberCurves = 1
pdc.numberOrdinates = len(stages)
pdc.labelsUsed = False
pdc.xunits = “FEET”
pdc.yunits = “CFS”
pdc.xtype = “LOG”
pdc.ytype = “LOG”
pdc.xparameter = xParam
pdc.yparameter = yParam
pdc.date = date
pdc.transformType = 2dssFile = HecDss.open(“myFile.dss”)
dssFile.put(pdc)
dssFile.done()
```

8.9 HecTime Class

HecTime objects are used to manipulate dates and times and to convert dates and times among different formats. To use **HecTime** objects, the **HecTime** class must be imported from the *hec.heclib.util* module (e.g. "from hec.heclib.util import *" or "from hec.heclib.util import HecTime"). After importing the class, a new HecTime object can be created by calling HecTime() (e.g. "myTime = HecTime()"). **HecTime** methods are described in Table 8.14.

Table 8.14 HecTime Methods

Method	Returns	Description
add(HecTime increment)	None	Adds the specified increment to the object’s date and time.
add(integer increment)	None	Adds the specified increment in minutes to the object’s date and time.

Method	Returns	Description
addDays(integer days)	None	Adds the specified number of days to the time
addHours (integer hours)	None	Adds the specified number of hours to the time
addMinutes (integer minutes)	None	Adds the specified number of minutes to the time
addSeconds (integer seconds)	None	Adds the specified number of seconds to the time
adjustToIntervalOffset(integer intervalMinutes, integer offset)	None	Changes the time to be at the standard time for that interval, according to HEC-DSS. For example, will adjust daily data to 2400 hours of that day.
compareTimes(HecTime other)	integer	Returns one of the following values: -1 The object's date and time is less than the other object's 0 The objects' dates and times are equal 1 The object's date and time is greater than the other object's
computeNumberIntervals(HecTime otherTime, integer intervalInMins)	integer	Determines the number of periods between the time set and the time passed in, where intervalInMins is the interval length.
convertTimeZone(HecTime hecTime, TimeZone fromTimeZone, TimeZone toTimeZone)	None	Changes the time passed in from the first time zone to the second time zone. Adjusts for daylight savings time, according to the time given.
date()	string	Returns a string representation of the object’s date. Same as date(2).
date(integer format)	string	Returns a string representation of the object’s date, formatted according to the integer parameter.[1]
dateAndTime()	string	Returns a string representation of the object’s date and time. Same as dateAndTime(2).
dateAndTime(integer format)	string	Returns a string representation of the object’s date and time, formatted according to the integer parameter. [1]
day()	integer	Returns the day portion of the object’s date as an integer
dayOfWeek()	integer	Returns the day of the week with Sunday starting as 1, Monday 2
dayOfWeekName()	string	Returns the name of the day of the week, such as "Sunday"

Method	Returns	Description
dayOfYear()	integer	Returns the Julian day of the year, with Jan.1 returned as "1".
equalTo(HecTime other)	Boolean	Returns True if the object's date and time is equal to the other object's date and time. Returns False otherwise.
greaterThan(HecTime other)	Boolean	Returns True if the object's date and time is greater (later) than the other object's date and time. Returns False otherwise.
greaterThanEqualTo(HecTime other)	Boolean	Returns True if the object's date and time is greater (later) than or equal to the other object's date and time. Returns False otherwise.
getJavaDate(integer minutesTimezoneOffset)	Date	Returns the Java Date for this time
getMinutes()	long	Returns the number of milliseconds since Jan 1, 1900
getTimeInMillis()	long	Returns the number of milliseconds since January 1, 1970, 00:00:00 GMT

getTimeWindow (String userLine, HecTime startTime, HecTime endTime)	integer
	Success=0
	Fail = -1
		Takes a user entered line with start date/time and end date/time and converts to HecTime.

getXMLDateTime (integer minutesTimezoneOffset)	string	Returns the date in XML format
hour()	integer	Returns the hours portion of the object’s time as an integer
isDefined()	Boolean	Returns True if the object is set to a valid date and time and False if not.
isDateDefined()	Boolean	Returns True if the date portion is valid (ignores time) and False if not.
isTimeDefined()	Boolean	Returns True if the time portion is valid (ignores date) and False if not.
increment(integer numberPeriods, integer minutesInPeriod)	None	Adds the number of periods to the date
isoDate()	integer	Returns the date in ISO format (YYMMDD)
isoTime()	integer	Returns the time in ISO format (HHMMSS)
lessThan(HecTime other)	Boolean	Returns True if the object's date and time is less (earlier) than the other object's date and time. Returns False otherwise.
lessThanEqualTo(HecTime other)	Boolean	Returns True if the object's date and time is less (earlier) than or equal to the other object's date and time. Returns False otherwise.

Method	Returns	Description
julian()	integer	Returns the number of days since Jan 1, 1900
minute()	integer	Returns the minutes portion of the object’s time as an integer
month()	integer	Returns the month portion of the object’s date as an integer
notEqualTo(HecTime other)	Boolean	Returns False if the object's date and time is equal to the other object's date and time. Returns True otherwise.
second()	integer	Returns the seconds portion of the object’s time as an integer
setCurrent()	None	Sets the date/time to the current time.
set (integer time)	None	Sets the object to the date and time represented by the integer (minutes since 31Dec1899 00:00)
set(string dateAndTime)	integer	Sets the object to the date and time represented by the string, and returns zero if successful.
set (string date, string time)	integer	Sets the object to the date represented by the date string and the time represented by the time string, and returns zero if successful.
set (HecTime time)	None	Sets the object to the date and time represented by the HecTime parameter.
setDate(string date)	integer	Sets the object to the date represented by the string, and returns zero if successful. The time portion of the object is not modified.
setTime(string time)	integer	Sets the object to the time represented by the string, and returns zero if successful. The date portion of the object is not modified.
setJulian(integer julian)	None	Sets the date to the Julian days (Since Jan 1, 1900)
setXML (String dateTime)	integer Success=0	Sets the date and time according to the XML string
setUndefined()	None	Sets the object to represent an undefined time, as if the object had just been created.
showTimeAsBeginningOfDay(Boolean showBeginning)	None	Specifies whether the object is to show midnight times as 00:00 (vs 24:00)

Method	Returns	Description
subtract(HecTime increment)	None	Subtracts the specified increment from the object’s date and time.
subtract(integer increment)	None	Subtracts the specified increment in minutes from the object’s date and time.
time()	string	Returns a string representation of the object’s time.
toString()	string	Returns the date and time in string format, for format style 2
toString(integer format)	string	Returns the date and time in string format, for format style given
value()	integer	Returns the object's date and time as in the number of minutes since 31Dec1899 0000.
year()	integer	Returns the year portion of the object’s date as an integer

[1] The format of the string returned by the *date(integer format)* method and the date portion of the string returned by the *dateAndTime(integer format)* method are displayed in Table 8.15.

Table 8.15 HecTime Date Formats

0	June 2, 1985	10	June 2, 85	100	JUNE 2, 1985	110	JUNE 2, 85
1	Jun 2, 1985	11	Jun 2, 85	101	JUN 2, 1985	111	JUN 2, 85
2	2 June 1985	12	2 June 85	102	2 JUNE 1985	112	2 JUNE 85
3	June 1985	13	June 85	103	JUNE 1985	113	JUNE 85
4	02Jun1985	14	02Jun85	104	02JUN1985	114	02JUN85
5	2Jun1985	15	2Jun85	105	2JUN1985	115	2JUN85
6	Jun1985	16	Jun85	106	JUN1985	116	JUN85
7	02 Jun 1985	17	02 Jun 85	107	02 JUN 1985	117	02 JUN 85
8	2 Jun 1985	18	2 Jun 85	108	2 JUN 1985	118	2 JUN 85
9	Jun 1985	19	Jun 85	109	JUN 1985	119	JUN 85

8.10 Plotting Basics

The title, viewport, axis label, axis tics, and legend of a plot, each of which are accessible via scripts, are identified in Figure 8.12.

Example 22: Creating a Plot

```
myPlot = Plot.newPlot()
or
thePlot = Plot.newPlot(“Elevation vs Flow”)
```

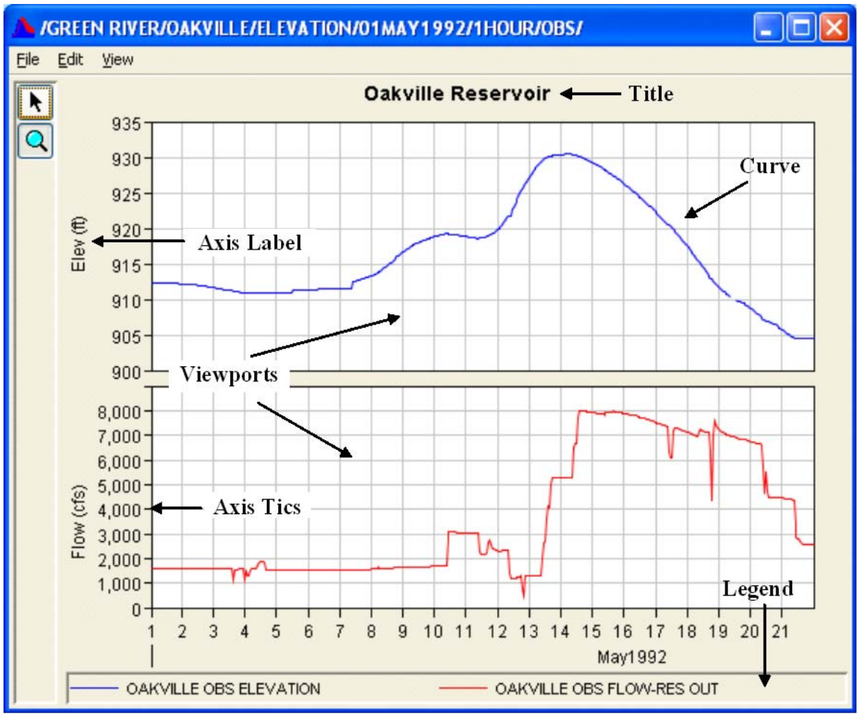


Figure 8.12 Plot Components

8.10.1 Plot Class

```
Plot.newPlot()
Plot.newPlot(string title)
```

The **Plot** class in the *hec.script* module is used to create a new Plot dialog. It contains two methods to create a Plot dialog, each of which returns a G2dDialog object.

8.10.2 Changing Plot Component Attributes

Not all Plot Component attributes are visible by default, and setting the attribute may not make that attribute visible. Often it is necessary to set the visibility of the attribute by calling *setAttributeVisible(True)*. Reading a flow data set from a DSS file, plotting the data set, setting the minor Y grid color to black and making it display are illustrated in Example 23.

8.10.3 G2dDialog Class

G2dDialog objects are the dialog that plots display in. **G2dDialog** methods are described in Table 8.16.

Example 23: Plotting DSS Data

```
from hec.script import *
from hec.script.Constants import TRUE, FALSE

theFile = HecDss.open(“myFile.dss”)          # open myFile.dss
thePath = “/BASIN/LOC/FLOW/01NOV2002/1HOUR/OBS/”
flowDataSet = theFile.get(thePath)           # read a path name
thePlot = Plot.newPlot()                     # create the plot
thePlot.addData(flowDataSet)                 # add the flow data set to the plot
thePlot.showPlot()                           # show the plot
viewport0=thePlot.getViewport(0)             # get the first viewport
viewport0.setMinorGridYColor(“black”)        # set the viewport’s minor Y grid to black
viewport0.setMinorGridYVisible(TRUE)         # tell the minor Y grid to display
```

Table 8.16 G2dDialog Methods

Method	Returns	Description
addData(DataContainer dc)	None	Add the DataContainer specified by dc to the plot. Must be called before showPlot(). *Do not use this if a PlotLayout object is used on this plot.*
applyTemplate(string templateFile)	None	Apply the given template to this plot
configurePlotLayout()	None	Display the "Configure Plot Layout" dialog for this plot
configurePlotLayout(PlotLayout layout)	None	Configures the plot layout for this plot according to the specified PlotLayout object. *If this method is used, do not use the addData() method with the same plot.*
close()	None	Closes the plot
configurePlotTypes()	None	Display the configure plot types dialog
copyToClipboard()	None	Copy the plot to the system clipboard
defaultPlotProperties()	None	Display the default plot properties dialog
exportProperties()	None	Allows you to save the properties of the plot to a disk.
exportProperties(string templateName)	None	Allows you to save the properties of the plot to the file specified by templateName.
getCurve(HecMath filenam)	G2dLine	Return the G2dLine for the DataSet specified by dataSet

Method	Returns	Description
getCurve(string dssPath)	G2dLine	Return the G2dLine for the path specified in dssPath
getHeight()	integer	Return the height of the dialog in screen coordinates.
getLegendLabel(DataContainer dc)	G2dLabel	Return the legend label object for the specified data container.
getLocation()	Point	Return the location of the dialog in screen coordinates.[1]
getPlotTitle()	G2dTitle	Return the title for the G2dDialog
getPlotTitleText()	string	Return the text of the title for the G2dDialog
getSize()	Dimension	Return the dimensions of the dialog in screen coordinates.
getViewport(HecMath filenam)	Viewport	Return the Viewport that contains the curve specified by dataSet
getViewport(integer viewportIndex)	Viewport	Return the viewport at index specified by viewportIndex
getViewport(string dataSetPath)	Viewport	Return the Viewport that contains the curve specified by dataSetPath
getWidth()	Integer	Return the width of the dialog in screen coordinates.
hide()	None	Hide the dialog
iconify()	None	Minimize (iconify) the dialog
isPlotTitleVisible()	Boolean	Return the visibility state of the title of this plot.
maximize()	None	Maximize the dialog
minimize()	None	Minimize (iconify) the dialog
newPlotLayout()	PlotLayout	Return a PlotLayout object that can be used to configure the layout of this plot.
plotProperties()	None	Display the plot properties dialog for this plot
print()	None	Display the print dialog for this plot

Method	Returns	Description
printMultiple()	None	Display the print multiple dialog for this plot
printPreview()	None	Display the print preview dialog for this plot
printToDefault()	None	Prints using the printer defaults such as page format and printer. This method does not display the printer dialog for user interaction.
repaint()	None	Forces the plot to be refreshed.
restore()	None	Restore the dialog from a minimized or maximized state
saveAs()	None	Display the saveAs dialog for this plot
saveToJpeg(string filename)	None	Save the plot to the Jpeg file specified by fileName
saveToJpeg(string filename, integer quality)	None	Save the plot to the Jpeg file specified by filename, with the specified quality[2].
saveToMetafile(string filename)	None	Save the plot to the Windows Meta file specified by filename
saveToPng(string filename)	None	Save the plot to the Portable Network Graphics file specified by filename
saveToPostscript(string filename)	None	Save the plot to the PostScript file specified by filename
setLegendBackgound(string color)	None	Sets the background color of the legend.
setLegendLabelText(DataContainer dc, string text)	None	Sets the legend label text
setLegendLocation(string location)	None	Sets the location of the legend[3].
setLocation(integer x,integer y)	None	Sets the location of the dialog in screen coordinates.[1]
setPlotTitleText(string text)	None	Sets the text of the title for this plot
setPlotTitleVisible(Boolean state)	None	Sets the visibility of the title for this plot

Method	Returns	Description
setSize(integer width, integer height)	None	Sets the size of the dialog in screen coordinates.
setVisible(Boolean visible)	None	Makes the plot visible
showPlot()	None	Show the dialog
stayOpen()	None	Keeps the plot on the screen for batch mode only
tabulate()	HecDataTableFrame	Display the table view of this plot

[1] The coordinate system used is a graphics coordinate system, where X increases to the right and Y increases downward from the origin (0,0) which is located in the top left corner of the display. Locations set or retrieved refer to the top left corner of the plot in reference to this coordinate system.

[2] The specified quality is limited to an effective range of 0 – 100, inclusive. Higher qualities produce larger files and take longer to generate. The *saveToJpeg(fileName)* call currently produces the same results as *saveToJpeg(fileName, 75)*.

[3] Valid legend locations are "Right" and "Bottom".

Example 24: Plot Dialog

```
from hec.script import *                              # for Plot class
from hec.heclib.dss import *                          # for DSS class
theFile = HecDss.open(“myFile.dss”)                   # open myFile.dss
thePath = “/BASIN/LOC/FLOW/01NOV2002/1HOUR/OBS/"
flowDataSet = theFile.get(thePath)                    # read a path name
thePlot = Plot.newPlot()                              # create a new Plot
thePlot.addData(flowDataSet)                         # add flow data container
thePlot.showPlot()                                    # show the plot
thePlot.setLocation(50,50)                            # moves plot to 50,50
```

8.10.4 PlotLayout Class

PlotLayout objects hold information about the layout of the plot dialog. The use of **ViewportLayout** objects, in conjunction with **PlotLayout** objects, allows scripts to specify the same layout information accessible interactive via the "Configure Plot Layout" dialog. A **PlotLayout** object is obtained by calling *Plot.newPlotLayout()*. **PlotLayout** methods are described in Table 8.17.

8.10.5 ViewportLayout Class

ViewportLayout objects hold information about the layout of an individual viewport within the plot dialog. The use of **ViewportLayout** objects, in conjunction with **PlotLayout** objects, allows scripts to specify the same layout information accessible interactive via the "Configure Plot Layout" dialog. A **ViewportLayout** object is obtained by calling one of

Table 8.17 PlotLayout Methods

Method	Returns	Description
addViewport()	ViewportLayout	Adds a ViewportLayout to the PlotLayout with a default weight of 100. Returns a reference to the new ViewportLayout.
addViewport(floating-point weight)	ViewportLayout	Adds a ViewportLayout to the PlotLayout with the specified weight. Returns a reference to the new ViewportLayout.
hasLegend()	Boolean	Returns whether this PlotLayout is configured to display the legend.
hasToolbar()	Boolean	Returns whether this PlotLayout is configured to display the toolbar.
getViewportCount()	integer	Returns the number of ViewportLayout objects currently in the PlotLayout object.
getViewports()	java.util.List of ViewportLayouts	Returns the ViewportLayout objects currently in the PlotLayout object.
getViewportWeights()	list of floating-points	Returns the weights of the ViewportLayout objects currently in the PlotLayout object.
setHasLegend(Boolean state)	None	Configures the PlotLayout object to display the legend or not, depending upon the specified state.
setHasToolbar(Boolean state)	None	Configures the PlotLayout object to display the toolbar or not, depending upon the specified state.

the *addViewport* methods of a **PlotLayout** object. **ViewportLayout** objects are only used to configure the plot layout. Manipulation of axis labels, background colors, etc. is performed using Viewport objects as described in table below. **ViewportLayout** methods are described in Table 8.18.

Table 8.18 ViewportLayout Methods

Method	Returns	Description
addCurve(string axis, DataContainer curve)	None	Adds the specified curve to the specified axis of the ViewportLayout object.
getMajorGridXStyleString()	string	
getMajorGridYStyleString()	string	
getMinorGridXStyleString()	string	
getMinorGridYStyleString()	string	

Method	Returns	Description
getY1Data()	List	Returns a *java.util.List* of all curves that have been added to the Y1 axis of this object
getY2Data()	List	Returns a *java.util.List* of all curves that have been added to the Y2 axis of this object
hasY1Data()	Boolean	Returns whether any curves have been added to the Y1 axis of this object
hasY2Data()	Boolean	Returns whether any curves have been added to the Y2 axis of this object
setMajorGridXStyleString(string majorGridXStyle)	None	
setMajorGridYStyleString(string majorGridYStyle)	None	
setMinorGridXStyleString(string minorGridXStyle)	None	
setMinorGridYStyleString(string minorGridYStyle)	None	
setLinear(string axisName)	None	
setLogarithmic(string axisName)	None	
scaleAxisFromOpposite(string axis)	None	

The script in Example 25 reads precipitation, stage and flow data set from a DSS file, and configures a plot to display the precipitation on top in a viewport that occupies thirty percent of the available space and to display the stage and flow on separate axes of a bottom viewport that occupies the remaining seventy percent of available space.

Example 25: PlotLayout and ViewportLayout Objects

```
from hec.script import *                          # for Plot class
from hec.heclib.dss import *                      # for DSS class
theFile = HecDss.open(“myFile.dss”)               # open myFile.dss
precipPath = “/BASIN/LOC/FLOW/01NOV2002/1HOUR/OBS/"
stagePath = “/BASIN/LOC/FLOW/01NOV2002/1HOUR/OBS/"
flowPath = “/BASIN/LOC/FLOW/01NOV2002/1HOUR/OBS/"
precipData = theFile.get(precipPath)              # read the precip
stageData = theFile.get(stagePath)                # read the stage
flowData = theFile.get(flowPath)                  # read the flow
thePlot = Plot.newPlot()                          # create a new Plot
layout = Plot.newPlotLayout()                     # create a new PlotLayout
topView = layout.addViewport(30)                  # get the top viewport
bottomView = layout.addViewport(70)               # get the bottom viewport
topView.addCurve(“Y1”, precipData)                # add the precip to top
bottomView.addCurve(“Y1”, stageData)              # add the stage to bottom
bottomView.addCurve(“Y2”, flowData)               # add the flow to bottom
thePlot.configurePlotLayout(layout)               # configure the plot
thePlot.showPlot()                                # show the plot
```

8.10.6 Viewport Class

Viewport objects hold the data set curves. **Viewport** methods are described in Table 8.19.

Table 8.19 Viewport Methods

Method	Returns	Description
addAxisMarker(AxisMarker marker)	None	Adds a marker line described by the AxisMarker parameter
addXAxisMarker()	None	Display the Axis Marker Properties Dialog for a marker line to add to the X axis
addXAxisMarker(floating-point value)	None	Add an X Axis marker at the location specified by value
addXAxisMarker(string value)	None	Add a X Axis marker at the location specified by value
addYAxisMarker()	None	Display the Axis Marker Properties Dialog for a marker line to add to the Y axis
addYAxisMarker(string value)	None	Add a Y Axis marker at the location specified by value
editProperties()	None	Display the Edit Properties dialog for this Viewport
getAxis(string axisName)	Axis	Return the **Axis** specified by **axisName** for this Viewport
getAxisLabel(string axisName)	AxisLabel	Return the **AxisLabel** for the axis specified by **axisName** for this Viewport
getAxisTics(string axisName)	AxisTics	Return the **AxisTics** for the axis specified by **axisName** for this Viewport
getBackground()	Color	Return the background color for the Viewport as a Color.
getBackgroundString()	string	Return the background color name for the Viewport as a string.
getBorderColor()	Color	Return the border color for the Viewport as a Color.
getBorderColorString()	string	Return the background color name for the Viewport as a string
getBorderWeight()	float	Return the border weight for this Viewport
getFillPatternString()	string	Return the fill pattern for this Viewport as a String
getMajorGridXColor()	Color	Return the color of the vertical lines of the major grid for this Viewport as a Color

Method	Returns	Description
getMajorGridXColorString ()	string	Return the color of the vertical lines of the major grid for this Viewport as a string
getMajorGridXWidth ()	floating point	Return the width of the vertical lines of the major grid for this Viewport
getMajorGridYColor ()	Color	Return the color of the horizontal lines of the major grid of this Viewport as a Color
getMajorGridYColorString ()	string	Return the color of the horizontal lines of the major grid for this Viewport as a string
getMajorGridYWidth ()	floating point	Return the width of the vertical lines of the major grid for this Viewport
getMinorGridXColor()	Color	Return the color of the vertical lines of the minor grid for this Viewport as a color
getMinorGridXColorString()	string	Return the color of the vertical lines of the minor grid for this Viewport as a string
getMinorGridXWidth()	floating point	Return the width of the vertical lines of the minor grid for this Viewport
getMinorGridYColor()	Color	Return the color of the horizontal lines of the minor grid for this Viewport as a Color.
getMinorGridYColorString()	string	Return the color of the horizontal lines of the minor grid for this Viewport as a string
getMinorGridYWidth()	floating point	Return the width of the vertical lines of the minor grid for this Viewport
isBackgroundVisible()	Boolean	Return whether the background is drawn for this Viewport
isBorderVisible()	Boolean	Return whether the border is drawn for this Viewport
isMajorGridXVisible()	Boolean	Return whether the vertical lines of the major grid are drawn for this Viewport
isMajorGridYVisible ()	Boolean	Return whether the horizontal lines of the major grid are drawn for this Viewport

Method	Returns	Description
isMinorGridXVisible()	Boolean	Return whether the vertical lines of the minor grid are drawn for this Viewport
isMinorGridYVisible ()	Boolean	Return whether the horizontal lines of the minor grid are drawn for this Viewport
setBackground(string colorString)	None	Set the background to the color specified by colorString
setBorderColor(string borderColor)	None	Set the border color for this Viewport
setBorderWeight(floating-point borderWeight)	None	Set the border weight for this Viewport
setBackgroundVisible(Boolean state)	None	Set whether to draw the background for this Viewport
setBorderVisible(Boolean state)	None	Set whether to draw the border for this Viewport
setFillPattern(string pattern)	None	Set the fill pattern for this Viewport
setGridColor(string colorString)	None	Set the color of the horizontal and vertical lines of the major and minor grids for this Viewport.
setGridXColor(string colorString)	None	Set the color of the vertical lines of the major and minor grids for this Viewport.
setGridYColor(string colorString)	None	Set the color of the horizontal lines of the major and minor grids for this Viewport.
setMajorGridXColor(string majorGridXColor)	None	Set the color of the vertical lines of the major grid for this Viewport.
setMajorGridXVisible(Boolean state)	None	Set whether to draw the vertical lines of the major grid for this Viewport
setMajorGridXWidth(floating-point gridLineWidth)	None	Set the width of the vertical lines of the major grid for this Viewport
setMajorGridYColor(string majorGridYColor)	None	Set the color of the horizontal lines of the major grid for this Viewport.
setMajorGridYVisible(Boolean state)	None	Set whether to draw the horizontal lines of the major grid for this Viewport
setMajorGridYWidth(floating-point gridLineWidth)	None	Set the width of the horizontal lines of the major grid for this Viewport
setMinorGridXColor(string minorGridXColor)	None	Set the color of the vertical lines of the minor grid for this Viewport.

Method	Returns	Description
setMinorGridXVisible(Boolean state)	None	Set whether to draw the vertical lines of the minor grid for this Viewport
setMinorGridXWidth(floating-point gridLineWidth)	None	Set the width of the vertical lines of the minor grid for this Viewport
setMinorGridYColor(string minorGridYColor)	None	Set the color of the horizontal lines of the minor grid for this Viewport.
setMinorGridYVisible(Boolean state)	None	Set whether to draw the horizontal lines of the minor grid for this Viewport
setMinorYGridWidth(floating-point gridLineWidth)	None	Set the width of the horizontal lines of the minor grid for this Viewport

8.10.7 AxisMarker Class

AxisMarker objects hold complete descriptions of marker lines to be added to viewports. **AxisMarker** objects have fields that are settable by the user to create marker lines of various styles. New **AxisMarker** objects are created by calls to *AxisMarker()* (e.g., *myMarker = AxisMarker()*).

AxisMarker fields are described in Table 8.20.

Table 8.20 AxisMarker Fields

Field	Type	Description	Default
axis	string	"X" or "Y"	"Y"
fillColor	string	Color of the filled area.	"black"
fillPattern	string	Pattern of the filled area.	"solid"
fillStyle	string	Specifies whether the filled area is to be above or below the marker line, or to not fill at all.	"none"
labelAlignment	string	Specifies whether the label text is to appear left justified, right justified or centered.	"left"
labelColor	string	Color of the label text	"black"
labelFont	string	The font to use for the label.[1]	None
labelPosition	string	Specifies whether the label text is to appear above, below, or in the center of the marker line	"above"
labelText	string	Text to appear with marker line	“”
lineColor	string	Color of the marker line	"black"
lineStyle	string	Style of the marker line	"solid"

Field	Type	Description	Default
lineWidth	floating point	Width of the marker line	1.0
value	string	Location of marker on axis (e.g. "712.5" or "23Aug2003 1015")	"0"

[1] Fonts are specified as name[,style[,size]] where style is Plain, Bold, Italic, or Bold Italic (e.g. "Arial,BoldItalic,12", "Lucida Console,Plain,10").

The script in Example 26 reads a data set from a DSS file, plots that data set, sets the Viewport's background to light gray and adds a marker line on the Y axis.

Example 26: Viewport Class

```
from hec.script import *                                # for Plot class
from hec.script.Constants import TRUE, FALSE
from hec.heclib.dss import *                            # for DSS class
theFile = HecDss.open(“myFile.dss”)                     # open myFile.dss
thePath = “/BASIN/LOC/FLOW/01NOV2002/1HOUR/OBS/"
flowDataSet = theFile.read(thePath)                     # read a path name
thePlot = Plot.newPlot()                                # create a new Plot
thePlot.addData(flowDataSet)                            # add the flow data
viewport0=thePlot.getViewport(0)                        # get the first Viewport
viewport0.setBackground(“lightgray”)                    # set the Viewport’s bg
viewport0.setBackgroundVisible(TRUE)                    # tell Viewport to draw bg
marker = AxisMarker()                                   # create a new marker
marker.axis = “Y”                                       # set the axis
marker.value = “20000”                                  # set the value
marker.labelText = “Damaging Flow”                      # set the text
marker.labelColor = “red”                               # set the text color
marker.lineColor = “red”                                # set the line color
marker.fillColor = “red”                                # set the fill color
marker.fillType = “above”                               # set the fill type
maker.fillPattern = “diagonal cross”                    # set the fill pattern
viewport0.addAxisMarker(marker)                         # add the marker to the
                                                        # viewport
```

8.10.8 Axis Class

Axis methods are described in Table 8.21.

Table 8.21 Axis Methods

Method	Returns	Description
getLabel()	string	Return the Axis label
getMajorTic()	floating-point	Return the major tic interval for this Axis
getMinorTic()	floating-point	Return the minor tic interval for this Axis
getNumTicLabelLevels()	integer	Return the number of tic label levels for this Axis

Method	Returns	Description
getScaledLabel()	String	Return the label with scientific notation
getScaleMax()	floating-point	Return the maximum value for this Axis
getScaleMin()	floating-point	Return the minimum value for this Axis
getTicColor()	Color	Return the tic color
getTicColorString()	String	Return the Tic color as a String
getTicTextColor()	Color	Return the tic text color
getTicTextColorString()	String	Return the tic text color as a String
getViewMax()	floating-point	Return the maximum value for the (possibly) zoomed view for this Axis
getViewMin()	floating-point	Return the minimum value for the (possibly) zoomed view for this Axis
isComputingMajorTics()	Boolean	Return if major tics are to be computed
isComputingMinorTics()	Boolean	Return if minor tics are to be computed
isReversed()	Boolean	Returns whether the Axis is reversed.[1]
setComputeMajorTics(Boolean state)	None	Set whether to compute major tics
setComputeMinorTics(Boolean state)	None	Set whether to compute minor tics
setLabel(string label)	None	Set the label of this Axis
setLinear()	None	
setLogarithmic	None	
setMajorTicInterval(floating-point interval)	None	Set the major tic interval for this Axis to interval
setMinorTicInterval(floating-point interval)	None	Set the minor tic interval for this Axis to interval
setNumTicLabelLevels(integer layers)	None	Set the maximum number of tic label layers to specified number. -1 is unrestricted. Used mostly for time series axis.
setReversed(Boolean state)	None	Set the reversed state of the Axis.[1]
setScaleLimits(floating-point min, floating-point max)	None	Sets the minimum and maximum values for the axis (range of un-zoomed view)
setTicColor(String colorString)	None	Set the tic color to the color represented by colorString

Method	Returns	Description
setTicTextColor(String colorString)	None	Set the tic text color to the color represented by colorString
setViewLimits(floating-point min, floating-poin max)	None	Zooms based on world coordinates
unZoom()	None	Returns the view to the full axis range.
zoomByFactor(floating-point factor)	None	Change the zoom scaling by the given factor

[1] The coordinate system used is a graphics coordinate system with the origin (0,0) located at the top left corner of the display, with X increasing to the right and Y increasing downward. The reversed state is in respect to this coordinate system (i.e. Y is reversed if it increases upward).

The script in Example 27 reads a data set from a DSS file, adds that data set to a new Plot, and zooms in on the Y Axis.

Example 27: Using Axis Objects

```
from hec.script import *                                    # for Plot class
from hec.heclib.dss import *                                # for DSS class
thePlot = Plot.newPlot()                                    # create a Plot
dssFile = HecDss.open(“C:/mydb.dss”)     # open the DSS file
flow = dssFile.get(“/BASIN/LOC/FLOW/01NOV2002/1HOUR/OBS/”)
                                                            # read a data set
thePlot.addData(flow)                                       # add the data set
thePlot.showPlot()                                          # show the plot
viewport0 = thePlot.getViewport(0)                          # get the first Viewport
yaxis = viewport0.getAxis(“Y1”)                             # get the Y1 axis
yaxis.setScaleLimits(0., 25000.)                            # set the scale
yaxis.zoomByFactor(.”5”)                                    # zoom in
```

8.10.9 AxisTics Class

AxisTics methods are described in Table 8.22.

The script in Example 28 creates a new Plot with a data set read from DSS and tells the data set's axis tics to draw its minor tic marks.

Table 8.22 AxisTics Methods

Method	Returns	Description
areMajorTicLabelsVisible()	Boolean	Return whether the major tic labels are visible.
areMajorTicsVisible()	Boolean	Return whether the major tics are visible
areMinorTicLabelsVisible()	Boolean	Return whether the minor tic labels are visible
areMinorTicsVisible()	Boolean	Return whether the minor tics are visible

Method	Returns	Description
computeRatingFromOppositeAxis()	None	When used on the right (Y2) AxisTics object, with related curves on the Y1 and Y2 axes (e.g. stage and flow, or elevation and storage), causes the AxisTics to behave in a non-linear fashion such that Y1 and Y2 curves are coincident.
editProperties()	None	Display the Edit Properties Dialog for the AxisTics
getAxis()	Axis	Returns a reference to the axis that this object draws
getAxisTicColor()	Color	Return the tic color
getAxisTicColorString()	String	Return the tic color as a String
getFontSizes()	tuple of 3 integers	Return the regular, tiny, min and max font sizes for this AxisTics
getMajorTicLength()	Integer	Return the major tic length
getMinorTicLength()	integer	Return the minor tic length
setAxisTicColor(string colorString)	None	Set the tic color to the color represented by colorString
setFontSizes(integer sz,integer tiny, integer min, intege max)	None	Set the regular, tiny, min and max font sizes for this AxisTics
setMajorTicLabelsVisible(Boolean state)	None	Set the visibility of the major tic labels
setMajorTicLength(int ticLength)	None	Set the major tic length
setMajorTicsVisible(Boolean state)	None	Set the visibility of the major tics
setMinorTicLabelsVisible(Boolean state)	None	Set the visibility of the minor tic labels.
setMinorTicLength(int ticLength)	None	Set the minor tic length
setMinorTicsVisible(Boolean state)	None	Set the visibility of the minor tics

Example 28: Using AxisTics Objects

```
from hec.script import *                        # for Plot class
from hec.script.Constants import TRUE, FALSE
from hec.heclib.dss import *                    # for DSS class
thePlot = Plot.newPlot()                        # create a Plot
dssFile = HecDss.open("C:/mydb.dss")            # open the DSS file
flow = dssFile.get("/BASIN/LOC/FLOW/01NOV2002/1HOUR/OBS/")
                                                # read a data set
thePlot.addData(flow)                           # add the data set
thePlot.showPlot()                              # show the plot
viewport0 = thePlot.getViewport(flow)           # get the viewport for the #flow data set
yAxisTics = viewport0.getAxisTics("Y1")         # get the axis tics for the #Viewport
yAxisTics.setMinorTicsVisible(TRUE)             # tell axis tics to show tics
```

8.10.10 G2dLine Class

G2dLine methods are described in Table 8.23.

Table 8.23 G2dLine Methods

Method	Returns	Description
areSymbolsAutoInterval()	Boolean	Return whether the symbols for this line are placed at program-decided intervals
areSymbolsVisible()	Boolean	Return whether this line draws its symbols
editLineProperties()	None	Method that allows the editing of line properties. This method displays a visible dialog for line editing.
getFillColor()	Color	Return the fill color for this line
getFillColorString()	string	Return the fill color for this line as a String
getFillPatternString()	string	Return the fill pattern for this line as a String
getFillTypeString()	string	Return the Fill type for this line as a String.
getFirstSymbolOffset()	integer	Return the offset for the first symbol for this line
getLineColor()	Color	Return the line color for this line
getLineColorString()	string	Return the line color for this line as a String
getLineStepStyleString()	string	Return the line step style for this line as a String
getLineStyleString()	string	Return the line style for this line as a string
getLineWidth()	floating-point	Return the Line Width of the line
getNumPoints()	integer	Returns the Number of Points that this line has
getSymbolFillColor()	Color	Return the symbol fill color for this line’s symbols
getSymbolFillColorString()	string	Return the symbol fill color for this line’s symbols as a String
getSymbolInterval()	integer	Return the interval of data points (>0) on which symbols are drawn.
getSymbolLineColor()	Color	Return the symbol line color for this line’s symbols
getSymbolLineColorString()	string	Return the symbol line color for this line’s symbols as a String
getSymbolSize()	floating-point	Return the symbol size for this line

Method	Returns	Description
getSymbolSkipCountl()	integer	Return the number of points skipped between symbols (same as getSymbolInterval() – 1)
getSymbolTypeString()	string	Return the symbol type for this line as a string
isLineVisible()	Boolean	Return this line is drawn
setFillColor(string fillColor)	None	Set the fill color for this line
setFillPattern(string fillPattern)	None	Set the fill pattern for this line
setFillType(string fillType)	None	Set the Fill type for this line
setFirstSymbolOffset(integer offset)	None	Set the offset for first symbol for this line
setLineColor(string lineColor)	None	Set the line color for this line
setLineStepStyle(string stepStyle)	None	Set the line step style for this line
setLineStyle(string style)	None	Set the line style for this line
setLineVisible(Boolean state)	None	Set whether to draw this line
setLineWidth(floating-point width)	None	Set the width for this line
setSymbolFillColor(string symbolFillColor)	None	Set the symbol fill color for this line’s symbols
setSymbolInterval(integer interval)	None	Set the interval of data points (>0) on which symbols are drawn.
setSymbolLineColor(string symbolLineColor)	None	Set the symbol line color for this line’s symbols
setSymbolsAutoInterval(Boolean state)	None	Set whether to have the program decide the interval at which to draw symbols
setSymbolSize(floating-point size)	None	Set the symbol size for this line
setSymbolSkipCount(integer count)	None	Set the number of points skipped between symbols.
setSymbolsVisible(Boolean state)	None	Set whether to draw the symbols for this line
setSymbolType(string symbolType)	None	Set the symbol type for this line

The script in Example 29 creates a plot with a data set read from DSS, the script then tells that data set's curve to draw its symbols auto skipped.

Example 29: Using G2dLine Objects

```
from hec.script import *                                # for Plot class
from hec.script.Constants import TRUE, FALSE
from hec.heclib.dss import *                            # for DSS class
thePlot = Plot.newPlot()                                # create a Plot
dssFile = HecDss.open(“C:/mydb.dss”)                    # open the DSS file
flow = dssFile.get(“/BASIN/LOC/FLOW/01NOV2002/1HOUR/OBS/”)
                                                           # read a data set
thePlot.addData(flow)                                      # add the data set
thePlot.showPlot()                                      # show the plot
stageCurve = thePlot.getCurve(stage)                    # get the stage curve
stageCurve.setSymbolsAutoInterval(TRUE)                 # turn on symbols auto skip
```

8.10.11 G2dLabel, G2dTitle, and AxisLabel Classes

G2dLabel, G2dTitle and AxisLabel methods are described in Table 8.24.

Table 8.24 Label Methods

Method	Returns	Description
editProperties()	None	Display the Edit Properties Dialog for the label
getAlignmentString()	string	Return the text alignment for this label as a String
getBackground()	Color	Return the background color for the label[1]
getBackgroundString()	string	Return the background color for the label as a String[1]
getBorderStyleString()	string	Return the border style for this label as a string
getBorderWeight()	floating-point	Return the border weight for this label
getFillColor()	Color	Return the fill color for this label as a Color[1]
getFillColorString()	string	Return the fill color for this label as a string[1]
getFillPatternString()	string	Return the fill pattern for this label as a string[1]
getFontFamily()	string	Return the font family for the label
getFontSize()	integer	Return the font size for the label
getFontSizes()	tuple of 3 integers	Return the regular, tiny, min and max font sizes for this label
getFontString()	string	Return the font for the label as a string[2].
getFontStyleString()	string	Return the font style for the label as a String
getForeground()	Color	Return the foreground color for the label
getForegroundString()	string	Return the foreground color for the label as a String
getIcon()	Icon	Return the Icon to display for this label
getIconPath()	string	Return the Icon path to display for this label
getRotation()	integer	Return the text rotation for this label
getSpacing()	integer	Return the spacing around this label
getText()	string	Return the text for the label
isBackgroundVisible()	Boolean	Return whether the background is visible
isBorderVisible()	Boolean	Return whether the border is visible
setAlignment(string alignment)	None	Set the text alignment for this label

Method	Returns	Description
setBackground(string colorString)	None	Set the background color for the label[1]
setBackgroundVisible(Boolean state)	None	Set the background visibility for the label
setBorderColor(string colorString)	None	Set the border color for this label
setBorderStyle(string style)	None	Set the border style for this label
setBorderVisible(Boolean state)	None	Set the border visibility for this label
setBorderWeight(floating-point weight)	None	Set the border weight for this label
setFillColor(string color)	None	Set the fill color for this label[1]
setFillPattern(string pattern)	None	Set the fill pattern for this label[1]
setFont(string font)	None	Set the font for the label[2].
setFontFamily(string fam)	None	Set the font family for the label
setFontSize(integer sz)	None	Set the font size for the label
setFontSizes(integer sz, integer tiny, integer min, integer max)	None	Set the regular, tiny, min and max font sizes for this label
setFontStyle(string style)	None	Set the font style for the label
setForeground(string colorString)	None	Set the foreground color for the label
SetIcon(Icon icon)	None	Set the Icon to display for this label
SetIcon(string iconPath)	None	Set the Icon to display for this label
setRotation(integer rotation)	None	Set the text rotation for this label
setSpacing(integer space)	None	Set the spacing around this label
SetText(string text)	None	Set the text for the label

[1] In the current version, fill color and background color are synonymous (e.g. fills are performed with the background color). Future version may support separate fill and background colors.

[2] Fonts are specified as name[,style[,size]] where style is Plain, Bold, Italic, or Bold Italic (e.g. “Arial,BoldItalic,12”, “Lucida Console,Plain,10”).

The script in Example 30 creates a plot from a DSS data set and sets the Y1 axis label text to blue.

Example 30: Using AxisLabel Objects

```
from hec.script import *                              # for Plot class
from hec.heclib.dss import *                            # for DSS class
thePlot = Plot.newPlot()                                # create a Plot
dssFile = HecDss.open(“C:/mydb.dss”)                    # open the DSS file
flow = dssFile.get(“/BASIN/LOC/FLOW/01NOV2002/1HOUR/OBS/”)
                                                        # read a data set
thePlot.addData(flow)                                   # add the data set
thePlot.showPlot()                                    # show the plot
viewport0 = thePlot.getViewport(0)                    # get the first viewport
yaxislabel = viewport0.getAxisLabel("Y1")             # get the Y1 axis label
yaxislabel.setForeground(“blue”)                      # set the Y1 axis label text to blue
```

8.10.12 Templates

Template files saved interactively from HEC-DSSVue may be applied to plots via scripting. When saving a template interactively from the plot window via the "Save Template…" entry on the "File" menu, HEC-DSSVue:

1. Chooses the "My Documents" subdirectory of the directory specified in the USERPROFILE environment variable as the default location for the template file.
2. Appends “.template” to the end of the specified file name.

The *applyTemplate(string filename)* G2dDialog method requires the actual file name for the template file. To apply a template saved in the default directory, the complete template file name must be re-created as demonstrated in Example 31.

Example 31: Applying Template Saved in Default Directory

```
import os                                                   # for getenv() & sep
from hec.script import *                                    # for Plot class
from hec.heclib.dss import *                                # for DSS class
thePlot = Plot.newPlot()                                    # create a Plot
dssFile = HecDss.open(“C:/mydb.dss”)                        # open the DSS file
flow = dssFile.get(“/BASIN/LOC/FLOW/01NOV2002/1HOUR/OBS/”)
                                                            # read a data set
thePlot.addData(flow)                                       # add the data set
thePlot.showPlot()                                          # show the plot
templateName = “myTemplate”                                 # template base name
templateFileName =                      \                   # re-create the file name
  os.getenv(“userprofile”)              \
  + os.sep                              \
  + “My Documents”                      \
  + os.sep                              \
  + templateName                        \
  + “.template”
thePlot.applyTemplate(templateFileName)                     # apply the template
```

8.11 Plot Component Properties

The following tables are the valid values to be used when calling plot related functions that take a color (setBackground(string color), etc…), an alignment (setAlignment()), a rotation (setRotation()), a fill pattern (setFillPattern()), a fill type (setFillType()), a line style (setLineStyle()), a step style (setLineStepStyle()).or a symbol type (setSymbolType()).

8.11.1 Colors

Colors can be specified either by a *String* or by a *java.awt.Color* object. If setting a color through the use of a *String* object the String can either be a standard color name (i.e. "darkred") or an RGB string (i.e. "255,20,20"). Standard color names are listed in Table 8.25.

Table 8.25 Standard Colors

black	darkmagenta	green	lightorange	orange
blue	darkorange	lightblue	lightpink	pink
cyan	darkpink	lightcyan	lightpurple	purple
darkblue	darkpurple	lightgray	lightred	red
darkcyan	darkred	lightgreen	lightyellow	white
darkgray	darkyellow	lightmagenta	magenta	yellow
darkgreen	gray			

8.11.2 Alignment

Supported text alignments are: Left, Center, and Right.

8.11.3 Positions

Supported text positions are: Above, Center, and Below.

8.11.4 Rotation

Supported text rotation values are: 0, 90, 180, 270.

8.11.5 Fill Patterns

Supported fill patterns are listed inTable 8.26.

Table 8.26 Fill Patterns

Solid	Horizontal	Vertical
Cross	FDiagonal	BDiagonal
Diagonal Cross	None	

8.11.6 Fill Types

Supported fill types are: None, Above, and Below.

8.11.7 Line Styles

Supported line style values are listed in Table 8.27.

Table 8.27 Line Styles

Solid	Dash	Dot
Dash Dot	Dash Dot-Dot	

8.11.8 Step Style

Supported step style values are: Normal, Step, and Cubic.

8.11.9 Symbol Types

Supported symbol type values are listed in Table 8.28.

Table 8.28 Symbol Types

Asterisk	Backslash	Backslash Square
Circle	Diamond	Forwardshlash
Forwardslash Square	Hash	Hash Diamond
Hash Square	Hash Triangle	Hash Triangle2
Hourglass	Open Circle	Open Diamond
Open Hourglass	Open Square	Open Triangle
Open Triangle2	Pipe	Pipe Diamond
Pipe Square	Plus	Plus Circle
Plus Diamond	Plus Square	Square
Triangle	Triangle2	X
X Circle	X Square	X Triangle
X Triangle2		

8.12 Tables

Tables allow you to view data in a vertical scrolling window that shows the ordinates, the dates and times and the values for the selected data sets.

8.12.1 Tabulate Class

```
Tabulate.newTable()
Tabulate.newTable(string title)
```

The Tabulate class in the *hec.script* module is used to create a new Table dialog. It contains two functions to create a Table dialog, each of which returns as a HecDataTableFrame object.

Creation of a table is illustrated in Example 32.

Example 32: Creating a Table

```
from hec.script import *
myTable = Tabulate.newTable()
or
from hec.script import *
myTable = Tabulate.newTable(“Elevation vs Flow”)
```

8.12.2 HecDataTableFrame Class

HecDataTableFrame methods are described in Table 8.29.

Table 8.29 HecDataTableFrame Methods

Method	Returns	Description
addData(DataContainer dc)	integer	Adds Data Set to the table.
close()	None	Closes the table
export()	None	Brings up the Table Export Options dialog.
export(string fileName, TableExportOptions options)	None	Exports table to specified file with specified options
exportAsHTML(string fileName)	None	Exports table in HTML format to specified file with no title and elements indented with tabs
exportAsHTML(string fileName, string title, string indent)	None	Exports table in HTML format to specified file with specified title and indentation string
exportAsXML(string fileName)	None	Exports table in XML format to specified file with no title and elements indented with tabs
exportAsXML(string fileName, string title, string indent)	None	Exports table in XML format to specified file with specified title and indentation string
getCellBackground(integer row, integer column)	Color	Returns the background color of the specified cell as a Color
getCellBackgroundString(integer row, integer column)	string	Returns the background color of the specified cell as a string
getCellForeground(integer row, integer column)	Color	Returns the foreground color of the specified cell as a Color
getCellForegroundString(integer row, integer column)	string	Returns the foreground color of the specified cell as a string

Method	Returns	Description
getColumn(DataContainer dc)	integer	Returns the number of the column that contains the specified data, if the parameter is time series data, or the number of the column that contains the x-ordinates if the parameter is paired data
getColumn(string header)	integer	Returns the number of the column that has the specified header text. Line breaks in the header text are specified as "\n"
getColumnBackground(integer column)	Color	Returns the background color of the specified column as a Color
getColumnBackgroundString(integer column)	string	Returns the background color of the specified column as a string
getColumnForeground(integer column)	Color	Returns the foreground color of the specified column as a Color
getColumnForegroundString(integer column)	string	Returns the foreground color of the specified column as a string
getColumnHeaderBackgroung(integer column)	Color	Returns the background color of the header of the specified column as a Color
getColumnHeaderBackgroungString(integer column)	string	Returns the background color of the header of the specified column as a string
getColumnHeaderFontString(integer column)	string	Returns the font of the header of the specified column as a string.[1]
getColumnHeaderForegroung(integer column)	Color	Returns the foreground color of the header of the specified column as a Color
getColumnHeaderForegroungString(integer column)	string	Returns the foreground color of the header of the specified column as a string
getColumnLabel(integer colNum)	string	Returns the column header text for the specified column
getColumnLabels()	list of strings	Returns the column header text for all columns
getColumnWidth(integer colNum)	integer	Returns the width of the specified column in pixels
getColumnWidths()	list of integers	Returns a list of all the column widths in pixels
getCommasState()	Boolean	Get whether the commas are shown
getDateTimeAsTwoColumnsState()	Boolean	Get whether date/time columns are shown as 1 or 2 columns in the table

Method	Returns	Description
getExportString(TableExportOptions options)	string	Returns a string representation of the table exported according to the specified options
getHeight()	integer	Return the height of the table in screen coordinates.
getHTMLExportString()	string	Returns a string representation of the table exported in HTML format with no title and elements indented with tabs
getHTMLExportString(string title, string indent)	string	Returns a string representation of the table exported in HTML format with the specified title and indentation string
getLocation()	Point	Returns the location of the table in screen coordinates[2].
getRowBackground(integer row)	Color	Returns the background color of the specified row as a Color
getRowBackgroundString(integer row)	string	Returns the row background color of the specified column as a string
getRowForeground(integer row)	Color	Returns the foreground color of the specified row as a Color
getRowForegroundString(integer row)	string	Returns the row foreground color of the specified column as a string
getSize()	Dimension	Return the dimensions of the table in screen coordinates.
getTableTitle()	G2dTitle	Returns the title of the HecDataTableFrame object as a G2dTitle object.
getTableTitleText()	string	Returns the title of the HecDataTableFrame object as a string.
getWidth()	integer	Return the width of the table in screen coordinates.
GetXMLExportString()	string	Returns a string representation of the table exported in XML format with no title and elements indented with tabs
getXMLExportString(string title, string indent)	string	Returns a string representation of the table exported in XML format with the specified title and indentation string
hide()	None	Hide the table
iconify()	None	Minimize (iconify) the table
maximize()	None	Maximize the table
minimize()	None	Minimize (iconify) the table

Method	Returns	Description
newTable()	HecDataTableFrame	Static, same as Tabulate.newTable()
newTable(string title)	HecDataTableFrame	Static, same as Tabulate.newTable(title)
print()	None	Display the print table
restore()	None	Restore the table from a minimized or maximized state
setCellBackgound(integer row, integer column, string color)	None	Sets the background color of the specified cell to the specified color
setCellForeground(integer row, integer column, string color)	None	Sets the foreground color of the specified cell to the specified color
setColumnBackgound(integer column, string color)	None	Sets the background color of the specified column to the specified color
setColumnForeground(integer column, string color)	None	Sets the foreground color of the specified column to the specified color
setColumnHeaderBackgound(integer column, string color)	None	Sets the background color of the header of the specified column to the specified color
setColumnHeaderFont(integer column, string font)	None	Sets the font of the header of the specified column to the specified font.[1]
setColumnHeaderForeground(integer column, string color)	None	Sets the foreground color of the header of the specified column to the specified color.
setColumnLabel(integer column, string label)	None	Sets the column header text of the specified column to the specified label.
setColumnLabels(list labels)	None	Sets the column header text of all columns to the labels specified in the list of strings
setColumnPrecision(integer colNum, integer precision)	None	Sets the number of decimal places to display for the specified column
setColumnWidth(integer colNum, integer width)	None	Sets the width of the specified column in pixels
setColumnWidths(list widths)	None	Sets the width in pixels of all the columns to those specified in the parameter (list of integers)
setCommasState(Boolean showCommas)	None	Set state to show commas or not
setDateTimeAsTwoColumnsState(integer showDateTimeAs2Columns)	None	Set whether date/time columns should show as 1 or 2 columns in the table

Method	Returns	Description
setLocation(integer x, integer y)	None	Sets the location of the table in screen coordinates[2].
setSize(int width, int height)	None	Sets the size of the table in screen coordinates.
setRowBackgound(integer row, string color)	None	Sets the background color of the specified row to the specified color
setRowForeground(integer row, string color)	None	Sets the foreground color of the specified row to the specified color
setTableTitleText(string title)	None	Sets the title of the HecDataTableFrame object.
showTable()	None	Show the table

[1] Fonts are specified as *name*[,*style*[,*size*]] where *style* is Plain, Bold, Italic, or Bold Italic (e.g. "Arial,BoldItalic,12", "Lucida Console,Plain,10").

[2] The coordinate system used is a graphics coordinate system, where X increases to the right and Y increases downward from the origin (0,0) which is located in the top left corner of the display. Locations set or retrieved refer to the top left corner of the plot in reference to this coordinate system.

8.13 TableExportOptions Class

TableExportOptions objects hold complete descriptions of options for exporting tables to fixed-column-width or column-delimited text files. **TableExportOptions** objects have fields that are settable by the user to create marker lines of various styles. New **TableExportOptions** objects are created by calls to TableExportOptions() (e.g., *myOptions = TableExportOptions()*). **TableExportOptions** fields are described in Table 8.30.

Table 8.30 TableExportOptions Fields

Field	Type	Description	Default
delimiter	string (one character)	Placed between fields if fixedWidthCol is False	'\t' (tabcharacter)
quotedStrings	Boolean	Specifies whether to enclose text in quotes	False
title	string	Title of the table	None
fixedWidthCols	Boolean	Specifies whether fields are exported as fixed-width columns or fields separated by delimiter	False
columnHeader	Boolean	Specifies whether the column headers are to be exported	True
rowHeader	Boolean	Specifies whether the row headers are to be exported	False
gridLines	Boolean	Specifies whether the table will be exported with text "lines" between the rows and columns	False

The script in Example 33 creates a table from two DSS data sets and exports the table as a comma-separated-value text file.

Example 33: Filling, Displaying and Exporting a Table

```
from hec.heclib.dss import *                        # for DSS
from hec.script import *                            # for Tabulate
file = “C:/mydb.dss"                                # specify the DSS file
dssfile = HecDss.open(file)                         # open the file
# read 2 records
stage = dssfile.get("//AXEMA/STAGE/01OCT2001/1HOUR/OBS/")
flow = dssfile.get("//AXEMA/FLOW/01OCT2001/1HOUR/OBS/")
theTable = Tabulate.newTable()                      # create the table
theTable.setTitle("Test Table")                     # set the table title
theTable.setLocation(5000,5000)                     # set the location of the table
off
                                                    the screen
theTable.addData(flow)                              # add the data
theTable.addData(stage)
theTable.showTable()                                # show the table
flowCol = theTable.getColumn(flow)                  # adjust columns
stageCol = theTable.getColumn(stage)
flowWidth = theTable.getColumnWidth(flow)
stageWidth = theTable.getColumnWidth(stage)
theTable.setColumnPrecision(flowCol, 0)
theTable.setColumnPrecision(stageCol, 2)
theTable.setColumnWidth(flowCol, flowWidth - 10)
theTable.setColumnWidth(stageCol, stageWidth + 10)
# get new export options
# delimit with commas
# set the title
# set the output file name
# export to the file
# close
```

8.14 HecMath Class

The objects returned from the *HecDss*..read(…) methods and supplied to the *HecDss*.write(…) method are HecMath objects. There are currently three types of HecMath classes: TimeSeriesMath class, PairedDataMath class, and StreamRatingMath class which represent time-series data, paired data, and stream rating data, respectively.

An HecMath object can be created for writing to new DSS data by first creating a new DataContainer as discussed in Sections 8.8.1 (TimeSeriesContainer Class) and 8.8.2 (), and then calling the HecMath.createInstance() function with the DataContainer object as the only parameter (e.g. myTimeSeriesMath = HecMath.createInstance(myTimeSeriesContainer)). The methods for HecMath Objects, which are described in Section 8.14, are listed inTable 8.31.

Table 8.31 HecMath Methods

Method	Returns	Description Section
abs()	HecMath	8.15.1
accumulation()	TimeSeriesMath	8.15.2
acos()	HecMath	8.15.3
add(floating-point constant)	HecMath	8.15.4
add(TimeSeriesMath dataset)	TimeSeriesMath	8.15.5
applyMultipleLinearRegression(string startTimeString, string endTimeString, sequence datasets, floating-point minLimit, floating-point maxLimit)	TimeSeriesMath	8.15.6
asin()	HecMath	8.15.7
atan()	HecMath	8.15.8
ceil()	HecMath	8.15.9
centeredMovingAverage(integer number, Boolean onlyValid, Boolean useReduced)	TimeSeriesMath	8.15.10
conicInterpolation(TimeSeriesMath dataset, string inputType, string outputType, floating-point storageFactor)	TimeSeriesMath	8.15.11
convertToEnglishUnits()	HecMath	8.15.12
convertToMetricUnits()	HecMath	8.15.13
correlationCoefficients(TimeSeriesMath dataset)	LinearRegression Statistics	8.15.14
cos()	HecMath	8.15.15
cyclicAnalysis()	sequence of TimeSeriesMath	8.15.16
decayingBasinWetnessParameter(TimeSeries Math tsPrecip, floating-point decayRate)	TimeSeriesMath	8.15.17
divide(floating-point constant)	HecMath	8.15.18
divide(TimeSeriesMath dataset)	TimeSeriesMath	8.15.19
estimateForMissingPrecipValues(integer maxMissing)	TimeSeriesMath	8.15.20
estimateForMissingValues(integer maxMissing)	TimeSeriesMath	8.15.21
exp()	HecMath	8.15.22
exponentiation(floating-point constant)	HecMath	8.15.23
exponentiation(HecMath tsMath)	HecMath	8.15.24
extractTimeSeriesDataForTimeSpecification(string timeLevel, string range, Boolean isInclusive, integer intervalWindow, Boolean setAsIrregular)	TimeSeriesMath	8.15.25
firstValidDate()	integer	8.15.26
firstValidValue()	floating-point	8.15.27
floor()	HecMath	8.15.28

Method	Returns	Description Section
flowAccumulatorGageProcessor(TimeSeries Math dataset)	TimeSeriesMath	8.15.29
fmod(HecMath tsMath)	HecMath	8.15.30
forwardMovingAverage(integer number)	TimeSeriesMath	8.15.31
forwardMovingAverage(integer numberToAverageOver, Boolean OnlyValidValues, Boolean useReduced)	Hecmath	8.15.32
generateDataPairs(TimeSeriesMath dataset, Boolean sort)	PairedDataMath	8.15.33
generateRegularIntervalTimeSeries(string startTime, string endTime, string timeInterval, string timeOffset, floating-point initialValue)	TimeSeriesMath	8.15.34
getData()	DataContainer	8.15.35
getType()	string	8.15.36
getUnits()	string	8.15.37
gmean(list of HecMath tsMathArray)	HecMath	8.15.38
hmean(list of HecMath tsMathArray)	HecMath	8.15.39
integerDivide(floating-point constant)	HecMath	8.15.40
integerDivide(HecMath tsMath)	HecMath	8.15.41
interpolateDataAtRegularInterval(string timeInterval, string timeOffset)	TimeSeriesMath	8.15.42
inverse()	HecMath	8.15.43
isEnglish()	Boolean	8.15.44
isMetric()	Boolean	8.15.45
isMuskingumRoutingStable(integer subReachCount, floating-point muskingumK, floating-point muskingumX)	string	8.15.46
lastValidDate()	integer	8.15.47
lastValidValue()	floating-point	8.15.48
log()	HecMath	8.15.49
log10()	HecMath	8.15.50
max()	floating-point	8.15.51
max(list of HecMath tsMathArray)	HecMath	8.15.52
maxDate()	integer	8.15.53
mean()	floating-point	8.15.54
mean(list of HecMath tsMathArray)	HecMath	8.15.55
med(list of HecMath tsMathArray)	HecMath	8.15.56
mergePairedData(PairedDataMath dataset)	PairedDataMath	8.15.57
mergeTimeSeries(TimeSeriesMath dataset)	TimeSeriesMath	8.15.58
min()	floating-point	8.15.59
min(list of HecMath tsMathArray)	HecMath	8.15.60
minDate()	integer	8.15.61

Method	Returns	Description Section
modifiedPulsRouting(TimeSeriesMath dataset, integer subReachCount, floating-point muskingumX)	TimeSeriesMath	8.15.62
modulo(floating-point constant)	HecMath	8.15.63
modulo(HecMath tsMath)	HecMath	8.15.64
multipleLinearRegression(sequence datasets, floating-point minLimit, floating-point maxLimit)	PairedDataMath	8.15.65
multiply(floating-point constant)	HecMath	8.15.66
multiply(TimeSeriesMath dataset)	TimeSeriesMath	8.15.67
muskingumRouting(integer subReachCount, floating-point muskingumK, floating-point muskingumX)	TimeSeriesMath	8.15.68
neg()	HecMath	8.15.69
numberInvalidValues()	integer	8.15.70
numberMissingValues()	integer	8.15.71
numberQuestionedValues()	integer	8.15.72
numberRejectedValues()	integer	8.15.73
numberValidValues()	integer	8.15.74
olympicSmoothing(integer number, Boolean onlyValid, Boolean useReduced)	TimeSeriesMath	8.15.75
p1(list of HecMath tsMathArray)	HecMath	8.15.76
p2(list of HecMath tsMathArray)	HecMath	8.15.77
p5(list of HecMath tsMathArray)	HecMath	8.15.78
p10(list of HecMath tsMathArray)	HecMath	8.15.79
p20(list of HecMath tsMathArray)	HecMath	8.15.80
p25(list of HecMath tsMathArray)	HecMath	8.15.81
p75(list of HecMath tsMathArray)	HecMath	8.15.82
p80(list of HecMath tsMathArray)	HecMath	8.15.83
p89(list of HecMath tsMathArray)	HecMath	8.15.84
p90(list of HecMath tsMathArray)	HecMath	8.15.85
p95(list of HecMath tsMathArray)	HecMath	8.15.86
p99(list of HecMath tsMathArray)	HecMath	8.15.87
periodConstants(TimeSeriesMath dataset)	TimeSeriesMath	8.15.88
polynomialTransformation(TimeSeriesMath dataset)	TimeSeriesMath	8.15.89
polynomialTransformationWithIntegral(TimeSeriesMath dataset)	TimeSeriesMath	8.15.90
product(list of HecMath tsMathArray)	HecMath	8.15.91

Method	Returns	Description Section
ratingTableInterpolation(TimeSeriesMath dataset)	TimeSeriesMath	8.15.92
replaceSpecificValues(HecDouble from, HecDouble to)	HecMath	8.15.93
reverseRatingTableInterpolation(TimeSeriesMath dataset)	TimeSeriesMath	8.15.94
rms(list of HecMath tsMathArray)	HecMath	8.15.95
round()	HecMath	8.15.96
roundOff(integer digits, integer place)	HecMath	8.15.97
screenWithConstantValue(String durationStr, floating-point rejectTolerance, floating-point questionTolerance, floating-point minThreshold, integer maxMissing)	HecMath	8.15.98
screenWithDurationMagnitude(String durationStr, floating-point minRejectLimit, floating-point minQuestionLimit, floating-point maxRejectLimit, floating-point maxQuestionLimit)	HecMath	8.15.99
screenWithForwardMovingAverage(integer numberToAverageOver, floating-point changeLimit)	HecMath	8.15.100
screenWithForwardMovingAverage(integer number, floating-point changeLimit, Boolean setMissing, string invalidQuality)	TimeSeriesMath	8.15.101
screenWithMaxMin(floating-point min, floating-point max, floating-point rate, Boolean setMissing, string invalidQuality)	TimeSeriesMath	8.15.102
screenWithMaxMinfloating-point minValueLimit, floating-point maxValueLimit, floating-point changeLimit)	HecMath	8.15.103
screenWithMaxMin(floating-point minValueLimit, floating-point maxValueLimit, floating-point changeLimit, Boolean setInvalidToSpecified, floating-point invalidValueReplacement, string qualityFlagForInvalidValue)	HecMath	8.15.104
screenWithMaxMin(floating-point minRejectLimit, floating-point minQuestionLimit, floating-point maxQuestionLimit, floating-point maxRejectLimit)	HecMath	8.15.105
screenWithRateOfChange(floating-point minRejectLimit, floating-point minQuestionLimit, floating-point maxQuestionLimit, floating-point maxRejectLimit)	HecMath	8.15.106
setCurve(string name)	None	8.15.107

Method	Returns	Description Section
setCurve(integer number)	None	8.15.108
setData(DataContainer data)	None	8.15.109
setLocation(string location)	None	8.15.110
setParameter(string parameter)	None	8.15.111
setPathname(string pathname)	None	8.15.112
setTimeInterval(string interval)	None	8.15.113
setType(string type)	None	8.15.114
setUnits(string units)	None	8.15.115
setVersion(string version)	None	8.15.116
setWatershed(string watershed)	None	8.15.117
shiftAdjustment(TimeSeriesMath dataset)	TimeSeriesMath	8.15.118
shiftInTime(string timeShift)	TimeSeriesMath	8.15.119
sign()	HecMath	8.15.120
sin()	HecMath	8.15.121
skewCoefficient()	floating-point	8.15.122
snapToRegularInterval(string timeInterval, string timeOffset, string timeBackward, string timeForward)	TimeSeriesMath	8.15.123
sqrt()	HecMath	8.15.124
standardDeviation()	floating-point	8.15.125
standardDeviation (list of HecMath tsMathArray)	HecMath	8.15.126
straddleStaggerRouting(integer avgCount, integer lagCount, integer subReachCount)	TimeSeriesMath	8.15.127
subtract(floating-point constant)	HecMath	8.15.128
subtract(TimeSeriesMath dataset)	TimeSeriesMath	8.15.129
successiveDifferences()	TimeSeriesMath	8.15.130
sum()	floating-point	8.15.131
sum (list of HecMath tsMathArray)	HecMath	8.15.132
tan()	HecMath	8.15.133
timeDerivative()	TimeSeriesMath	8.15.134
transformTimeSeries(string timeInterval, string timeOffset, string functionType)	TimeSeriesMath	8.15.135
transformTimeSeries(TimeSeriesMath dataset, string functionType)	TimeSeriesMath	8.15.136
truncate()	HecMath	8.15.137
twoVariableRatingTableInterpolation(TimeSeriesMath dataset1, TimeSeriesMath dataset2)	TimeSeriesMath	8.15.138
variance(list of HecMath tsMathArray)	HecMath	8.15.139

8.15 Math Functions

Math functions are accessible through the general class called **HecMath**. **HecMath** objects hold data sets and allow you to perform mathematical operations on them. They can also be passed to plots and tables to display the data. A **HecMath** object is either a **TimeSeriesMath** object or a **PairedDataMath** object, which handle time series and paired data sets, respectively.

Before using **PairedDataMath** methods, be sure to read the description for the *setCurve* method. Paired data sets may contain multiple curves. The *setCurve* method provides user control over which paired data curve is operated upon by subsequent function calls.

8.15.1 Absolute Value

`abs``()`

Derive a new time series or paired data set from the absolute value of values of the current data set. For time series data, missing values are kept as missing. For paired data sets, use the *setCurve* method to first select the paired data curve(s).

See also: `setCurve()`
Parameters: Takes no parameters
Example: `NewDataSet = dataSet.abs()`
Returns: A new **HecMath** object of the same type as the current object

8.15.2 Accumulation (Running)

`accumulation``()`

Derive a new time series by computing a running accumulation of the current time series.

For time points in which the current time series value are missing, the value in the accumulation time series remains constant (same as the accumulated value at the last valid point location).

Parameters: Takes no parameters
Example: `NewTimeSeries = timeSeries.accumulation`**`()`**
Returns: A new **TimeSeriesMath** object

8.15.3 Arccosine Trigonometric Function

`acos``()`

Derive a new time series or paired data set from the arccosine of values of the current data set. The resultant data set values are in radians. For time series data, missing values are kept as missing.

For paired data sets, use the *setCurve* (see Sections 8.15.07 and 8.15.08) function to first select the paired data curve (or all curves) to apply the function. By default the function is applied to all paired data curves.

See also: setCurve()
Example: `newDataSet = dataSet.acos()`
Parameters: Takes no parameters
Returns: A **HecMath** object of the same type as the current object

8.15.4 Add a Constant

`add``(floating-point constant)`

Add the value constant to all valid values in the current time series or paired data set. For time series data, missing values are kept as missing.

For paired data, constant is added to y-values only. Use the *setCurve* method to first select the paired data curve(s).

See also: add(HecMath dataSet)
setCurve()
Example: `newDataSet = dataSet.add(2.5)`
Parameters: `constant` - A floating-point value
Returns: A new **HecMath** object of the same type as the current object

8.15.5 Add a Data Set

`add``(TimeSeriesMath tsData)`

Add the values in the data set *tsData* to the values in the current data set. The function only applies to time series data sets.

When adding one time series data set to another, there is no restriction that times in the two data sets match exactly. However, only values with coincident times will be added. Times in the current time series data set that cannot be matched with times in the second data set are set to missing.

Values in the current data set that are missing are kept as missing. Either or both data sets may be regular or irregular interval time series. This function will not merge data sets. Use the *mergeTimeSeries* (for time series data sets) or the *mergePairedData* (for paired data sets) functions for this purpose.

See also: add(floating-point constant)
mergeTimeSeries(TimeSeriesMath)
mergePairedData(PairedDataMath)
Example: `newTsData = tsData.add(otherTsData)`
Parameters: `tsData` - A **TimeSeriesMath** object
Returns: A new **TimeSeriesMath** object

8.15.6 Apply Multiple Linear Regression Equation

```
applyMultipleLinearRegression(string startTimeString,
   string endTimeString,
   sequence tsDataSetSequence,
   floating-point minimumLimit,
   floating-point maximumLimit)
```

Apply the regression coefficients contained in the current paired data set to the array of time series data sets in **tsDataSetSequence** to develop a new time series data set. The *applyMultipleLinearRegression* function applies the multiple linear regression coefficients computed with the *multipleLinearRegression* function (see Section 8.15.65).

For the general linear regression equation, a dependent variable, Y, may be computed from a set independent variables, Xn:

$$Y = B0 + B1*X1 + B2*X2 + B3*X3$$

where Bn are linear regression coefficients.

For time series data sets, an estimate of the original time series data set values may be computed from a set of independent time series data sets using regression coefficients such that:

$$TsEstimate(t) = B0 + B1*TS1(t) + B2*TS2(t) + \ldots + Bn*TSn(t)$$

where Bn are the set of regression coefficients and TSn are the time series data sets contained in *tsDataSetSequence*.

The number of regression coefficients in the current **PairedDataMath** object must be one more than the number of independent time series data sets in *tsDataSetSequence*. The collection of selected time series data sets must be in the same order as when the regression coefficients were computed with the *multipleLinearRegression* method.

All the time series data sets must be regular interval and have the same time interval. The function filters the data to determine the time period common to all time series data sets and uses only those points in the regression analysis. For any given time, if a value is missing in any time series, the value in resultant time series is set to missing.

The parameters *minimumLimit* and *maximumLimit* can be used to specify the range of valid values for the resultant data set. Values which fall outside the specified range are set to missing. *minimumLimit* or *maximumLimi***t** may be entered as *Constants.UNDEFINED* to ignore the minimum or maximum value check.

If *startTimeString* or *endTimeString* are blank strings, the start and end time of the resultant time series will be defined by the time period common to all time series data sets in *tsDataSetSequence*. Otherwise the time series start and end may be defined using *startTimeString* and *endTimeStrin***g** which have the usual HEC time window format (e.g. "01JAN2001 1400").

Names, parameter type and unit labels for the new time series data set are copied over from the first time series data set in *tsDataSetSequence*. The F-part in the new data set is set to "COMPUTED".

Parameters:

`startTimeString` – A string containing an HEC time (e.g. "01JAN2001 1400") specifying the start time of the resultant time series data set. May be blank (" ")

`endTimeString` – A string containing an HEC time (e.g. "01JAN2001 1400") specifying the ending time of the resultant time series data set. May be blank (" ")

`tsDataSetSequence` – Sequence of **TimeSeriesMath** objects. Must all be regular interval and have the same time interval.

`minimumLimit` – A floating-point value specifying the minimum valid value in the resultant time series data set. Set to Constants.UNDEFINED to ignore this option.

`maximumLimit` – A floating-point value specifying the maximum valid value in the resultant time series data set. Set to Constants.UNDEFINED to ignore this option.

Example:

```
newTsData =
pairedData.applyMultipleLinearRegression(
“01Jan2000 0000”,
“31Dec2000 2300”,
(tsData1, tsData2, tsData3),
Constants.UNDEFINED,
Constants.UNDEFINED)
```

Returns: A new regular interval **TimeSeriesMath** object

Generated Exceptions: Throws a *HecMathException* if the number of data sets in *tsDataSetSequence* is not equal to the number of regression coefficients -1, or if the data sets in *tsDataSetSequence* are not regular interval time series data sets with the same interval time.

8.15.7 Arcsine Trigonometric Function

```
asin()
```

Derive a new time series or paired data set from the arcsine of values of the current data set. The resultant data set values are in radians. For time series data, missing values are kept as missing.

For paired data sets, use the *setCurve* (see Sections 8.15.107 and 8.15.108) function to first select the paired data curve (or all curves) to apply the function. By default the function is applied to all paired data curves.

See also: setCurve()
Example: `newDataSet = dataSet.asin()`
Parameters: Takes no parameters
Returns: A **HecMath** object of the same type as the current object

8.15.8 Arctangent Trigonometric Function

```
atan()
```

Derive a time series or paired data set computed from the arctangent of values of the current data set. For time series data, missing values are kept as missing. If the cosine of the current time series value is zero, the value is set missing.

For paired data sets, use the *setCurv*e method to first select the curve(s).

See also: setCurve()
Example: `newDataSet = dataSet.atan()`
Parameters: Takes no parameters
Returns: A new **HecMath** object of the same type as the current object

8.15.9 Ceiling Function

```
ceil()
```

Derive a time series or paired data set with values of the current time series rounded up to the nearest whole number that is greater to or equal to the value. For time series data, missing values are kept as missing.

For paired data sets, use the *setCurve* method to first select the curve(s).

See also: setCurve(), floor()
Example: `newDataSet = dataSet.ceil()`
Parameters: Takes no parameters
Returns: A new **HecMath** object of the same type as the current object

8.15.10 Centered Moving Average Smoothing

```
centeredMovingAverage(integer numberToAverageOver,
  boolean onlyValidValues,
  boolean useReduced)
```

Derive a new time series from the centered moving average of *numberToAverageOver* values in the current time series. numberToAverageOver must be an odd integer greater than two.

If *onlyValidValues* is set to true, then if any points in the averaging interval are missing, the point in the new time series is set to missing. If *onlyValidValues* is set to false and missing values are contained in the averaging interval, a smoothed point is still computed using the remaining valid values in the interval. If there are no valid values in the averaging interval, the point is set to missing.

If *useReduced* is set to true, then centered moving average points can still be computed at the beginning and end of the time series, even if there are less than *numberToAverageOver* values in the averaging interval. If *useReduced* is set to false, then the first and *last numberToAverageOver*/2 points of the resultant time series are set to missing.

Parameters:

`numberToAverageOver` – An integer containing the number of values to average over for computing the centered moving average. Must be odd and greater than two.
`onlyValidValues` – Either True or False, specifying whether all values in the averaging interval must be valid for the computed point in the new time series to be valid.
`useReduced` – Either True or False, specifying whether to allow points at the beginning and end of the resultant time series to be computed from a reduced (less than *numberToAverageOver*) set of points.

Example:

```
  avgData = tsData.centeredMovingAverage (5, TRUE, TRUE)
```

Returns: A new **TimeSeriesMath** object
Generated Exceptions: Throws a *HecMathException* if the *numberToAverageOver* is less than three or not odd.

8.15.11 Conic Interpolation from Elevation/Area Table

```
conicInterpolation(TimeSeriesMath tsData,
  string inputType,
  string outputType,
  floating-point storageScaleFactor )
```

Use the conic interpolation table in the current paired data set to develop a new time series data set from the interpolation of **tsData**.

The current paired data should be an Elevation-Area table. However, the first data pair is the initial conic depth, and the storage value at the first elevation in the table. If the initial conic depth is undefined, the function will calculate a value. Elevation-Area values in the table must be in ascending order.

tsData is either a time series of reservoir elevation or storage. The type is specified by setting *inputType* as "S(TORAGE)" or "E(LEVATION)". The desired output time series type is similarly set using outputType. The valid settings for *outputTyp*e are "S(TORAGE)", "E(LEVATION)" or "A(REA)". *inputType* and *outputType* must not be the same.

storageScaleFactor is an optional parameter used to scale input (by multiplying) and output (by dividing) storage values. For example, if the area in the conic interpolation table is expressed in square feet, *storageScaleFactor* could be set to 43,560 to convert the storage output to acre-feet.

Parameter type in the new time series is set according to *outputType*. If the output time series values are elevation, the time series units are set to the paired data x-units label. If the output time series values are area, the time series units are set to the paired data y-units label. If the output is storage, the units are not set and should be set by the user with the *setUnits* function.

See also: setUnits()
Parameters:
`tsData` – A TimeSeriesMath object representing elevation or storage.

`inputType` – A string specifying the parameter type for the input time series, either "S(TORAGE)" or "E(LEVATION)". Only the first character of the string is interpreted by the function.

`outputType` – A string specifying the parameter type for the output time series, either "S(TORAGE)", "E(LEVATION)" or "A(REA)". Only the first character of the string is interpreted by the function.

`storageScaleFactor` – A floating-point number used to scale input (by multiplying) and output (by dividing) storage values.

Examples:

```
tsStorage =
conicElevAreaCurve.conicInterpolation(
tsElev,
“Elevation”,
“Storage”,
1.0)

tsArea =
conicElevAreaCurve.conicInterpolation(
tsElev,
“Elevation”,
“Area”,
1.0)
```

Returns: A **new TimeSeriesMath** object

Generated Exceptions: Throws a *hec.hecmath.HecMathException* if *inputType* or *outputTyp*e cannot be interpreted as one of the allowed values; if *inputType* and *outputTyp*e are the same parameters; if values in the conic interpolation table are not in ascending order.

8.15.12 Convert Values to English Units

`convertToEnglishUnits``()`

Perform unit conversion of data values and unit labels in the current time series or paired data set from Metric (SI) units to English units. Determination of the unit system will be based upon the current units labels and parameter types. If the data units are already in English units or the unit system cannot be determined, no conversion occurs.

For paired data, both x and y values are converted. For time series data, missing values remain missing.

See also: convertToMetricUnits(), isEnglish(), isMetric()

Example: `englishDataSet= siDataSet.convertToEnglishUnits()`

Parameters: Takes no parameters

Returns: A **HecMath** object of the same type as the current object

8.15.13 Convert Values to Metric (SI) Units

`convertToMetricUnits``()`

Perform unit conversion of data values and unit labels in the current time series or paired data set from English units to Metric (SI) units. Determination of the unit system will be based upon the current units' labels and parameter types. If the units are already in Metric units or the unit system cannot be determined, no conversion occurs.

For paired data, both x and y values are converted. For time series data, missing values remain missing.

See also: convertToEnglishUnits(), isEnglish(), isMetric()
Parameters: Takes no parameters
Example: `siDataSet = englishDataSet.convertToMetricUnits()`
Returns: An **HecMath** object of the same type as the current object

8.15.14 Correlation Coefficients

```
correlationCoefficients(TimeSeriesMath tsData)
```

Computes the linear regression and other correlation coefficients between data in the current time series and **tsData**. Values in the current time series and **tsData** are matched by time to form data pairs for the correlation analysis. The data sets may be either regular or irregular time interval data.

The correlations statistics computed by the function are:

Number of Valid Values
Regression Constant
Regression Coefficient
Determination Coefficient
Standard Error of Regression
Adjusted Determination Coefficient
Adjusted Standard Error of Regression

These values are contained in a **LinearRegressionStatistics** object. The current **TimeSeriesMath** object forms the values of the independent variable (x-values), while values of the second time series comprise the dependent variable (y-values). The linear regression coefficients thus express how values in the second data set can be derived from values in the primary data set:

$$TS2(t) = a + b * TS1(t)$$

where "a" is the regression constant and "b" the regression coefficient.

See also: LinearRegressionStatistics
Parameters: tsData - A **TimeSeriesMath** object that forms the dependent variable for the regression analysis
Example:

```
linearRegressionData =
tsData.correlationCoefficients(otherTsData)
```

Returns: A **LinearRegressionStatistics** object holding the correlation data.
Generated Exceptions: Throws an *hec.hecmath.HecMathException* if the times in the current time series do not exactly match times in **tsData**.

8.15.15 Cosine Trigonometric Function

```
cos()
```

Derive a new time series or paired data set from the cosine of values of the current data set. The resultant data set values are in radians. For time series data, missing values are kept as missing.

For paired data sets, use the *setCurve* (see Sections 8.15.107 and 8.15.108) function to first select the paired data curve (or all curves) to apply the function. By default the function is applied to all paired data curves.

See also: setCurve()
Example: `newDataSet = dataSet.cos()`
Parameters: Takes no parameters
Returns: A **HecMath** object of the same type as the current object

8.15.16 Cyclic Analysis (Time Series)

```
cyclicAnalysis()
```

Derive a set of cyclic statistics from the current regular interval time series data set. The time series data set must have a time interval of "1HOUR", "1DAY" or "1MONTH". The function sorts the time series values into statistical "bins" relevant to the time interval. Values for the 1HOUR interval data are sorted into twenty-four bins representing the hours of the day, 0100 to 2400. The 1DAY interval data is apportioned to 365 bins for the days of the year. The 1MONTH interval data is sorted into tweleve bins for the months of the year.

The format of the resultant data sets is as a "pseudo" time series for the year 3000. For example, the cyclic analysis of one month of hourly interval data will produce pseudo time series data sets having twenty-four hourly values for the day January 1, 3000. If the statistical parameter is the "maximum" value, then the twenty-four values represent the maximum value occurring at that hour of the day in the current time series. The cyclic analysis of daily interval data will produce pseudo time series data sets having 365 daily values for the year 3000. The cyclic analysis of monthly interval data will result in pseudo time series data sets having twelve monthly values for the year 3000.

Fourteen pseudo time series data sets are derived by the cyclic analysis function for the following statistical parameters:

Number of values processed for each time interval
Maximum value
Time of maximum value
Minimum value
Time of minimum value
Average value
Probability exceedence percentiles for 5%, 10%, 25%, 50% (median value), 75%, 90%, and 95%
Standard deviation

The fourteen pseudo time series of cyclic statistics are returned by the function as an array of time series data sets. The parameter part of the record path for each time series is modified to indicate the type of the statistical parameter. For a flow record, the parameter "FLOW" would become "FLOW-MAX" for the maximum values statistics, "FLOW-P5" for the five percent percentile statistics, etc.

Parameters: Takes no parameters
Example: `cyclicData = tsData.cyclicAnalysis()`
Returns: A sequence of fourteen **TimeSeriesMath** objects, each of which is a pseudo time series data sets representing a statistical parameter.
Generated Exceptions: Throws a *hec.hecmath.HecMathException* if the time series is not regular interval or does not have a time interval of "1HOUR", "1DAY", or "1MONTH".

8.15.17 Decaying Basin Wetness Parameter

```
decayingBasinWetnessParameter(TimeSeriesMath tsPrecip,
floating-point decayRate)
```

Compute a time series of decaying basin wetness parameters from the regular interval time series data set of incremental precipitation, **tsPrecip,** by:

TSResult(t) = Rate * TSResult(t-1) + TSPrecip(t)

where Rate is **decayRate**, and 0 < Rate < 1.

The first value of the resultant time series data set, *TSResult(1)*, is set to the first value in the current time series data set. The current time series data set can be the same time series data set as **tsPrecip**. Missing values in the precipitation time series are taken as zero when applying the above equation.

Parameters:
`tsPrecip` – A regular interval **TimeSeriesMath** object representing precipitation
`decayRate` – a floating-point number in the range 0 < decayRate < 1.
Example:

```
tsWetness =
tsPrecip.decayingBasinWetnessParameter(
tsPrecip,
0.87)
```

Returns: A new **TimeSeriesMath** object

8.15.18 Divide by a Constant

```
divide(floating-point constant)
```

Divide all valid values in the current time series or paired data set by the value constant. For time series data, missing values are kept as missing. For paired data, constant divides the y-values only. Use the *setCurve* method to select the paired data curve(s).

See also: divide(TimeSeriesMath tsData); setCurve()
Parameters:
`constant` - A floating-point value to divide the values in the current data set (cannot be zero)
Example: `newDataSet = dataSet.divide(1.1)`
Returns: A new **HecMath** object of the same type as the current object

8.15.19 Divide by a Data Set

```
divide(TimeSeriesMath tsData)
```

Divide valid values in the current data set by the corresponding values in the data set **tsDat**a. Both data sets must be time series data sets.

When dividing one time series data set by another, there is no restriction that times in the two data sets match exactly. However, only values with coincident times will be divided. Times in the current time series data set that cannot be matched with times in the second data set are set to missing. Values in the current data set that are missing are kept as missing. If a value in the second data set is zero or missing, the value in the resultant data set is set to missing (divide by zero not allowed). Either or both data sets may be regular or irregular interval time series.

See also: divide(floating-point constant)
Parameters: `tsData` - A time series data set
Example: `newTsData = tsData.divide(otherTsData)`
Returns: A new **TimeSeriesMath** object

8.15.20 Estimate Values for Missing Precipitation Data

```
estimateForMissingPrecipValues(integer maxMissingAllowed)
```

Linearly interpolate estimates for missing values in the current regular or irregular interval time series data set. The current data set is expected to be cumulative precipitation and the data must be of type "INST-CUM". Use the *estimateForMissingValues* method for filling missing values in other types of time series data.

The rules used for interpolation of missing cumulative precipitation data are:

- If the values bracketing the missing period are increasing with time, only interpolate if the number of successive missing values does not exceed the value of *maxMissingAllowed*.
- If the values bracketing the missing period are decreasing with time, do not estimate for any missing values.
- If the values bracketing the missing period are equal, then estimate any number of missing values.

See also: estimateForMissingValues()
Parameters: `maxMissingAllowed -` an integer value for the maximum number of consecutive missing values between valid values
Example:

```
newPrecip =tsPrecip.estimateForMissingPrecipValues(5)
```

Returns: A new **TimeSeriesMath** object

8.15.21 Estimate Values for Missing Data

```
estimateForMissingValues(integer maxMissingAllowed)
```

Linearly interpolate estimates for missing values in the current regular or irregular interval time series data set. Do not interpolate if the number of successive missing values exceeds *maxMissingAllowed*.

See also: estimateForMissingPrecipValues()
Parameters: `maxMissingAllowed` - an integer value for the maximum number of consecutive missing values allowed for interpolation
Example: `newTsData = tsData.estimateForMissingValues(5)`
Returns: A new TimeSeriesMath object

8.15.22 Exponent

```
exp()
```

Derive a new time series or paired data set which is the e raised to the values of the current time series. For time series data, values that are missing in the current time series remain missing in the new time series. Also, values less than 0.0 will be set to missing the new time series.

For paired data sets, use the setCurve method to first select the paired data curve(s).

See also: setCurve()
Parameters: Takes no parameters
Example: `squaredDataSet = dataSet.exp()`
Returns: A new **HecMath** object of the same type as the current object

8.15.23 Exponentiation Function

```
exponentiation(floating-point constant)
```

Derive a new time series or paired data set from the exponentiation of values in the current data set by constant, by:

$$T2\ (i) = T1(i)^{constant}$$

For time series data, values that are missing in the current time series remain missing in the new time series.

For paired data sets, use the *setCurve* method to first select the paired data curve(s).

See also: setCurve()

Parameters: `constant` – a floating-point value representing the exponent.
Example: `squaredDataSet = dataSet.exponentiation(2.)`
Returns: A new **HecMath** object of the same type as the current object

8.15.24 Exponentiation Timeseries Function

```
exponentiation(HecMath tsMath)
```

Raise values in the current time series to the power of the parameter time series, **tsMath**. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current time series and **tsMat**h, the value in the current timeseries will be raised to the power of the value in **tsMat**h provided both values are valid (not missing). Points in the current time series which cannot be matched to valid points in **tsMath** are set to missing. Values in the current time series which are missing remain missing in the new time series.The new time series will always have quality defined. If a specific quality value in the parameter time series is questionable or rejected, that quality will be copied to the new time series.

Parameters: tsMath the time series of exponents for the current time series.
Example: `squaredDataSet = dataset.exponentiation`**`(HecMath tsMath)`**
Returns: a new time series resulting from the subtraction operation.. Derive a new time series or paired data set from the exponentiation of values in the current data set by constant, by:

$$T2\ (i) = T1(i)^{constant}$$

For time series data, values that are missing in the current time series remain missing in the new time series. For paired data sets, use the setCurve method to first select the paired data curve(s).

See also: setCurve()
Parameters: `constant` – a floating-point value representing the exponent.
Example: `squaredDataSet = dataSet.exponentiation(2.)`
Returns: A new **HecMath** object of the same type as the current object.

8.15.25 Extract Time Series Data at Unique Time Specification

```
extractTimeSeriesDataForTimeSpecification(
    string timeLevelString,
    string rangeString,
    boolean isInclusive,
    integer intervalWindow,
    boolean setAsIrregular )
```

Select/extract data points from the current regular or irregular interval time series data set based upon user defined time specifications. For example, the function may be used to extract from hourly interval data, the values observed every day at noon.

`timeLevelString` defines the time level/interval for extraction (year, month, day of the month, day of the week, or twenty-four hour time).

`rangeString` defines the interval range for data extraction applicable to the time level. For example, if *timeLevelString* is "MONTH”, a valid range would be "JAN-MAR". The **rangeString** variable can define a single interval value (e.g. "JAN" - select data from January only) or a beginning and ending range (e.g. "JAN-MAR" - select data for January through March). The valid *timeLevelString* and *rangeString* values are shown in Table 8.32.

Table 8.32 Valid timeLevelString and rangeString Values

timeLevelString	rangeString	Example rangeString
"YEAR"	Four-digit year value	"1938" or "1938-1945"
"MONTH”	Standard three-character abbreviation for month	"JAN" or "JAN-MAR" or “OCT-FEB”
"DAYMON(TH)"	Day of the month or "LASTDAY" string	"15" or "1-15 or "27-5" or "16-LASTDAY"
"DAYWEE(K)"	Standard three-character abbreviation for day of the week	"MON" or "SUN-TUE" or "FRI-WED"
"TIME"	Four digit 24-hour military-style clock time	"2400" or "0300-0600" or "2200-0130"

If desired, you may use one of the enumerated string constants to specify *timeLevelString*:

Year	TimeSeriesMath.LEVEL_YEAR_STRING
Month	TimeSeriesMath. LEVEL_MONTH_STRING
Day of Month	TimeSeriesMath. LEVEL_DAYMONTH_STRING
Day of Week	TimeSeriesMath.LEVEL_DAYWEEK_STRING
24-hour time	TimeSeriesMath.LEVEL_TIME_STRING

The parameter *isInclusive* determines whether the data extraction operation is either inclusive or exclusive of the specified range. For example, if *isInclusive* is "True" and the range is set to "JAN-MAR" for the "MONTH" time level, the extracted data will include all data in the months January through March for all the years of time series data. If *isInclusive* is "False" for this example, the extracted data covers the time April through December (is exclusive of the period January through March).

intervalWindow is only used when the *timeLevelString* is "TIME.” *intervalWindow* is the minutes before and after the time of day within which the data will be extracted. *intervalWindow* effectively increases the time range at the beginning and end *intervalWindow* minutes. For example, with a *rangeString* of "0300" and an *intervalWindow* of 10, data will be extracted from the selected time series if times falls within in the period 0250 to 0310.

setAsIrregular defines whether the extracted data is saved as regular interval or irregular interval data. Most often the time series data formed by the extraction process will no longer be regular interval, and *setAsIrregular* should be set to "True". Setting *setAsIrregular* to "False" will force an attempt to save the data as regular interval time data.

Parameters:

`timeLevelString` – A string specifying the time level selection.
`rangeString` – A string specifying time or time range for selection. Must be consistent with *timeLevelString*.
`isInclusive` – Either True or False, value. If true, data is extracted inclusive of the range specified by rangeString. If false, data is extracted exclusive of the range specified by *rangeString*.
`intervalWindow` – An integer value representing the minutes before and after the time of day within which the data will be extracted. Only applied when the *timeLevelString* is "TIME".
`setAsIrregular` – Either True orFalse, value. If true, data is automatically set as irregular time interval data. If false, the function will attempt to classify the data as regular time interval data.

Example:

```
SelectedData =
  tsData.extractTimeSeriesDataForTimeSpecification(
  “DAYMONTH”,
  “16-LASTDAY”,
  TRUE,
  0,
  TRUE)
```

Returns: A new **TimeSeriesMath** object
Generated Exceptions: Throws a *hec.hecmath.HecMathException* if the function could not successfully interpret *timeLevelString* or *rangeString*.

8.15.26 First Valid Date

`firstValidData``()`

Find the date and time of the first valid time series value

See also: firstValidValue()
Parameters: Takes no parameters
Example: `squaredDataSet = dataSet.firstValidData()`
Returns: An integer representing the date and time of the first valid value as an integer value translatable by **HecTime.**

8.15.27 First Valid Value

`firstValidValue``()`

Find the first valid value in the time series

See also: firstValidDate()
Parameters: Takes no parameters
Example: `squaredDataSet = dataSet.firstValidValue ()`
Returns: The floating-point value of the first valid time series value

8.15.28 Floor Function

`floor``()`

Derive a time series or paired data set with values of the current time series rounded down to the nearest whole number that is less than or equal to the value. For time series data, missing values are kept as missing.

For paired data sets, use the *setCurv*e method to first select the curve(s).

See also: setCurve(), ceil()
Example: `newDataSet = dataSet.floor()`
Parameters: Takes no parameters
Returns: A new **HecMath** object of the same type as the current object

8.15.29 Flow Accumulator Gage (Compute Period Average Flows)

`flowAccumulatorGageProcessor``(TimeSeriesMath tsCounts)`

Derive a new time series of period-average flows from a flow accumulator type gage. The current time series is assumed to containe the accumulated flow data, while the parameter time series, **tsCounts**, is assumed to have the corresponding time series of counts. The two time series data sets must match times exactly. The two time series are combined to compute a new time series of period average flow:

TsNew(t) = (TsAccFlow(t) - TsAccFlow(t-1)) / (TsCount(t) - TsCount(t-1))

where *TsAccFlow* is the gage accumulated flow time series and *TsCount* is the gage time series of counts.

In the above equation, if TsAccFlow(t), TsAccFlow(t-1), TsCount(t) or TsCount(t-1) are missing, TsNew(t) is set to missing. The new time series is assigned the data type "PER-AVER".

Parameters: `tsCounts` – A **TimeSeriesMath** object containing the counts.
Example:

```
tsPerAvgFlow =
tsAccumFlow.flowAccumulatorGageProcessor(tsCounts)
```

Returns: A new TimeSeriesMath object.
Generated Exceptions: Throws a *hec.hecmath.HecMathException* if times in the current object do not exactly match the times in **tsCounts**.

8.15.30 Modulo Function with both Arguments are Greater than Zero

```
fmod(HecMath tsMath)
```

Return the remainder of integer division of current time series by the parameter time series, **tsMath**. A new time series will be created which duplicates the time points of the current time series, where time points match for the current time series and **tsMath**, the value in the current time series will be devided by the value in **tsMath** provided both values are valid (no missing). Devide by zero is not allowed.When the **tsMath** time series value is zero, the value for the new time series will be set to missing. Also, points in the current time series which can not be match to valid points in **tsMath** are set to missing. For time series data, missing values are kept as missing. The new time series will always have quality defined. If a specific quality value in the parameter time series is questionable or rejected, that quality will be copied to the new time series.

For paired data sets, use the *setCurve* method to first select the curve(s).

See also: setCurve(),modulo()

Parameters:
tsMath the time series to be divided into the current time series.
`constant` – a floating-point value representing the exponent.
Example: `squaredDataSet = dataSet.fmod(2.)`
Returns: A new **HecMath** object of the same type as the current object.

8.15.31 Forward Moving Average Smoothing

```
forwardMovingAverage(integer numberToAverageOver)
```

Derive a new time series from the forward moving average of *numberToAverageOver* values in the current time series. *numberToAverageOve*r must be an integer greater than two. If the averaging interval contains a missing value, the smoothed value is computed from the remaining valid values in the interval. However, if there are less than two valid values in the interval, the value in the resultant data set is set to missing.

Parameters: `numberToAverageOver` – An integer containing the number of values to average over for computing the forward moving average.
Example: `tsAveraged = tsData.forwardMovingAverage(4)`
Returns: A new **TimeSeriesMath** object.
Generated Exceptions: Throws a *hec.hecmath.HecMathException* if the *numberToAverageOver* is less than two.

8.15.32 Forward Moving Average Smoothing of Time Series

```
forwardMovingAverage(integer numberToAverageOver, Boolean
onlyValidValues, Boolean useReduced)
```

Derive a new time series from the forward moving average of the last *numberToAverageOver* values of the current time series. If *onlyValidValues* is set to true, then if points in the averaging interval are missing values, the point in the new time series is set to missing. If *onlyValidValues* is set to false and missing values are contained in the averaging interval, a smoothed point is still computed using the valid values in the interval. If there are no valid values in the averaging interval, the point in the new time series is set to missing.

If *useReduced* is set to true, then forward moving average points can be still be computed at the beginning of the time series even if there are less than *numberToAverageOver* values in the interval. If *useReduced* is set to false, then the first *numberToAverageOver* points of the new time series are set to missing.

Parameters:

`numberToAverageOver` – An integer containing the number of values to average over for computing the forward moving average.

`onlyValidValues` – Either True or False, specifying whether all values in the averaging interval must be valid for the computed point in the new time series to be valid.

`useReduced` – Either True or False, specifying whether to allow points at the beginning and end of the resultant time series to be computed from a reduced (less than *numberToAverageOver*) set of points.

Example: `tsAveraged = tsData.forwardMovingAverage (4,TRUE,TRUE)`

Returns: A new time series computed from the forward moving average of current time series.

Generated Exceptions: Throws a *hec.hecmath.HecMathException* if the *numberToAverageOver* is less than two.

8.15.33 Generate Data Pairs from Two Time Series

```
generateDataPairs(TimeSeriesMath tsData, Boolean sort)
```

Generate a paired data set by pairing values (by time) from the current time series data set and the time series data set **tsDat**a. The values of the current time series form the x-ordinates, while values from **tsData** form the y-ordinates of the resulting paired data set. The times in the two time series data sets must match exactly. If a value for a time is missing in either time series, no data value pair is formed or added to the paired data set. If sort is "True", data pairs in the paired data set are sorted by ascending x-value.

The units and parameter type from the current time series data set are assigned to the paired data set x-units and x-parameter type. The units and parameter type from tsData are assigned to the paired data set y-units and y-parameter type.

An example application of the function would be to mate a time series record of stage to one of flow to generate a stage-flow paired data set.

Parameters:

`tsData` – A **TimeSeriesMath** object that forms the y-ordinates of the resulting paired data set.

`sort` – Either True or False, value. If true, sort data pairs in ascending x-value. If false, leave unsorted.

Example: `ratingCurve = tsStage.generateDataPairs(tsFlow)`

Returns: A **PairedDataMath** object with x-ordinates from the current time series, and y-ordinates from **tsData**.

Generated Exceptions: Throws a *hec.hecmath.HecMathException* if times from the current time series and **tsData** do not match exactly.

8.15.34 Generate a Regular Interval Time Series

```
generateRegularIntervalTimeSeries(string startTimeString,
  string endTimeString,
  string timeIntervalString,
  string timeOffsetString,
  floating-point initialValue)
```

Generate a new regular interval time series data set from scratch with times and values specified by the parameters. This is a function provided by the **TimeSeriesMath** module, and not an object method.

The parameters *startTimeString* and *endTimeString* are strings used to specify the beginning and ending time of the generated data set. These two parameters have the form of the standard HEC time string (e.g. "01JAN2001 0100").

The regular time interval is specified by *timeIntervalString*, and is a valid HEC time increment string (e.g. "1MIN", "15MIN", "1HOUR", "6HOUR", "1DAY", "1MONTH").

timeOffsetString is used to shift times in the resultant time series from the standard interval time. As an example, the offset could be used to shift times in regular hourly interval data from the top of the hour to six minutes past the hour. The parameter has the form "nT", where "n" is an integer number and "T" is one of the time increments: "M(INUTES)", "D(AYS)", "H(OUR)", "W(EEKS)", "MON(THS)" or "Y(EARS)" (characters in the parenthesis are optional). For example, a time offset of nine minutes would be expressed as "9M" or "9MIN".

Values in the time series data set are initialized to *initialValue*.

Parameters:

`startTimeString` - a string specifying a standard HEC time defining the time series data start date/time.
`endTimeString` - a string specifying a standard HEC time defining the time series data end date/time.
`timeIntervalString` - a string specifying a valid DSS regular time interval which defines the time interval of the new time series.
`timeOffsetString` – a string specifying the offset of the new time points from the regular interval time. This string may be an empty string or None.
`initialValue` - a floating-point number set to the initial value for all time series points. Set to *HecMath.UNDEFINED* to set all values to missing.

Example:

```
newTsData=TimeSeriesMath.generateRegularIntervalTimeSeries(
  “01FEB2002 0100”,
  “28FEB2002 2400”,
  “1HOUR”,
  “0M”,
  100.)
```

Returns: A new regular interval **TimeSeriesMat**h object initialized to **initialValue**. Data units and type are unset.
Generated Exceptions: Throws a *hec.hecmath.HecMathException* if time parameters cannot be successfully interpreted.

8.15.35 Get Data Container

```
getData()
```

Returns a copy of the *hec.io.DataContainer* for the current data set. For time series data sets, returns a *hec.io.TimeSeriesContainer*. For paired data sets, returns a *hec.io.PairedDataContainer*.

The *hec.io.TimeSeriesContainer* contains the time series values for a time series data set. The *hec.io.PairedDataContainer* contains the paired data values for a paired data set.

Parameters: Takes no parameters
Example: `container = dataset.getData()`
Returns: A *hec.io.DataContainer*

8.15.36 Get Data Type for Time Series Data Set

```
getType()
```

Get the data type for a time series data set.

Parameters: Takes no parameters
Example: dataSet.getType()
Returns: A string - "INST-CUM", "INST-VAL", "PER-AVER" or "PER-CUM".

8.15.37 Get Units Label for Data Set

```
getUnits()
```

Get the units label of the current data set. For a paired data set, returns the y-units label.

Parameters: Takes no parameters
Example: `dataSet.getUnits()`
Returns: A string

8.15.38 Gmean

```
gmean(list of HecMath tsMathArray)
```

Determine the geometric mean of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the geometric mean of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series.

The new time series will always have quality defined. For a specific time, if any of the quality values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: tsMathArray the array of time series to be compared with the current time series.
Example: `newDataSet = dataSet. gmean(list of HecMath tsMathArray)`
Returns: a new time series representing the geometric mean of all time series.

8.15.39 Hmean

`hmean``(list of HecMath tsMathArray)`

Determine the harmonic mean of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the harmonic mean of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series.

The new time series will always have quality defined. For a specific time, if any of the quality values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: t*sMathArray* the array of time series to be compared with the current time series.
Example: `newDataSet = dataSet. hmean(list of HecMath tsMathArray)`
Returns: a new time series representing the harmonic mean of all time series.

8.15.40 Integer Division by a Constant

```
integerDivide(floating-point constant)
```

Divide values in the current time series by a constant and truncate the result to an integer value. Times in which values are missing in the current time series remain missing in the new time series.

Parameters: constant the value to divide values in current time series.
Example: `dataSet.integerDivide(2.0)`
Returns: a new time series resulting from the integer division operation.

8.15.41 Integer Divison by an Object

```
integerDivide(HecMath tsMath)
```

Divide the current time series by the parameter time series, **tsMath** and truncate the result to an integer value. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current time series and **tsMath**, the value in the current time series will be divided by the value in **tsMath** provided both values are valid (not missing). Divide by zeroes are not allowed. When the **tsMath** time series value is zero, the value for the new time series will be set to missing. Points in the current time series which cannot be matched to valid points in **tsMath** are set to missing. Values in the current time series which are missing remain missing in the new time series.

The new time series will always have quality defined. If a specific quality value in the parameter time series is questionable or rejected, that quality will be copied to the new time series.

Parameters: tsMath the time series to be divided into the current time series.
Example: `newDataSet = dataset.integerDivide(HecMath tsMath)`
Returns: a new time series resulting from the integer division operation.

8.15.42 Interpolate Time Series Data at Regular Intervals

```
interpolateDataAtRegularInterval(
  string timeIntervalString,string timeOffsetString)
```

Derive a regular interval time series data set by interpolation of the current regular or irregular interval time series data set.

The new time interval is set by *timeIntervalString* which must be a valid HEC time interval string (e.g. "1MIN", "15MIN", "1HOUR", "6HOUR", "1DAY", "1MONTH").

Times in the resultant time series may be shifted (offset) from the regular interval time by the increment specified by *timeOffsetString*. As an example, the offset could be used to shift times from the top of the hour to six minutes past the hour. If no offset is used *timeOffsetString* should be a blank or empty string.

Whether the time series data type is "INST-VAL", "INST-CUM", "PER-AVE", or "PER-CUM" controls how the interpolation is performed. Interpolated values are derived from "INST-VAL" or "INST-CUM" data using linear interpolation. Values are derived from "PER-AVE" data by computing the period average value over the time interval. Values are derived from "PER-CUM" data by computing the period cumulative value over the new time interval.

For example, if the original data set is hourly data and the new regular interval data set is to have a six hour time interval:

The value for "INST-VAL" or "INST-CUM" type data is computed from the linear interpolation of the hourly points bracketing the new six hour time point.

The value for "PER-AVE" type data is computed from the period average value over the six hour interval.

The value for "PER-CUM" type data is computed from the accumulated value over the six hour interval.

The treatment of missing value data is also dependent upon data type. Interpolated "INST-VAL" or "INST-CUM" points must be bracketed or coincident with valid (not missing) values in the original time series; otherwise the interpolated values are set as missing. Interpolated "PER-AVE" or "PER-CUM" data must contain all valid values over the interpolation interval; otherwise the interpolated value is set as missing.

Parameters:

`timeIntervalString` – A string specifying the regular time interval for the resultant time series.

`timeOffsetString` – A string specifying the offset of the new time points from the regular interval time. This variable may be an empty string (" ").

Example:

```
newTsData =
tsData.interpolateDataAtRegularInterval(
  “15MIN”,
  “  ”)
```

Returns: A new regular interval **TimeSeriesMath** object.

8.15.43 Inverse (1/X) Function

```
inverse()
```

Derive a new time series or paired data set from the inverse (1/x) of values of the current data set. The inverse value is computed by 1.0 divided by the value of the current data set. If a data value is equal to 0.0, the value in the resultant data set is set to missing. For time series data, if the original value is missing, the value remains missing in the resultant data set.

For paired data sets, use the *setCurve* method to first select the paired data curve(s).

See also: setCurve()
Parameters: Takes no parameters
Example: `newDataSet = dataSet.inverse()`
Returns: A **HecMath** object of the same type as the current object.

8.15.44 Determine if Data is in English Units

```
isEnglish()
```

Determine if the current time series or paired data set is in English units. The function examines the data set parameter type and units label to establish the unit system.

See also: isMetric(); convertToEnglishUnits()
Parameters: No parameters
Example: `if dataSet.isEnglish() : print “English Units”`
Returns: True if the data set units are English, otherwise False.
Generated Exceptions: Throws a *hec.hecmath.HecMathException* if the unit system cannot be determined (parameter type and units label undefined).

8.15.45 Determine if Data is in Metric Units

```
isMetric()
```

Determine if the current time series or paired data set is in Metric (SI) units. The function examines the data set parameter type and units label to establish the unit system.

See also: isEnglish(); convertToMetricUnits()
Parameters: Takes no parameters
Example: `if dataSet.isMetric() : print “SI Units”`
Returns: True if the data set units are Metric, otherwise False.
Generated Exceptions: Throws a *hec.hecmath.HecMathException* if the unit system cannot be determined (parameter type and units label undefined).

8.15.46 Determine if Computation Stable for Given Muskingum Routing Parameters

```
isMuskingumRoutingStable(integer numberSubreaches,
  floating-point muskingumK,
  floating-point muskingumX)
```

Check for possible instability for the given Muskingum Routing parameters.
Test if the input parameters satisfy the stability criteria:

1/(2(1-x)) <= K/deltaT <= 1/2x

where deltaT = (time series time interval)/numberSubreaches

Parameters:

`numberSubreaches` – integer specifying the number of routing subreaches.

`muskingumK` –floating-point number specifying the Muskingum "K" parameter, in hours.

`muskingumX` - floating-point number specifying the Muskingum "x" parameter, between 0.0 and 0.5 (inclusive).

Example:

```
warning = tsDataSet.isMuskingumRoutingStable(
reachCount,
kVal,
xVal)
  if warning :
  print warning
  return
```

Returns: A string if the stability criteria is not met. The string contains a warning message detailing the specific instability problem. Otherwise returns None.

Generated Exceptions: Throws a *hec.hecmath.HecMathException* if the current time series is not a regular interval time series, or if values for *numberSubreaches* or *muskingumX* are invalid.

8.15.47 Last Valid Value's Date and Time

`lastValidDate``()`

Find and return the date and time of the last valid (non-missing) value in a time series data set.

Parameters: Takes no parameters

Example: `tsData.lastValidDate()`

Returns: An integer value translatable by **HecTime** representing the date and time of the last valid time series value.

8.15.48 Last Valid Value in a Time Series

`lastValidValue``()`

Find and return the last valid (non-missing) value in a time series data set.

Parameters: Takes no parameters

Example: `tsData.lastValidValue()`

Returns: A floating-point value representing the last valid time series value.

8.15.49 Natural Log, Base "e" Function

`log``()`

Derive a new time series or paired data set from the natural log (log base "e") of values of the current data set. Missing values in the original data set remain missing. Values less than or equal to 0.0 will be set to missing.

For paired data sets, use the *setCurve* method to first select the paired data curve(s).

See also: log10(), setCurve()
Parameters: Takes no parameters
Example: `newDataSet = dataSet.log()`
Returns: A new **HecMath** object of the same type as the current object.

8.15.50 Log Base 10 Function

`log10()`

Derive a new time series or paired data set from the log base 10 of values of the current data set. Missing values in the original data set remain missing. Values less than or equal to 0.0 will be set to missing.

For paired data sets, use the *setCurve* method to first select the paired data curve(s).

See also: `log(), setCurve()`
Parameters: Takes no parameters
Example: `newDataSet = dataSet.log10()`
Returns: A new **HecMath** object of the same type as the current object.

8.15.51 Maximum Value in a Time Series

`max()`

Find and return the maximum value of the current time series data set. Missing values are ignored.

Parameters: Takes no parameters
Example: `maxVal = tsData.max()`
Returns: A floating-point value representing the maximum value of the current time series.

8.15.52 Maximum Value in a Time Series (*tsMathArrary*)

max(list of HecMath tsMathArray)

Determine the maximum of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the maximum of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series.

The new time series will always have quality defined. For a specific time, if any of the quality values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: *tsMathArray* the array of time series to be compared with the current time series.
Example: `newDataSet = dataSet.max(list of HecMath tsMathArray)`
Returns: a new time series representing the maximum values of all time series.

8.15.53 Maximum Value's Date and Time

maxDate()

Find and return the date and time of the maximum value for the current time series data set. Missing values are ignored.

Parameters: Takes no parameters
Example: `maxDateTime = tsData.maxDate()`
Returns: An integer value translatable by **HecTime** representing the date and time of the maximum time series value.

8.15.54 Mean Time Series Value

mean()

Compute the mean value of the current time series data set. Missing values are ignored.

Parameters: Takes no parameters
Example: `meanVal = tsData.mean()`
Returns: A floating-point value representing the mean value of the current time series.

8.15.55 Mean Time Series Value (*tsMathArray*)

```
mean(list of HecMath tsMathArray)
```

Determine the arithmetic mean of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the arithmetic mean of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series.

The new time series will always have quality defined. For a specific time, if any of the quailty values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: *tsMathArray* the array of time series to be compared with the current time series.
Example: `newDataSet = dataSet.mean(list of HecMath tsMathArray)`
Returns: a new time series representing the arithmetic mean of all time series.

8.15.56 Median Time Series Value

```
med(list of HecMath tsMathArray)
```

Determine the median (50th percentile) of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the median of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series.

The new time series will always have quality defined. For a specific time, if any of the quailty values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.

Parameters: *tsMathArray* the array of time series to be compared with the current time series.
Example: `newDataSet = dataSet.med(list of HecMath tsMathArray)`
Returns: a new time series representing the median of all time series.

8.15.57 Merge Paired Data Sets

```
mergePairedData(PairedDataMath pdData)
```

Merge the current paired data set with the paired data set *pdDat*a. The resultant paired data set includes all the paired data curves from the current data set. Depending upon a previous use of the *setCurveMethod* on *pdData*, a single selected paired data curve or all curves from *pdData* are appended to the merged data set. The x-values for the two paired data sets must match exactly.

See also: setCurve()
Parameters: `pdData` - a paired data set with x-ordinates matching those of the current data set.
Example: `mergedCurve = curve.mergePairedData(anotherCurve)`
Returns: A *new PairedDataMath* object.

8.15.58 Merge Two Time Series Data Sets

```
mergeTimeSeries(TimeSeriesMath tsData)
```

Merge data from the current time series data set with the time series data set **tsDat**a. The resultant time series data set includes all the data points in the two time series, except where the data points occur at the same time. When data points from the two data sets are coincident in time, valid values in the current time series take precedence over valid values from **tsData**. However, if a coincident point is set to missing in the current time series data set, a valid value from **tsData** will be used for time in the resultant data set. If the values are missing for both data sets, the value is missing in the resultant data set.

The data sets for merging may have either regular or irregular time interval time series data. The data sets are tested to determine if they both have the same regular time interval. If not, the resultant data set is typed as an irregular interval data set.

Parameters: `tsData` - a time series data set for merging with the current time series data set.
Example: `tsMerged = tsData.mergeTimeSeries(otherTsData)`
Returns: A new **TimeSeriesMath** object.

8.15.59 Minimum Value in a Time Series

```
min()
```

Find and return the minimum value of the current a time series data set. Missing values are ignored.

Parameters: Takes no parameters
Example: `minVal = tsData.min()`
Returns: A floating-point value representing the minimum value of the current time series.

8.15.60 Minimum Value in a Time Series (*tsMathArray*)

```
min(list of HecMath tsMathArray)
```

Determine the minimum of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the minimum of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series.

The new time series will always have quality defined. For a specific time, if any of the quailty values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: *tsMathArra*y the array of time series to be compared with the current time series.

Example: `newDataSet = dataSet.min(list of HecMath tsMathArray)`
Returns: a new time series representing the minimum values of all time series.

8.15.61 Minimum Value's Date and Time

```
minDate()
```

Find and return the date and time of the minimum value for the current time series data set. Missing values are ignored.

Parameters: Takes no parameters
Example: `minDateTime = tsData.minDate()`
Returns: An integer value translatable by HecTime representing the date and time of the minimum time series value.

8.15.62 Modified Puls or Working R&D Routing Function

```
modifiedPulsRouting(TimeSeriesMath tsFlow,
  integer numberSubreaches,
  floating-point muskingumX)
```

The current data set is a paired data set containing the storage-discharge table for Puls routing, where the x-values are storage and the y-values are discharge. The function derives a new time series data set from the Modified Puls or Working R&D routing of the time series data set **tsFlow**. *numberSubreaches* is the number of routing subreaches.

The Working R&D method provides a means of including the effects of inflow on reach storage by use of the Muskingum "x" wedge coefficient. The Working R&D method is activated in the computation if muskingumX is greater than 0.0. However, muskingumX cannot be greater that 0.5.

Parameters:
`tsFlow` – A regular interval time series data set for routing.
`numberSubreaches` – Number of routing subreaches.
`muskingumX` - Muskingum "X" parameter, between 0.0 and 0.5 (inclusive). Enter 0.0 to route by the Modified Puls method, or a value greater than 0.0 to apply the Working R&D.
Example:

```
routedFlow =
storDichareCurve.modifiedPulsRouting(
  tsFlow,
  reachCount,
  coefficient)
```

Returns: A new **TimeSeriesMath** object.
Generated Exceptions: Throws a *hec.hecmath.HecMathException* if the **tsMath** is not a regular interval time series; if *muskingumX* is less than 0.0 or greater than 0.5; if the current paired data set does not have both ascending x and y values.

8.15.63 Modulo

```
modulo(floating-point constant)
```

Return the remainder of integer division of current time series by a constant.Times in which values are missing in the current time series remain missing in the new time series.

Parameters: constant the value to divide values in current time series.
Example: `newDataSet = dataSet.modulo(floating-point constant)`
Returns: a new time series resulting from the operation.

8.15.64 Modulo (tsMath)

```
modulo(HecMath tsMath)
```

Return the remainder of integer division of current time series by the parameter time series, **tsMat**h. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current time series and **tsMath**, the value in the current time series will be divided by the value in **tsMath** provided both values are valid (not missing). Divide by zeroes are not allowed. When the **tsMath** time series value is zero, the value for the new time series will be set to missing. Points in the current time series which cannot be matched to valid points in **tsMath** are set to missing. Values in the current time series which are missing remain missing in the new time series.

The new time series will always have quality defined. If a specific quality value in the parameter time series is questionable or rejected, that quality will be copied to the new time series.

Parameters: **tsMath** the time series to be divided into the current time series.
Example: `newDataSet = dataSet.modulo(HecMath tsMath)`
Returns: a new time series resulting from the operation.

8.15.65 Multiple Linear Regression Coefficients

```
multipleLinearRegression( sequence tsDataSequence,
  floating-point minimumLimit,
  floating-point maximumLimit)
```

Compute the multiple linear regression coefficients between the current time series data set and the array of independent time series data sets in *tsDataSequence*. The function stores the regression coefficients in a new paired data set. This paired data set may be used with the *multipleLinearRegressio*n function to derive a new estimated time series data set.

For the general linear regression equation, a dependent variable, Y, may be computed from a set independent variables, Xn:

Y = B0 + B1*X1 + B2*X2 + B3*X3

where Bn are linear regression coefficients.

For time series data sets, an estimate of the original time series data set values may be computed from a set of independent time series data sets using regression coefficients such that:

TsEstimate(t) = B0 + B1*TS1(t) + B2*TS2(t) + … + Bn*TSn(t)

where Bn are the set of regression coefficients and TSn are the time series data sets contained in *tsDataSequence*.

The parameters *minimumLimit* and *maximumLimit* may be used to exclude out of range values in the current time series data set from the regression determination. *minimumLimit* or *maximumLimit* may be entered as "Constants.UNDEFINED" to ignore the minimum or maximum value check.

See also: applyMultipleLinearRegression()
Parameters:

`tsDataSequence` – sequence of TimeSeriesMath objects, which form the independent variables in the regression equation. Must all be regular interval and have the same time interval.
`minimumLimit` – A floating-point value. Values in the current time series exceeding minimumLimit are excluded from the regression analysis. Set to Constants.UNDEFINED to ignore this option.
`maximumLimit` – A floating-point value. Values in the current time series exceeding maximumLimit are excluded from the regression analysis. Set to Constants.UNDEFINED to ignore this option.

Example:
```
  regression = tsFlow.multipleLinearRegression (
    [tsUpstrFlow1, tsUpstrFlow2, tsUpstrFlow3],
    0.,
    100000.)
```
Returns: A **new PairedDataMath** object containing the computed regression coefficients.
Generated Exceptions: Throws a *hec.hecmath.HecMathException* if the current data set and the data sets in *tsDataSequence* are not regular interval time series data sets with the same interval time.

8.15.66 Multiply by a Constant

```
multiply(floating-point constant)
```

Multiply the value constant to all valid values in the current time series or paired data set. For time series data, missing values are kept as missing. For paired data, constant multiplies the y-values only. Use the *setCurveMethod* to first select the paired data curve(s).

See also: multiply(TimeSeriesMath tsData); setCurve()
Parameters: `constant` - A floating-point precision value.
Example: `newDataSet = dataSet.multiply(1.5)`
Returns: A new **HecMath** object of the same type as the current object.

8.15.67 Multiply by a Data Set

```
multiply(TimeSeriesMath tsData)
```

Multiply valid values in the current data set by the corresponding values in the data set **tsData**. Both data sets must be time series data set.

When multiplying one time series data set to another, there is no restriction that times in the two data sets match exactly. However, only values with coincident times will be multiplied. Times in the current time series data set that cannot be matched with times in the second data set are set to missing. Values in the current data set that are missing are kept as missing. Either or both data sets may be regular or irregular interval time series.

See also: multiply(floating-point constant)
Parameters: `tsData` - A time series data set.
Example: `newTsData = tsData.multiply(otherTsData)`
Returns: A new **TimeSeriesMath** object.

8.15.68 Muskingum Hydrologic Routing Function

```
muskingumRouting(integer numberSubreaches,
  floating-point muskingumK,
  floating-point muskingumX)
```

Route the current regular interval time series data set by the Muskingum Routing method. The current data set must be a regular interval time series data set. *muskingumK* is the Muskingum "K" parameter, in hours, and *muskingumX* is the Muskingum "x" parameter. *muskingumX* cannot be less than 0.0 or greater than 0.5.

The set of Muskingum routing parameters may potentially produce numerical instabilities in the routed time series. Use the function *isMuskingumRoutingStable()* to test if the Muskingum routing parameters may potentially have instabilities.

See also: isMuskingumRoutingStable()
Parameters:
`numberSubreaches` – An integer specifying the number of routing subreaches.
`muskingumK` – A floating-point number specifying the Muskingum "K" parameter in hours.
`muskingumX` – A floating-point number specifying the Muskingum "x" parameter, between 0.0 and 0.5
Example:

```
routedFlows = tsFlows.muskingumRouting(reachCount, K, x)
```

Returns: A new TimeSeriesMath object.
Generated Exceptions: Throws a *hec.hecmath.HecMathException* if the current time series is not a regular interval time series; if *muskingumX* is less than 0.0 or greater than 0.5.

8.15.69 Negation Function

```
negative()
```

Derive a new time series composed of the negatives of the values of the current time series.Values which are missing in the original time series will be missing in the new time series.

Parameters: Takes no parameters
Example: `newDataSet = dataSet.neg()`
Returns: A new time series composed of the negatives of the values of the current time series.

8.15.70 Number of Invalid Values in a Time Series

`numberInvalidValues``()`

Count and return the number of invalid values in the current time series data set.

Parameters: Takes no parameters
Example: `invalidCount = tsData.numberInvalidValues()`
Returns: An integer of the count of invalid (non-missing) time series values.

8.15.71 Number of Missing Values in a Time Series

`numberMissingValues``()`

Count and return the number of missing values in the current time series data set.

Parameters: Takes no parameters
Example: `missingCount = tsData.numberMissingValues()`
Returns: An integer of the count of missing time series values.

8.15.72 Number of Questioned Values in a Time Series

`numberQuestionedValues``()`

Count and return the number of questioned values in the current time series data set.

Parameters: Takes no parameters
Example: `quiestionedCount = tsData.numberQuestionedValues()`
Returns: An integer of the count of questioned (non-missing) time series values.

8.15.73 Number of Rejected Values in a Time Series

`numberRejectedValues()`

Count and return the number of rejected values in the current time series data set.

Parameters: Takes no parameters
Example: `rejectedCount = tsData.numberRejectedValues()`

Returns: An integer of the count of rejected (non-missing) time series values.

8.15.74 Number of Valid Values in a Time Series

```
numberValidValues()
```

Count and return the number of valid values in the current time series data set.

Parameters: Takes no parameters
Example: `validCount = tsData.numberValidValues()`
Returns: An integer of the count of valid (non-missing) time series values.

8.15.75 Olympic Smoothing

```
olympicSmoothing(integer numberToAverageOver,
  boolean onlyValidValues,
  boolean useReduced)
```

Derive a new time series from the Olympic smoothing of *numberToAverageOver* values in the current time series. *numberToAverageOver* must be and odd integer and greater than. Similar to centered moving average smoothing, except that the minimum and maximum values over the averaging interval are excluded from the computation.

If *onlyValidValues* is set to true, then if any values in the averaging interval are missing, the point in the resultant time series is set to missing. If *onlyValidValues* is set to false and there are missing values in the averaging interval, a smoothed point is still computed using the remaining valid values in the interval. If there are no valid values in the averaging interval, the point in the resultant time series is set to missing.

If *useReduced* is set to true, then moving average values can be still be computed at the beginning and end of the time series even if there are less than *numberToAverageOver* values in the interval. If *useReduced* is set to false, then the first and last *numberToAverageOver*/2 points of the resultant time series are set to missing.

Parameters:

`numberToAverageOver` – An integer specifying the number of values to average over for computing the smoothed time series. Must be an odd integer greater than two.

`onlyValidValues` – Either True or False, specifying whether all values in the averaging interval must be valid for the computed point in the resultant time series to be valid.
`useReduced` - Either True or False, specifying whether to allow points at the beginning and end of the smoothed time series to be computed from a reduced (less than *numberToAverageOver*) number of values. Otherwise, set the first and last *numberToAverageOver*/2 points of the new time series to missing.

Example: `avgData = tsData.olympicSmoothing(5, )`
Returns: A new TimeSeriesMath object.
Generated Exceptions: Throws a **HecMathException** if the *numberToAverageOver* is less than three or not odd.

8.15.76 P1 Function

```
p1(list of HecMath tsMathArray)
```

Determine the first percentile of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the 1st percentile of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series.

The new time series will always have quality defined. For a specific time, if any of the quailty values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: *tsMathArray* the array of time series to be compared with the current time series.
Example: `newDataSet = dataSet.p1(list of HecMath tsMathArray)`
Returns: a new time series representing the first percentile of all time series.

8.15.77 P2 Function

```
p2(list of HecMath tsMathArray)
```

Determine the 2nd percentile of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the 2nd percentile of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series.

The new time series will always have quality defined. For a specific time, if any of the quailty values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: *tsMathArray* the array of time series to be compared with the current time series.
Examples: `newDataSet = dataSet.p2(list of HecMath tsMathArray)`
Returns: a new time series representing the 2nd percentile of all time series.

8.15.78 P5 Function

```
p5(list of HecMath tsMathArray)
```

Determine the fifth percentile of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the fifth percentile of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series.

The new time series will always have quality defined. For a specific time, if any of the quailty values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: *tsMathArray* the array of time series to be compared with the current time series.

Example: `newDataSet = dataSet.p5(list of HecMath tsMathArray)`
Returns: a new time series representing the fifth percentile of all time series.

8.15.79 P10 Function

```
p10(list of HecMath tsMathArray)
```

Determine the tenth percentile of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the tenth percentile of all time series for that time provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series.

The new time series will always have quality defined. For a specific time, if any of the quality values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: *tsMathArray* the array of time series to be compared with the current time series.
Example: `newDataSet = dataSet.p10(list of HecMath tsMathArray)`
Returns: a new time series representing the tenth percentile of all time series.

8.15.80 P20 Function

```
p20(list of HecMath tsMathArray)
```

Determine the 20th percentile of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the 20th percentile of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series.

The new time series will always have quality defined. For a specific time, if any of the quailty values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: *tsMathArray* the array of time series to be compared with the current time series.
Example: `newDataSet = dataSet.p20(list of HecMath tsMathArray)`
Returns: a new time series representing the 20th percentile of all time series.

8.15.81 P25 Function

```
p25(list of HecMath tsMathArray)
```

Determine the 25th percentile of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the 25th percentile of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series.

The new time series will always have quality defined. For a specific time, if any of the quailty values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: *tsMathArray* the array of time series to be compared with the current time series.
Example: `newDataSet = dataSet.p25(list of HecMath tsMathArray)`
Returns: a new time series representing the 25th percentile of all time series.

8.15.82 P75 Function

```
p75(list of HecMath tsMathArray)
```

Determine the 75th percentile of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the 75th percentile of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArra*y are set to missing. Values in the current time series which are missing remain missing in the new time series.

The new time series will always have quality defined. For a specific time, if any of the quailty values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: *tsMathArray* the array of time series to be compared with the current time series.
Example: `newDataSet = dataSet.p75(list of HecMath tsMathArray)`
Returns: a new time series representing the 75th percentile of all time series.

8.15.83 P80 Function

```
p80(list of HecMath tsMathArray)
```

Determine the 80th percentile of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the 80th percentile of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series. The new time series will always have quality defined. For a specific time, if any of the quailty values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: **tsMathArray** the array of time series to be compared with the current time series.
Example: `newDataSet = dataSet.p80(list of HecMath tsMathArray)`
Returns: a new time series representing the 80th percentile of all time series.

8.15.84 P89 Function

```
p89(list of HecMath tsMathArray)
```

Determine the 89th percentile of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the 89th percentile of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series. The new time series will always have quality defined. For a specific time, if any of the quailty values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: *tsMathArray* the array of time series to be compared with the current time series.
Example: `newDataSet = dataSet.p89(list of HecMath tsMathArray)`
Returns: a new time series representing the 89th percentile of all time series.

8.15.85 P90 Function

```
p90(list of HecMath tsMathArray)
```

Determine the 90th percentile of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the 90th percentile of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series. The new time series will always have quality defined. For a specific time, if any of the quailty values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: *tsMathArray* the array of time series to be compared with the current time series.

Example: `newDataSet = dataSet.`**`p90(`**`list of HecMath tsMathArray`**`)`**
Returns: a new time series representing the 90th percentile of all time series.

8.15.86 P95 Function

```
p95(list of HecMath tsMathArray)
```

Determine the 95th percentile of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the 95th percentile of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series. The new time series will always have quality defined. For a specific time, if any of the quailty values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: *tsMathArray* the array of time series to be compared with the current time series.
Example: `newDataSet = dataSet.`**`p95(`**`list of HecMath tsMathArray`**`)`**
Returns: a new time series representing the 95th percentile of all time series.

8.15.87 P99 Function

```
p99(list of HecMath tsMathArray)
```

Determine the 99th percentile of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the 99th percentile of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series. The new time series will always have quality defined. For a specific time, if any of the quailty values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: *tsMathArray* the array of time series to be compared with the current time series.
Example: `newDataSet = dataSet.`**`p99(`**`list of HecMath tsMathArray`**`)`**
Returns: a new time series representing the 99th percentile of all time series.

8.15.88 Period Constants Generation

`periodConstants``(TimeSeriesMath tsData)`

Derive a new time series data set by applying values in the current time series data set to the times defined by the time series data set **tsData**. Both time series data sets may be regular or irregular interval. Values in a new time series are set according to:

$$ts1(j) \leq tsnew(i) < ts1(j+1)\,, \quad TSNEW(i) = TS1(j)$$

where ts1 is the time in the current time series, TS1 is the value in the current time series, tsnew is the time in the new time series, TSNEW is the value in the new time series.

If times in the new time series precede the first data point in the current time series, the value for these times is set to missing. If times in the new time series occur after the last data point in the current time series, the value for these times is set to the value of the last point in the current time series. The interpolation of values with the *periodConstants* function is shown in Figure 8.13.

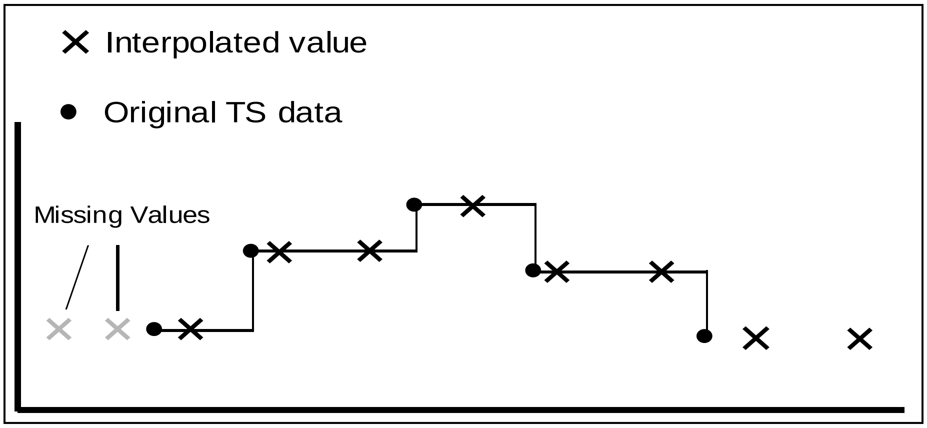


Figure 8.13 Interpolation of Time Series Values Using Period Constants Function

Parameters: tsData - a regular or irregular interval time series data set.

Example: `tsConstants = tsValues.periodConstants(tsData)`
Returns: A new **TimeSeriesMath** object.

8.15.89 Polynomial Transformation

```
polynomialTransformation(TimeSeriesMath tsData)
```

Compute a polynomial transformation of a regular or irregular interval time series data set, **tsData**, using the polynomial coefficients stored in the current paired data set. Missing values in **tsData** remain missing in the resultant data set.

A new time series can be computed from an existing time series with the polynomial expression:

$$TS2\ (t) = B1 * TS1(t) + B2 * TS1(t)^{2} + ... + Bn * TS1(t)^{n}$$

where Bn are the polynomial coefficients for term “n.”

Values for the polynomial coefficients are stored in the x-values of the current paired data set. Before the above equation is applied, values in the input time series are adjusted by subtracting off the paired data “datum” value if defined. The units label and parameter type for the resultant time series are copied from the current paired data set x-units and parameter type.

See also: polynomialTransformationWithIntegral()
Parameters: **tsData** - a regular or irregular interval time series data set.
Example: `tsXform = pdCoef.polynomialTransformation(tsData)`
Returns: A new **TimeSeriesMath** object.

8.15.90 Polynomial Transformation with Integral

```
polynomialTransformationWithIntegral(TimeSeriesMath tsData)
```

Compute a polynomial transformation with integral of a regular or irregular interval time series data set, **tsData**, using the polynomial coefficients stored in the current paired data set. Missing values in **tsData** remain missing in the resultant data set.

This function is similar to the polynomialTranformation method, and the same set of polynomial coefficients is used. The equation for the polynomial transform is modified so that the transform of **tsData** is computed from the integral of the polynomial coefficients:

$$TS2(t) = B1 * TS1(t)^{2}/2 \quad + B2*TS1(t)^{3}/3 + \ldots + Bn * TS1(t)^{n+1}/(n+1)$$

where Bn are the polynomial coefficients for term "n".

Values for the polynomial coefficients are stored in the x-values of the current paired data set. Before the above equation is applied, values in the input time series are adjusted by subtracting off the paired data "datum" value if defined. The units label and parameter type for the resultant time series are copied from the current paired data set x-units and parameter type.

See also: polynomialTransformation()
Parameters: **tsData** - a regular or irregular interval time series data set.
Example: `tsXform = pdCoef.polynomialTransformationWithIntegral(tsData)`
Returns: A new **TimeSeriesMath** object.

8.15.91 Product Function

```
product(list of HecMath tsMathArray)
```

Multiply the current time series with the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be multiplied provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series. The new time series will always have quality defined. For a specific time, if any of the quailty values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: *tsMathArray* the array of time series to be multiplied with the current time series.
Example: `newDataSet = dataset.product(list of HecMath tsMathArray)`
Returns: a new time series representing the multiplication of the two time series

8.15.92 Rating Table Interpolation

```
ratingTableInterpolation(TimeSeriesMath tsData)
```

Transform/interpolate values in the time series data set **tsData** using the rating table x-y values stored in the current paired data set. For example, you can use the function to transform a time series of stage to a time series of flow using a stage-flow rating table. **tsData** may be a regular or irregular time interval data set. Missing values in **tsData** are kept missing in the resultant data set.

Create the paired data set with the rating table option to set values for "datum", "shift", and "offset". By default these values are 0.0. The shift is added to and the datum subtracted from all input time series values. If the rating table is Log-Log, the table x-values are adjusted by subtracting the offset.

Units and parameter type in resultant time series data set are defined by the y-units label and parameter type of the current paired data set. All other names and labels are copied over from **tsData**.

See also: reverseRatingTableInterpolation()
Parameters: **tsData** - a regular or irregular interval **TimeSeriesMath** object.
Example: `tsFlow =`
```
      stageFlowCurve.ratingTableInterpolation(tsStage)
```
Returns: A new **TimeSeriesMath** object.

8.15.93 Replace Specific Values

```
replaceSpecificValues(HecDouble from, HecDouble to)
```

Replace specific values for another in the time series.

The function searches through the time series array and replaces values that match the "from" value with the "to" value. **HecDouble** controls the precision to which the value is compared (e.g., 12.345 is to three places after the decimal).

Parameters: from value which will be replaced to value with which to replace with
Examples: `newDataSet = dataset.replaceSpecificValues (HecDouble from, HecDouble to)`
Returns: a copy of the current time series with the specific values replaced.

8.15.94 Reverse Rating Table Interpolation

```
reverseRatingTableInterpolation(TimeSeriesMath tsData)
```

Transform/interpolate values in the time series data set **tsData** using the reverse of the rating table stored in the current paired data set. For example, the function may be used to transform a time series of flow to a time series of stage using a stage-flow rating table. **tsData** may be a regular or irregular time interval data set. Missing values in **tsData** are kept missing in the resultant data set.

The paired data set should be created with the rating table option to set values for "datum", "shift", and "offset". By default, these values are 0.0. The shift is subtracted from, and the datum added to all input time series values. If the rating table is Log-Log, the table x-values are adjusted by subtracting the offset. Refer to the *ratingTableInterpolation()* description for comparison to this function.

Units and parameter type in resultant time series data set are defined by the x-units label and parameter type of the current paired data set. All other names and labels are copied over from the **tsData**.

See also: ratingTableInterpolation
Parameters: **tsData** - a regular or irregular interval **TimeSeriesMath** object.
Example: `tsStage = stageFlowCurve.reverseRatingTable Interpolation(tsFlow)`
Returns: A new **TimeSeriesMath** object.

8.15.95 RMS Function

```
rms(list of HecMath tsMathArray)
```

Determine the root mean square (rms, or quadraticharmonic mean) of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the rms of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series.

The new time series will always have quality defined. For a specific time, if any of the quailty values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: *tsMathArray* the array of time series to be compared with the current time series.
Example: `newDataSet = dataset.rms(list of HecMath tsMathArray)`
Returns: a new time series representing the rms of all time series.

8.15.96 Round to Nearest Whole Number

```
round()
```

Rounds values in a time series or paired data set to the nearest whole number. The function rounds up the decimal portion of a number if equal to or greater than .5 and rounds down decimal values less than .5. For example:

10.5	is rounded to 11.
10.499	is rounded to 10.

The x-values in paired data sets are unaffected by the function, only the y-value data are rounded. For time series data sets, missing values are kept missing.

For paired data sets, use the *setCurve()* method to first select the paired data curve(s).

See also: roundOff(); truncate(); setCurve().
Parameters: Takes no parameters
Example: `roundedData = dataSet.round()`
Returns: A new **HecMath** object of the same type as the current object.

8.15.97 Round Off to Specified Precision

```
roundOff(integer significantDigits, integer powerOfTensPlace)
```

Round values in a time series or paired data set to a specified number of significant digits and/or power of tens place. For the power of tens place, -1 specifies rounding to one-tenth (0.1), while +2 rounds to the hundreds (100). For example:

1234.123456 will round to:
1230.0 for number of significant digits = 3, power of tens place = -1
1234.1 for number of significant digits = 6, power of tens place = -1
1234 for number of significant digits = 6, power of tens place = 0
1230 for number of significant digits = 6, power of tens place = 1

The x-values in paired data sets are unaffected by the function, only the y-value data are rounded. For time series data sets, missing values are kept missing.

For paired data sets, use the *setCurve()* method to first select the paired data curve(s).

See also: round(); truncate(); setCurve().
Parameters:
`significantDigits` – An integer specifying the number of significant digits to use in the rounding.
`powerOfTensPlace` – An integer specifying the power of tens place to use in the rounding.
Example: `roundedData = dataSet.roundOff(5, -2)`
Returns: A new **HecMath** object of the same type as the current object.

8.15.98 Screen for Erroneous Values Based on Constant Value

```
screenWithConstantValue(string duration,
  floating-point rejectTolerance,
  floating-point questionTolerance,
  floating-point minThreshold,
  integer maxMissing)
```

Flag values in time series whose values do not change more than a specified amount over a specified duation as questionable or rejected.

Values in the time series are screened for quality. All (non-missing) values whose difference over the specified duration is below *rejectTolerance* are marked with a rejected quality flag. Other values whose difference are over the specified duration is below *questionTolerance* are marked with a questioned quality flag. Other values are marked with an okay quality flag unless they are already marked as rejected or questioned. Only values above the specified *minThreshold* value are screened as described. Only time periods that contain no more that *maxMissing* missing values are screened as described.

Parameters:
tsc - The time series to derive from.
durationStr - The time period over which to perform the test, of the form nX, where in is an integer and X is one of Minutes, Hours, or Days, which can be abbreviated to one or more characters.
rejectTolerance - Values will be marked as rejected if they do not differ by more than this amount over the specified duration. To

disable marking values as rejected, set this tolerance to less than zero.
questionTolerance - Values will be marked as questioned if they do not differ by more than this amount over the specified duration. To disable marking values as questioned, set this tolerance to less than zero.
minThreshold - Values will be screened only if they are greater than this value.
maxMissing - The maximum number of missing values to tolerate in the specified duration for screening purposes.

Example:

```
newDataSet = dataset.screenWithConstantValue(string
duration,
  floating-point rejectTolerance,
  floating-point questionTolerance,
  floating-point minThreshold,
  integer maxMissing)
```

Returns: A copy of the time series with values with the quality set as described.

8.15.99 Screen for Erroneous Values Based on Duration Magnitude

```
screenWithDurationMagnitude(string durationStr,
  floating-point minRejectLimit,
  floating-point minQuestionLimit,
  floating-point maxRejectLimit,
  floating-point maxQuestionLimit)
```

Flag values in time series whose accumulation over a specified period lies minimum/maximum limits values as questionable or rejected.

Values in the time series are screened for quality. All (non-missing) values whose accumulated value for the specified duration is below *minRejectLimit* or above *maxRejectLimit* are marked with a rejected quality flag. Other values whose accumulated value for the specified duration is below *minQuestionLimit* or above *maxQuestionLimit* are marked with a questioned quality flag. Other values are marked with an okay quality flag unless they are already marked as rejected or questioned.

Parameters:

DurationStr - The time period over which to perform the test, of the form nX, where in is an integer and X is one of Minutes, Hours, or Days, which can be abbreviated to one or more characters.
minRejectLimit - Values whose accumulated value for the specified duration is below this will be marked as rejected.
minQuestionLimit - Values whose accumulated value for the specified duration is below this, but not below

minRejectLimit - will be marked as questioned.
maxQuestionLimit - Values whose accumulated value for the specified duration is above this, but not above
maxRejectLimit - will be marked as questioned.
maxRejectLimit - Values whose accumulated value for the specified duration is above this will be marked as rejected.

Example:

```
screenedData = tsData.
screenWithDurationMagnitude(string durationStr,
  floating-point minRejectLimit,
  floating-point minQuestionLimit,
  floating-point maxRejectLimit,
  floating-point maxQuestionLimit)
```

Returns: A copy of the time series with values with the quality set as described.

8.15.100 Screen for Erroneous Values Based on Forward Moving Average

```
screenWithForwardMovingAverage(integer numberToAverageOver,
  floating-point changeLimit)
```

Flag values in time series exceeding maximum change from a forward moving average. A running forward moving average of valid values is progressively computed over *numberToAverageOver* points. Values exceeding the moving average computed for the preceeding point location by more than *changeLimit* are flagged or set to the "Missing" value depending upon the settings for *setInvalidToMissingValue* or *qualityFlagForInvalidValue* parameters. Values failing the quality test are excluded from the forward moving average computation for the subsequent points. The maximum change comparison is done only when consecutive values are not flagged.

Parameters:

numberToAverageOver - the number of values to average over for computing the forward moving average. Must be greater than two.
changeLimit - allowed deviation in the tested value from the forward moving average.

Example:

```
screenedData = tsData.screenWithForwardMovingAverage(
integer numberToAverageOver, floating-point
changeLimit)
```

Returns: a copy of the time series with values failing the quality test set to undefined and with the quality flag set to missing.

8.15.101 Screen for Erroneous Values Based on Forward Moving Average (Missing Values)

```
screenWithForwardMovingAverage(integer
numberToAverageOver,
   floating-point changeLimit,
   boolean setInvalidToMissingValue,
   string qualityFlagForInvalidValue)
```

Screen the current time series data set for possible erroneous values based on the deviation from the forward moving average over *numberToAverageOver* values computed at the previous point. If the deviation from the moving average is greater than changeLimit, the value fails the screening test. Data values failing the screening test are assigned a quality flag and/or are set to missing.

Missing values and values failing the screening test are not counted in the moving average and the divisor of the average is less one for each such value. At least two values must be defined in the moving average else the moving average is undefined and value being examined is screened acceptable.

If *setInvalidToMissingValue* is true, values failing the screening test are set to missing.

If *qualityFlagForInvalidValue* is set to a character or string recognized as a valid quality flag, the quality flag will be set for tested values. If there is no previously existing quality available for the time series, the quality flag array will be created for the time series. Values failing the quality test are set to the user specified quality flag for invalid values. If there is existing quality data and the time series value passes the quality test, the existing quality flag for the points is unchanged. If there was no previously existing quality and the time series value passes the quality test, the quality flag for the point is set to "Okay".

The acceptable values for *qualityFlagForInvalidValue* strings are: "M" or "Missing”, "R" or "Rejected", "Q" or "Questionable". A blank string (" ") is entered to disable the setting of the quality flag.

For the example:

```
resultantDataSet = dataSet.screenWithForwardMovingAverage
(16, 100., TRUE,  "R")
```

the forward moving average will be computed over 16 values, values deviating from the moving average by more than 100.0 will be set to missing and flagged as rejected.

Parameters:

`numberToAverageOver` – An integer specifying the number of averaging values. Must be at least two.

`changeLimit` – A floating-point number specifying the maximum change allowed in the tested value from the forward moving average value.

`setInvalidToMissingValue` – Either True or False, specifying whether time series values failing the screening test are set to the "Missing" value.

`qualityFlagForInvalidValue` - A string representing the quality flag setting for values failing the screening test. The accepted character strings are: "M" or "Missing", "R" or "Rejected", "Q" or "Questionable". An empty string (" ") is entered to disable the setting of the quality flag.

Example:

```
screenedData = tsData.screenWithForwardMovingAverage
  (16, 100., TRUE, “R”)
```

Returns: A new TimeSeriesMath object.

Generated Exceptions: Throws a *HecMathException* if *numberToAverageOver* is less than two; if an unrecognized quality flag is entered for *qualityFlagForInvalidValue* or if *setInvalidToMissing Value* is false and *qualityFlagForInvalidValue* is blank (no action would occur).

8.15.102 Screen for Erroneous Values Based on Maximum/Minimum Range (Missing Values)

```
screenWithMaxMin(floating-point minValueLimit,
  floating-point maxValueLimit,
  floating-point changeLimit,
  boolean setInvalidToMissingValue,
  string qualityFlagForInvalidValue)
```

Flag values in a time series data set exceeding minimum and maximum limit values or maximum change limit.

Values in the time series are screened for quality. Values below *minValueLimit* or above *maxValueLimit* or with a change from the previous time series value greater than *changeLimit* fail the screening test. The maximum change comparison is done only when consecutive values are not flagged.

If *setInvalidToMissingValue* is set to true, values failing the screening test are set to the "Missing" value.

If *qualityFlagForInvalidValue* is set to a character or string recognized as a valid quality flag, the quality flag will be set for tested values. If there is no previously existing quality available for the time series, the quality flag array will be created for the time series. Values failing the quality test are set to the user specified quality flag for invalid values. If there is existing quality data and the time series value passes the quality test, the existing quality flag for the points is unchanged. If there was no previously existing quality and the time series value passes the quality test, the quality flag is set to "Okay".

For example:

```
resultantDataSet = dataSet.screenWithMaxMin (0.0, 1000.,
100., FALSE, “R”)
```

time series values less than 0.0, or greater than 1000., or with a change from a previous point greater than 100 will be flagged as "Rejected". Flagged points however will not be set to the "Missing" value.

Parameters:

`minValueLimit` – A floating-point number specifying the minimum valid value limit.

`maxValueLimit` - A floating-point number specifying the maximum valid value limit.

`changeLimit` - A floating-point number specifying the maximum change allowed in the tested value from the previous time series value.

`setInvalidToMissingValue` - Either True, or False, specifying whether time series values failing the screening test are set to the "Missing" value.

`qualityFlagForInvalidValue` - A string representing the quality flag setting for values failing the screening test. The accepted character strings are: "M" or "Missing", "R" or "Rejected", "Q" or "Questionable". An empty string (" ") is entered to disable the setting of the quality flag.

Example:

```
screenedData = tsData.screenWithMaxMin(0., 1000., 100.,
FALSE, “R”)
```

Returns: A new **TimeSeriesMath** object.

Generated Exceptions: Throws a *HecMathException* if an unrecognized quality flag is entered for **`qualityFlagForInvalidValue`** or if **`setInvalidToMissingValue`** is false and **`qualityFlagForInvalidValue`** is blank (no action would occur).

8.15.103 Screen for Erroneous Values Based on Maximum/Minimum Range

```
screenWithMaxMin(floating-point minValueLimit,
  floating-point maxValueLimit,
  floating-point changeLimit)
```

Flag values in time series exceeding minimum and maximum limit values or maximum change limit.

Values in the time series are screened for quality. Values below *minValueLimit* or above *maxValueLimit* or with a change from the previous time series value greater than *changeLimit* fail the quality test. The maximum change comparison is done only when consecutive values are not flagged.

Values failing the screening test are set to the "Missing" value.

Parameters:
minValueLimit - minimum valid value limit.
maxValueLimit - maximum valid value limit.
changeLimit - maximum change allowed in the tested value from the previous time series value.

Example: `screenedData = screenWithMaxMin(floating-point minValueLimit, floating-point maxValueLimit, floating-point changeLimit)`

Returns: a copy of the time series with values failing the quality test set to missing.

8.15.104 Screen for Erroneous Values Based on Maximum/Minimum Range (Quality)

```
screenWithMaxMin(floating-point minValueLimit,
  floating-point maxValueLimit,
  floating-point changeLimit,
  boolean setInvalidToMissingValue,
  floating-point invalidValueReplacement,
  string qualityFlagForInvalidValue)
```

Flag values in time series exceeding minimum and maximum limit values or maximum change limit.

Values in the time series are screened for quality. Values below *minValueLimit* or above *maxValueLimit* or with a change from the previous time series value greater than *changeLimit* fail the quality test. The maximum change comparison is done only when consecutive values are not flagged.

Parameters:

minValueLimit - minimum valid value limit.
maxValueLimit - maximum valid value limit.
changeLimit - maximum change allowed in the tested value from the previous time series value.
setInvalidValueToSpecified - if true, time series values failing the quality test are set to the specified value.
invalidValueReplacement - The value to use for replacing invalid values is setInvalidToSpecifiedValue is set to "true".
qualityFlagForInvalidValue - character string representing the quality flag setting for values failing the quality test. The accepted character strings are: "M" or "Missing", "R" or "Rejected", "Q" or "Questionable".

Example:
```
screenedData = tsData.screenWithMaxMin(floating-
point minValueLimit, floating-point maxValueLimit,
  floating-point changeLimit,
  boolean setInvalidToMissingValue,
  floating-point invalidValueReplacement,
  string qualityFlagForInvalidValue)
```

Returns: a copy of the time series with values failing the quality test set to missing.

8.15.105 Screen for Erroneous Values Based on Maximum/Minimum Range (Limits)

```
screenWithMaxMin(floating-point minRejectLimit,
  floating-point minQuestionLimit,
  floating-point maxQuestionLimit,
  floating-point maxRejectLimit)
```

Flag values in time series exceeding minimum and maximum limit values as questionable or rejected. Values in the time series are screened for quality. Values below *minRejectLimit* or above *maxRejectLimit* are marked with a rejected quality flag. Other values below *minQuestionLimit* or above *maxQuestionLimit* are marked with a questioned quality flag. Other values are marked with an okay quality flag unless they are already marked as rejected or questioned.

Parameters:

minRejectLimit - Values below this will be marked as rejected.
minQuestionLimit - Values below this, but not below minRejectLimit will be marked as questioned.
maxQuestionLimit - Values above this, but not above maxRejectLimit will be marked as questioned.
maxRejectLimit - Values above this will be marked as rejected.

Example:
```
screenedData = tsData.screenWithMaxMin(floating-
point minRejectLimit,
  floating-point minQuestionLimit,
  floating-point maxQuestionLimit,
  floating-point maxRejectLimit)
```

Returns: A copy of the time series with values with the quality set as described.

8.15.106 Screen for Erroneous Values Based on Rate of Change

```
screenWithRateOfChange(floating-point minRejectLimit,
  floating-point minQuestionLimit,
  floating-point maxQuestionLimit,
  floating-point maxRejectLimit)
```

Flag values in time series whose rate of change from the last valid value exceed minimum/maximum limits values as questionable or rejected.

Values in the time series are screened for quality. Values whose rate of change from the last valid value is below *minRejectLimit* or above *maxRejectLimit* are marked with a rejected quality flag. Other whose rates of change from the last valid value are values below *minQuestionLimit* or above *maxQuestionLimit* is marked with a questioned quality flag. Other values are marked with an okay quality flag unless they are already marked as rejected or questioned.

Rates are computed in units per hour.

Parameters:

minRejectLimit - Values whose rate of change from the last valid value are below this will be marked as rejected.
minQuestionLimit - Values whose rate of change from the last valid value are below this, but not below
minRejectLimit - will be marked as questioned.
maxQuestionLimit - Values whose rate of change from the last valid value are above this, but not above maxRejectLimit will be marked as questioned.
maxRejectLimit - Values whose rate of change from the last valid value are above this will be marked as rejected.

Example:
```
newDataSet = dataset.screenWithRateOfChange(
floating-point minRejectLimit,
  floating-point minQuestionLimit,
  floating-point maxQuestionLimit,
  floating-point maxRejectLimit)
```

Returns: A copy of the time series with values with the quality set as described.

8.15.107 Select a Paired Data Curve by Curve Label

```
setCurve(string curveName)
```

Select, by curve label, the paired data curve for performing subsequent arithmetic operations or math functions. By default, a paired data set loaded from file has all curves selected.

A paired data set may contain more than one set of y-values. However, a user may wish to modify only one curve of the data set. For example, using the function ".add(2.0)" would by default add 2.0 to all y-values for all curves. The *setCurve()* call may be used to limit the operation to just one selected set of y-values. The function searches the paired data set list of curve labels for a match to *curveName*. If a match is found, that curve is set as the selected curve.

See also: setCurve(integer curveNumber)
Example: `damageCurve.setCurve(“RESIDENTIAL”)`
Parameters: curveName – The curve label (a string) to set as the selected curve.
Returns: Nothing
Generated Exceptions: Throws a **HecMathException** – if *curveName* is not found in the paired data set curve labels.

8.15.108 Select a Paired Data Curve by Curve Number

```
setCurve(integer curveNumber)
```

Select, by curve number, the paired data curve for performing subsequent arithmetic operations or math functions. By default, a paired data set loaded from file has all curves selected.

A paired data set may contain more than one set of y-values. However, a user may wish to modify only one curve of the data set. For example, using the function ".add(2.0)" would by default add 2.0 to all y-values for all curves. The *setCurve()* call can be used to limit the operation to just one selected set of y-values. The function sets a curve index internal to the paired data set. The option is to select one curve or all curves.

Curve numbering begins with "0". If a paired data set has two curves, the first curve is selected by, "setCurve(0)". To select the second curve, use "setCurve(1)".

All curves in a paired data set are selected by setting *curveNumber* to -1.

See also: setCurve(String curveName)

Parameters: curveNumber – An integer specifying the curve to set as the selected curve. Curve numbering begins with 0. Set to –1 to select all curves.
Example: `ruleCurve.setCurve(-1)`
Returns: Nothing

8.15.109 Set Data Container

```
setData(hec.io.DataContainer container)
```

Sets the data container for the current data set. For time series data sets, this is a *hec.io.TimeSeriesContaine*r. For paired data sets, container should be a *hec.io.PairedDataContainer*. Containers are generated by some of the other functions.

The *hec.io.DataContainer* class and the *hec.io.TimeSeriesContainer* and the *hec.io.PairedDataContainer* subclasses contain the time series and paired data values.

Parameters: container – A hec.io.TimeSeriesContainer for time series data sets, or a hec.io.PairedDataContainer for paired data sets.
Example: `dataSet.setData(TSContainer)`
Returns: Nothing
Generated Exceptions: Throws a **HecMathException** if container is not of type *hec.io.TimeSeriesContainer* for time series data sets or not of type *hec.io.PairedDataContainer* for paired data sets.

8.15.110 Set Location Name for Data Set

```
setLocation(String locationName)
```

Set the location name for a data set, which changes the B-Part of the HEC-DSS pathname. The new pathname will be used in plots, tables, and in the *write()* method of DSSFile objects.

Parameters: locationName – A string specifying the new location name for the data set.
Example: `dataSet.setLocation(“OAKVILLE”)`
Returns: Nothing

8.15.111 Set Parameter for Data Set

```
setParameterPart(String parameterName)
```

Set the parameter name for a data set, which changes the C-Part of the HEC-DSS pathname. The new pathname will be used in plots, tables, and in the *write()* method of DSSFile objects.

Parameters: parameterName – A string specifying the new parameter name for the data set.
Example: `dataSet.setParameterPart(“ELEV”)`
Returns: Nothing

8.15.112 Set Pathname for Data Set

```
setPathname(String pathname)
```

Set the pathname for a data set to the specified pathname. Subsequent operations using the data set such as *getData()* or *DSSFile.write()* will use or reflect the new pathname.

Parameters: pathname – A string specifying the new pathname for the data set.
Example: `dataSet.setPathname(“//OAKVILLE/STAGE//1HOUR/OBS/”)`
Returns: Nothing

8.15.113 Set Time Interval for Data Set

```
setTimeInterval(String interval)
```

Set the time interval for a data set, which changes the E-Part of the pathname. The new pathname will be used in plots, tables, and in the *write()* method of DSSFile objects.

Parameters: interval – A string specifying the new interval for the data set.
Example: `dataSet.setTimeInterval(“1HOUR”)`
Returns: Nothing

8.15.114 Set Data Type for Time Series Data Set

```
setType(string typeString)
```

Set the data for a time series data set.

Parameters: typeString – A string specifying the data type for the data set. This should be "INST-CUM", "INST-VAL", "PER-AVER" or "PER-CUM"

Example: `dataSet.setType(“PER-AVER”)`
Returns: Nothing

8.15.115 Set Units Label for Data Set

```
setUnits(String unitsString)
```

Set the units label for a data set. For a paired data set, the call sets the y-units label.

Parameters: unitsString – A string specifying the units label for the data set.
Example: `dataSet.setUnits(“CFS”)`
Returns: Nothing

8.15.116 Set Version Name for Data Set

```
setVersion(String versionName)
```

Set the version name for a data set, which changes the F-Part of the pathname. The new pathname will be used in plots, tables, and in the *write()* method of DSSFile objects.

Parameters: version – A string specifying the new location for the data set.
Example: `dataSet.setVersion(“OBSERVED”)`
Returns: Nothing

8.15.117 Set Watershed Name for Data Set

```
setWatershed(String watershedName)
```

Set the watershed (or river) name for a data set, which changes the A-Part of the pathname. The new pathname will be used in plots, tables, and in the *write()* method of DSSFile objects.

Parameters: watershedName – A string specifying the new watershed name for the data set.
Example: `dataSet.setWatershed(“OAK RIVER”)`
Returns: Nothing.

8.15.118 Shift Adjustment of Time Series Data

```
shiftAdjustment(TimeSeriesMath tsData)
```

Derive a new time series data set by linear interpolation of values in the current time series data set at the times defined by the time series data set **tsData**. If times in the new time series precede the first data point in the current time series, the value for these times is set to 0.0. If times in the new time series occur after the last data point in the current time series, the value for these times is set to the value of the last point in the current time series. Interpolation of values with the *shiftAdjustment* function is shown in Figure 8.14.

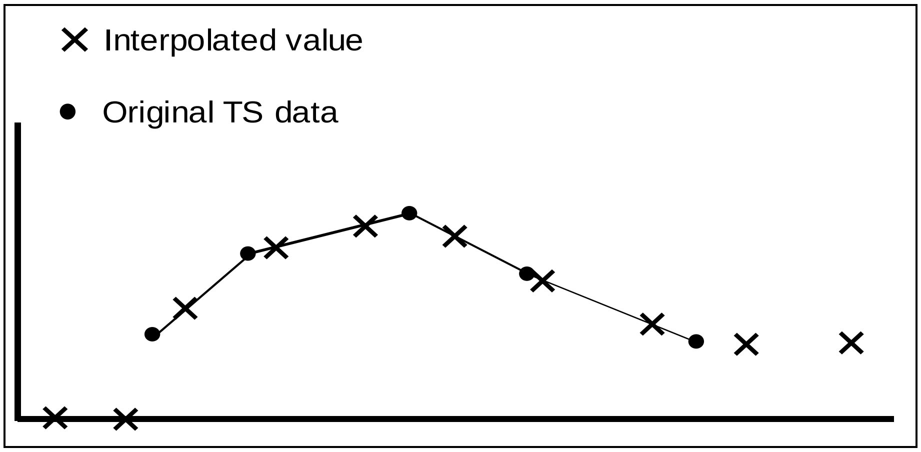


Figure 8.14 Interpolation of Time Series Values using Shift Adjustment Function

Both time series data sets may be regular or irregular interval. Interpolated points must be bracketed or coincident with valid (not missing) values in the original time series; otherwise the values are set as missing.

Parameters: tsData – A regular or irregular interval time series data set.
Example: `tsInterp = tsValues.shiftAdjustment(tsData)`
Returns: A new **TimeSeriesMath** object.

8.15.119 Shift Time Series in Time

```
shiftInTime(string timeShiftString)
```

Shift the times in the current time series data set by the amount specified with *timeShiftString*. The data set may be regular or irregular interval time series data. Data set values are unchanged.

timeShiftString has the form "nT", where "n" is an integer number and "T" is "M"(inute), "H"(our), "D"(ay), "W"(eek), "Mo"(nth),or "Y"(ear).

Only the first character is significant for "T", except for "Month", which requires at least two characters.

Parameters: *timeShiftString* – A string specifying the time increment to shift times in the current time series data set.
Example: `TsShifted = tsData.shiftInTime("3H")`
Returns: A new **TimeSeriesMath** object.

8.15.120 Sign Function

```
sign()
```

Derive a new time series composed of the signs of the values of the current time series. Values which are missing in the original time series will be missing in the new time series. Values in the current time series will be represented as -1.0, 0.0 or 1.0 in the derived time series, depending on whether the original value is &lt; 0.0, 0.0 or &gt; 0.0.

Parameters: Takes no Parameters
Example: `newDataSet = dataSet.sign()`
Returns: A new time series composed of the negatives of the values of the current time series

8.15.121 Sine Trigonometric Function

```
sin()
```

Derive a new time series or paired data set from the sine of values of the current data set. The resultant data set values are in radians. For time series data, missing values are kept as missing.

For paired data sets, use the *setCurveMethod* to first select the paired data curve(s).

See also: setCurve()
Parameters: Takes no parameters
Example: `newDataSet = dataSet.sin()`
Returns: A new **HecMath** object of the same type as the current object

8.15.122 Skew Coefficient

```
skewCoefficient()
```

Compute the skew coefficient of the current time series data set. Missing values are ignored.

Parameters: Takes no parameters
Example: `skewCoefficient = dataSet.skewCoefficient()`
Returns: A floating-point value representing the skew coefficient of the current time series.

8.15.123 Snap Irregular Times to Nearest Regular Period

```
snapToRegularInterval(string timeIntervalString,
  string timeOffsetString,
  string timeBackwardString,
  string timeForwardString)
```

"Snap" data from the current irregular or regular interval time series to form a new regular interval time series of the specified interval and offset. For example, a time series record from a gauge recorder collects readings 6 minutes past the hour. The function may be used to “snap” or shift the time points to the top of the hour.

The regular interval time of the resultant time series is specified by *timeIntervalString*. *timeIntervalString* is a valid HEC time increment string (e.g. "1MIN", "15MIN", "1HOUR", "6HOUR", "1DAY", "1MONTH").

Times in the resultant time series may be shifted (offset) from the regular interval time by the increment specified by *timeOffsetString*. As an example, the offset could be used to shift times from the top of the hour to instead six minutes past the hour. Data from the original time series is "snapped" to the regular interval if the time of the data falls within the time window set by the *timeBackwardString* and the *timeForwardString*. That is, if the new regular interval is at the top of the hour and the time window extends to nine minutes before the hour and fifteen minutes after the hour, an original data point at 0852 would be snapped to the time 0900 while a point at 0916 would be ignored.

timeOffsetString, *timeBackwardString* and *timeForwardString* are time increment strings expressed as "nT", where "n" is an integer number and "T" is one of the time increments: "M(INUTES)", "D(AYS)" or "H(OUR)" (characters in the parenthesis are optional). For the example of the previous paragraph, *timeIntervalString* would be "1HOUR", *timeOffsetString* would be "0M", *timeBackwardString* would be "9M" (or "9min") and *timeForwardString* would be "15M.” A blank string (" ") is equivalent to "0M".

By default values in the resultant regular interval time series data set are set to missing unless matched to times in the current time series data set within the time window tolerance set by *timeBackwardString* and *timeForwardString*.

Parameters:

`timeIntervalString` – A string specifying the regular time interval for the resultant time series.

`timeOffsetString` – A string specifying the offset of the new time points from the regular interval time. This variable may be an empty string (" ") or None.

`timeBackwardString` – A string specifying the time to look backwards from the regular time interval for valid time points.

`timeForwardString` – A string specifying the time to look forward from the regular time interval for valid time points.

Example: `rtsData = itsData.snapToRegularInterval(“1HOUR”, None, “5Min”, “5Min”)`

Returns: A new regular interval **TimeSeriesMath** object.

8.15.124 Square Root

```
sqrt()
```

Derive a new time series or paired data set computed from the square root of values of the current data set. For time series data, missing values are kept as missing. Values less than zero are set to missing.

For paired data sets, use *setCurve* to first select the paired data curve(s).

See also: setCurve()

Parameters: Takes no parameters

Example: `newDataSet = dataSet.sqrt()`

Returns: A new **HecMath** object of the same type as the current object.

8.15.125 Standard Deviation of Time Series

```
standardDeviation()
```

Compute the standard deviation value of the current time series data set. Missing values are ignored.

Parameters: Takes no parameters

Example: `stdDev = tsData.standardDeviation()`

Returns: A floating-point value representing the standard deviation of the current time series

8.15.126 Standard Deviation of Time Series (*tsMathArray*)

```
standardDeviation(list of HecMath tsMathArray)
```

Determine the standard deviation of the current time series and the each time series in the parameter, *tsMathArray*. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the standard deviation of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series.

The new time series will always have quality defined. For a specific time, if any of the quailty values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: *tsMathArray* - the array of time series to be compared with the current time series.
Example: `stdDev = tsData.standardDeviation(list of HecMath tsMathArray)`
Returns: a new time series representing the standard deviation of all time series.

8.15.127 Straddle Stagger Hydrologic Routing

```
straddleStaggerRouting(integer numberToAverage,
  integer numberToLag,
  integer numberSubreaches)
```

Route the current regular interval time series data set using the Straddle-Stagger hydrologic routing method. *numberToAverage* specifies the number of ordinates to average over (Straddle). *numberToLag* specifies the number ordinates to lag (Stagger). The number of routing subreaches is set by *numberSubreaches*.

Parameters:
numberToAverage – An integer specifying the number of ordinates to average over (Straddle).
numberToLag – An integer specifying the number of ordinates to lag (Stagger).

numberSubreaches – An integer specifying the number of routing subreaches.

Example: `tsRouted = tsFlow.straddleStaggerRouting (numberAver, lag, reachCount)`
Returns: A new **TimeSeriesMath** object.

8.15.128 Subtract a Constant

```
subtract(floating-point constant)
```

Subtract the value constant from all valid values in the current time series or paired data set. For time series data, missing values are kept as missing.

For paired data, constant is subtracted from y-values only. Use the *setCurve* method to first select the paired data curve(s).

See also: subtract(HecMath hecMath); setCurve()
Parameters: constant - A floating-point value.
Example: `newDataSet = dataSet.subtract(5.3)`
Returns: A new **HecMath** object of the same type as the current object.

8.15.129 Subtract a Data Set

```
subtract(TimeSeriesMath tsData)
```

Subtract the values in the data set tsData from the values in the current data set. Both data sets must be time series data set.

When subtracting one time series data set from another, there is no restriction that times in the two data sets match exactly. However, only values with coincident times will be subtracted. Times in the current time series data set that cannot be matched with times in the second data set are set missing. Values in the current data set that are missing are kept as missing. Either or both data sets may be regular or irregular interval time series.

See also: subtract(floating-point constant)
Parameters: tsData - a **TimeSeriesMath** object.
Example: `newDataSet = dataSet.subtract(otherDataSet)`
Returns: A new **TimeSeriesMath** object.

8.15.130 Successive Differences for Time Series

```
successiveDifferences()
```

Derive a new time series from the difference between successive values in the current regular or irregular interval time series data set. The current data must be of type "INST-VAL" or "INST-CUM". A value in the resultant time series is set to missing if either the current or previous value in the current time series is missing (need to have two consecutive valid values). If the data type of the current data set is "INST-CUM" the resultant time series data set is assigned the type "PER-CUM", otherwise the data type does not change.

See also: timeDirivative()
Parameters: Takes no parameters
Example: `newTsData = tsData.successiveDifferences()`
Returns: A new **TimeSeriesMath** object.
Generated Exceptions: Throws a **HecMathException** if the current data set is not of type "INST-VAL" or "INST-CUM".

8.15.131 Sum Values in Time Series

```
sum()
```

Sum the values of the current time series data set. Missing values are ignored.

Parameters: Takes no parameters
Example: `total = tsData.sum()`
Returns: A floating-point value representing the sum of all valid values of the current time series.

8.15.132 Sum Values in Time Series (*tsMathArray*)

```
sum(list of HecMath tsMathArray)
```

Add each of the time series in the parameter, *tsMathArray*, to the current time series. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current time series and each time series in *tsMathArray*, the values will be summed provided values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series.

The new time series will always have quality defined. For a specific time, if any of the quality values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: *tsMathArray* - the array of time series to be added to the current time series.
Example: `tsData.sum(list of HecMath tsMathArray)`
Returns: a new time series representing the sum of the two time series

8.15.133 Tangent Trigonometric Function

`tan()`

Derive a time series or paired data set computed from the tangent of values of the current data set. For time series data, missing values are kept as missing. If the cosine of the current time series value is zero, the value is set missing.

For paired data sets, use the *setCurve* method to first select the curve(s).

See also: setCurve()
Example: `newDataSet = dataSet.tan()`
Parameters: Takes no parameters
Returns: A new **HecMath** object of the same type as the current object.

8.15.134 Time Derivative (Difference per Unit Time)

`timeDerivative()`

Derive a new time series data set from the successive differences per unit time of the current regular or irregular interval time series data set. For the time “t”,

$$TS2(t) = (TS1(t) – TS1(t-1)) / DT$$

where DT is the time difference between t and t-1. For the current form of the function, the units of DT are minutes.

A value in the resultant time series is set to missing if either the current or previous value in the original time series is missing (need to have two consecutive valid values). By default, the data type of the resultant time series data set is assigned as "PER-AVER".

See also: successiveDifferences()
Parameters: Takes no parameters
Example: `newTsData = tsData.timeDerivative()`
Returns: A new **TimeSeriesMath** object

8.15.135 Transform Time Series to Regular Interval

```
transformTimeSeries(string timeIntervalString,
  string timeOffsetString,
  string functionTypeString)
```

Generate a new regular interval time series data set from the current regular or irregular time series. The new time series is computed having the regular time interval specified by *timeIntervalString* and time offset set by *timeOffsetString*.

Values for the new time series are computed from the original time series data set using one of seven available functions. The function is selected by setting *functionTypeString* to one of the following types:

"INT"	-	Interpolate at end of interval
"MAX"	-	Maximum over interval
"MIN"	-	Minimum over interval
"AVE"	-	Average over interval
"ACC"	-	Accumulation over interval
"ITG"	-	Integration over interval
"NUM"	-	Number of valid data over interval

where "interval" is the interval between time points in the new time series.

The regular interval time of the new time series is specified by *timeIntervalString*. *timeIntervalString* is a valid HEC time increment string (e.g. “"1MIN", "15MIN", "1HOUR", "6HOUR", "1DAY", "1MONTH").

Times in the resultant time series may be shifted (offset) from the regular interval time by the increment specified by *timeOffsetString*. As an example, the offset could be used to shift times from the top of the hour to six minutes past the hour. Typically no offset is used.

The data type of the original time series data governs how values are interpolated. Data type "INST-VAL" (or "INST-CUM") considers the value to change linearly over the interval from the previous data value to the current data value. Data type "PER-AVER" considers the value to be constant at the current data value over the interval. Data type "PER-CUM" considers the value to increase from 0.0 (at the start of the interval) up to the current value over the interval. Interpolation of the three data types is illustrated in Figure 8.15.

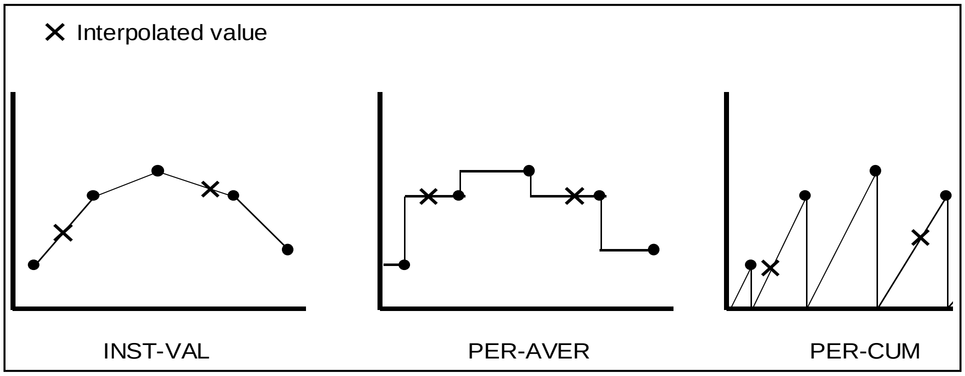


Figure 8.15 Interpolation of "INST-VAL", "PER-AVER" and "PER-CUM" Data

How interpolation is performed for a specific data type influences the computation of new time series values for the selected function. For example, if the data type is "INST-VAL", the function "Maximum over interval" is evaluated by: Finding the maximum value of the data points from the original time series that are inclusive in the new time interval. Linearly interpolate values at beginning and ending of the new time interval, and determine if these values represent the maximum over the interval.

Referring to the plots in Figure 8.15, the "Average over interval" function is applied to a time series by integrating the area under the curve between interpolated points and dividing the result by the interval time.

See also: `transformTimeSeries( TimeSeriesMath tsData, string functionTypeString)`

Parameters:

`timeIntervalString` – A string specifying the regular time interval for the resultant time series.

`timeOffsetString` – A string specifying the offset of the new time points from the regular interval time. This variable may be a blank string (" ").

`functionTypeString` – A string specifying the method for computing values for the new time series data set.

Example: `newTsData = tsData.transformTimeSeries(“1Day”, “0M”, “AVE”)`

Returns: A new regular interval **TimeSeriesMath** object.

8.15.136 Transform Time Series to Irregular Interval

`transformTimeSeries``(TimeSeriesMath tsData,string functionTypeString)`

Generate a new time series data set from the current regular or irregular time series. The times for the new data set are defined by the times in **tsData**, which may be a regular or irregular time series data set.

Values for the new time series are computed from the original time series data set using one of seven available functions. The function is selected by setting *functionTypeString* to one of the following types:

"INT"	-	Interpolate at end of interval
"MAX"	-	Maximum over interval
"MIN"	-	Minimum over interval
"AVE"	-	Average over interval
"ACC"	-	Accumulation over interval
"ITG"	-	Integration over interval
"NUM"	-	Number of valid data over interval

where "interval" is the interval between time points in the new time series.

The data type of the original time series data governs how values are interpolated. Data type "INST-VAL" (or "INST-CUM") considers the value to change linearly over the interval from the previous data value to the current value. Data type "PER-AVER" considers the value to be constant at the current value over the interval. Data type "PER-CUM" considers the value to increase from 0.0 (at the start of the interval) up to the current value over the interval. Interpolation of the three data types is illustrated in Figure 8.15 (page 8-156).

How interpolation is performed for a specific data type influences the computation of new time series values for the selected function. For example, if the data type is "INST-VAL", the function "Maximum over interval" is evaluated by: Finding the maximum value of the data points from the original time series that are inclusive in the new time interval. Linearly interpolate values at beginning and ending of the new time interval, and determine if these values represent the maximum over the interval.

Referring to the plots in Figure 8.15 (page 8-156), the "Average over interval" function is applied to a time series by integrating the area under the curve between interpolated points and dividing the result by the interval time.

See also: transformTimeSeries(string timeIntervalString, string timeOffsetString, string functionTypeString)

Parameters:

`tsMath` – A TimeSeriesMath object used to define the times for the new data set.

`functionTypeString` – A String specifying the method for computing values for the new time series data set.

Example: `newTsData = tsValues.transformTimeSeries (tsTimeTemplate, “MAX”)`
Returns: A new **TimeSeriesMath** object

8.15.137 Truncate to Whole Numbers

```
truncate()
```

Truncates values in a time series or paired data set to the nearest whole number. For example:

10.99 is truncated to 10.

The x-values in paired data sets are unaffected by the function, only the y-value data are truncated. Missing values remain missing.

For paired data, use the setCurve method to first select the curve(s).

See also: setCurve()
Parameters: Takes no parameters
Example: `newDataSet = dataSet.truncate()`
Returns: A new **HecMath** object of the same type as the current object

8.15.138 Two Variable Rating Table Interpolation

```
twoVariableRatingTableInterpolation(TimeSeriesMath
tsDataX,TimeSeriesMath tsDataZ)
```

Derive a new time series data set by using the x-y curves in the current paired data set to perform two-variable rating table interpolation of the time series *tsDataX* and *tsDataZ*. For two-variable rating table interpolation, the current paired data set should have more than one curve (multiple sets of y-values).

As an example, reservoir release is a function of both the gate opening height and reservoir elevation Figure 8.16 (page 8-159). For each gate opening height, there is a reservoir elevation-reservoir release curve, where reservoir elevation is the independent variable (x-values) and reservoir release the dependent variable (y-values) of a paired data set. Each paired data curve has a curve label. In this case, the curve label is assigned the gate opening height. Using the paired data set shown in Figure 8.16 (page 8-159), the function may be employed to interpolate time series values of reservoir elevation (*tsDataX*) and gate opening height (*tsDataZ*) to develop a time series of reservoir release.

No extrapolation is performed. If time series values from *tsDataX* or *tsDataZ* are outside the range bounded by the paired data, the new time series value is set to missing. Units and parameter type in the new time series are set to the y-units label and parameter of the current paired data set. All other names and labels are copied over from *tsDataX*.

Times for *tsDataX* and *tsDataZ* must match. Curve labels must be set for curves in the rating table paired data set and must be interpretable as numeric values.

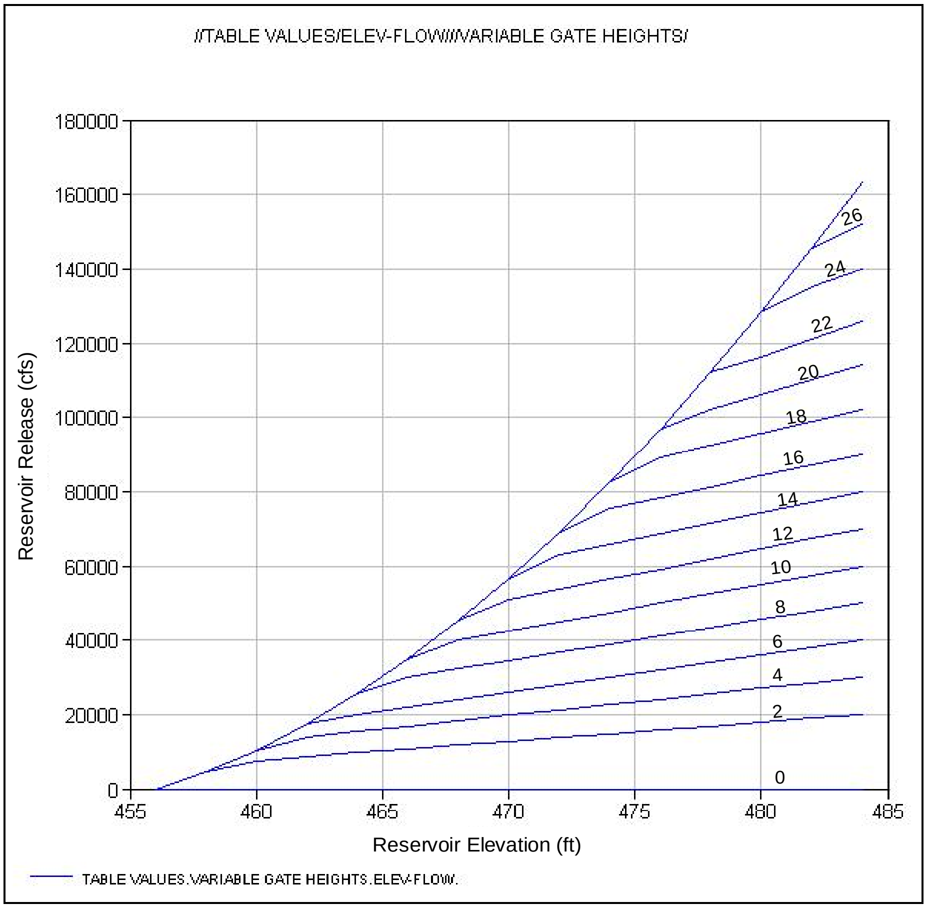


Figure 8.16 Example of two variable rating table paired data, reservoir release as a function of reservoir elevation and gate opening height (curve labels)

Parameters:

`tsDataX` – A regular or irregular interval **TimeSeriesMath** object, interpreted as x-ordinate values in the two variable interpolation.

`tsDataZ` – A regular or irregular interval **TimeSeriesMath** object, interpreted as z-ordinate values, (value defined by the paired data curve labels).

Example: `tsOutflow = gateCurve.twoVariableRatingTable Interpolation( tsElevation, tsGateOpening)`

Returns: A new **TimeSeriesMath** object.

Generated Exceptions: Throws a *HecMathException* if times do not match for *tsData***X** and *tsDataZ*; if the paired data curve labels are blank or cannot be interpreted as number values.

8.15.139 Variance Function

variance(list of HecMath tsMathArray)

Determine the variance of the current time series and the each time series in the parameter, *tsMathArra*y. A new time series will be created which duplicates the time points of the current time series. Where time points match for the current and *tsMathArray*, the values will be the variance of all time series for that time, provided the values for all time series are valid values (not missing). Points in the current time series which cannot be matched to valid points in *tsMathArray* are set to missing. Values in the current time series which are missing remain missing in the new time series.

The new time series will always have quality defined. For a specific time, if any of the quailty values in the current or parameter time series is questionable or rejected, the quality value for that time in the new time series will be set to the most severe quality for that time (e.g. if questionable and rejected are both encountered, the new quality will be set to rejected.)

Parameters: *tsMathArray* - the array of time series to be compared with the current time series.

Example:
```
newDataSet =
  dataSet.variance([ts1,ts2,ts3,ts4,ts5])
```

Returns: a new time series representing the variance of all time series.

Appendix A

Additional Plug-Ins

A.1 Introduction

This appendix documents additional plug-ins that are provided with HEC-DSSVue. A "plug-in" is an executable "*.jar*" file that is saved in the "**Plugins**" directory that can access and perform functions on top of HEC-DSSVue. No modifications are needed to the original program to provide this additional functionality. Access and use of a plug-in should appear seamless to a user; generally their functionality is accessed through a standard menu.

Plug-ins can be updated without changing the original program. The HEC-DSSVue web site will provide updates for HEC plug-ins. Other agencies may provide their own plug-ins to meet custom needs. The HEC-DSSVue plug-in web site is:

http://www.hec.usace.army.mil/software/hec-dss/plug-ins.htm

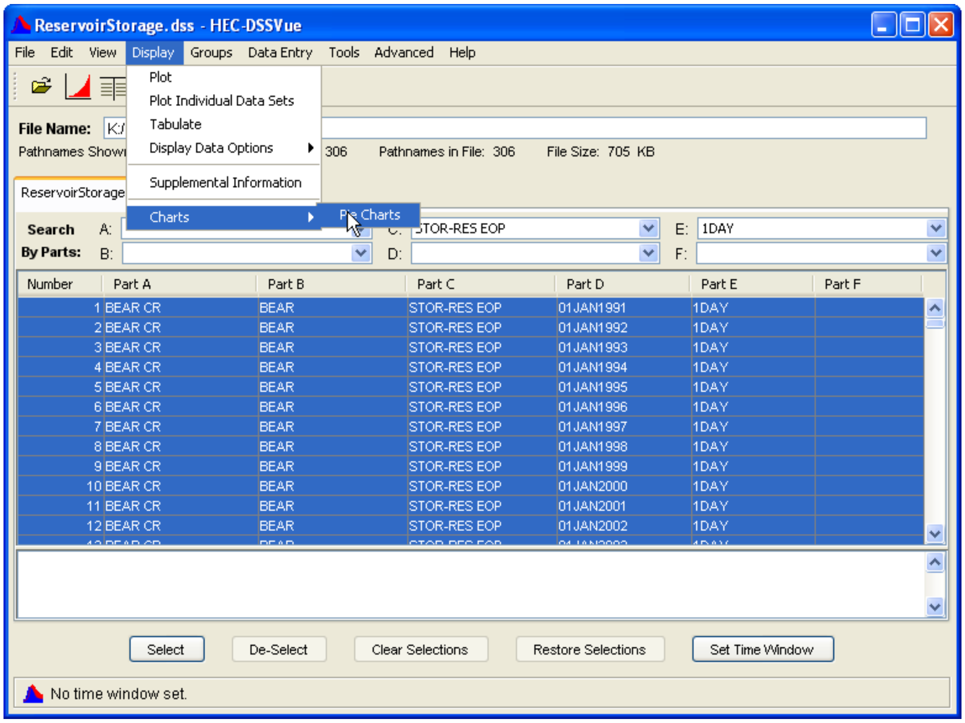
A.2 Pie Charts

The **Pie Chart** plug-in for HEC-DSSVue allows you to plot a pie chart of any value of interest (e.g., flow, stage, or reservoir storage) for a single date and time at multiple locations. Either a two-dimensional (2D) or three-dimensional (3D) plot can be created. The pie chart is plotted using a default color sequence, but you can change the color of each individual slice. The transparency of the chart can also be adjusted. By default, all sections are labeled with the site name, units, and percentage of the total. The unit and percent labels can be individually turned off, or section labels can be turned off altogether. A legend is created below the chart by default, which can also be turned off. The section labels and legend are updated dynamically. The final product can be printed, or it can be saved to a graphics file to be included in presentations, documents, or web pages.

A.2.1 Creating a Pie Chart

First, load a data set into DSSVue. Select the data type to plot by clicking the drop-down list to select the C part of the DSS path. Select the time interval using the drop-down list for the D part of the path. Specific sites and dates/times can be selected by Ctrl-clicking on each row of the table, or all of the data can be used to create the plot by pressing **Ctrl-A**.

An example of creating a pie chart of daily reservoir storage data is shown in Figure A.1. For this figure, all sites and all dates/times were selected for plotting, after filtering on the C and E parts of the DSS path. Click the **Display** menu, and then point to **Charts**, click **Pie Charts** to open the main pie chart window. A window with an empty chart panel will appear, as shown in Figure A.2 (page A-3). Click the desired options, as described below, and then click the **Create Chart** button at the bottom of the window to create a new pie chart. If you need to change the selection of data to be plotted, or otherwise do not wish to create a pie chart, the window can be closed using either the **Cancel** button at the bottom of this window, or from the **File** menu, click **Close** from the pie chart window.



File Name: K:/

Pathnames Shown: 306 Pathnames in File: 306 File Size: 705 KB

A: B: C: STOR-RES EOP D: E: 1DAY F:

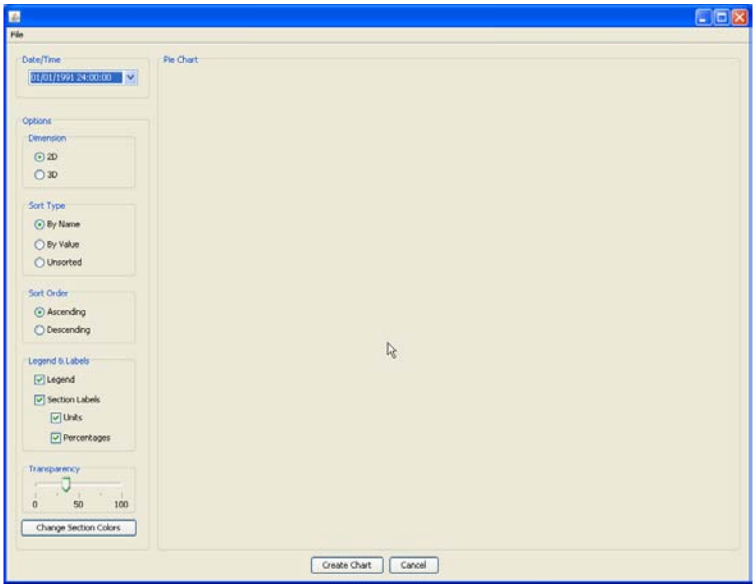
Number	Part A	Part B	Part C	Part D	Part E	Part F
1	BEAR CR	BEAR	STOR-RES EOP	01JAN1991	1DAY	
2	BEAR CR	BEAR	STOR-RES EOP	01JAN1992	1DAY	
3	BEAR CR	BEAR	STOR-RES EOP	01JAN1993	1DAY	
4	BEAR CR	BEAR	STOR-RES EOP	01JAN1994	1DAY	
5	BEAR CR	BEAR	STOR-RES EOP	01JAN1995	1DAY	
6	BEAR CR	BEAR	STOR-RES EOP	01JAN1996	1DAY	
7	BEAR CR	BEAR	STOR-RES EOP	01JAN1997	1DAY	
8	BEAR CR	BEAR	STOR-RES EOP	01JAN1998	1DAY	
9	BEAR CR	BEAR	STOR-RES EOP	01JAN1999	1DAY	
10	BEAR CR	BEAR	STOR-RES EOP	01JAN2000	1DAY	
11	BEAR CR	BEAR	STOR-RES EOP	01JAN2001	1DAY	
12	BEAR CR	BEAR	STOR-RES EOP	01JAN2002	1DAY	

No time window set.

Figure A.1 Create New Pie Chart

A.2.2 Date/Time

Select the date and time from the **Date/Time** drop-down list at the top-left of the main chart window. If only one date/time was selected from the data



Date/Time: 01/01/1991 24:00:00

Options

Dimension: ● 2D ○ 3D

Sort Type: ● By Name ○ By Value ○ Unsorted

Sort Order: ● Ascending ○ Descending

Legend & Labels: ☑ Legend ☑ Section Labels ☑ Units ☑ Percentages

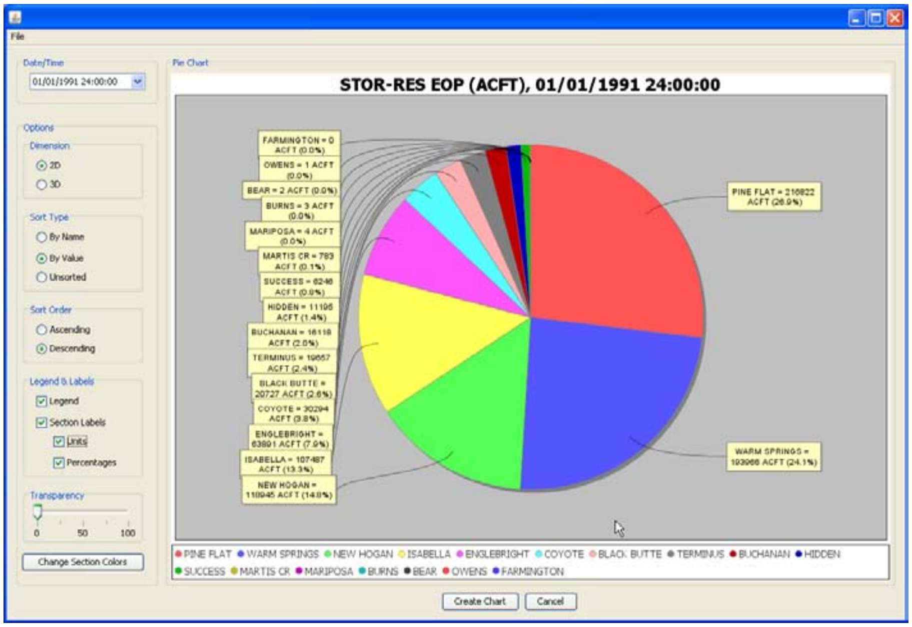
Transparency: 0 50 100

Figure A.2 Main Pie Chart Window

set before opening this window, only one date will appear in this list. Otherwise, all dates will be present, with the earliest date selected by default.

A.2.3 Chart Dimension

A two-dimensional (2D) or three-dimensional (3D) chart can be created by selecting **2D** or **3D** in the **Dimension** option box to the left of the plot panel. An example of a 2D plot is shown in Figure A.3, and a 3D plot is shown in Figure A.4 (page A-4).



File
Pie Chart
STOR-RES EOP (ACFT), 01/01/1991 24:00:00
FARMINGTON = 0 ACFT (0.0%)
OWENS = 1 ACFT (0.0%)
BEAR = 2 ACFT (0.0%)
BURNS = 3 ACFT (0.0%)
MARIPOSA = 4 ACFT (0.0%)
MARTIS CR = 783 ACFT (0.1%)
SUCCESS = 6246 ACFT (0.8%)
HIDDEN = 11195 ACFT (1.4%)
BUCHANAN = 16118 ACFT (2.0%)
TERMINUS = 19657 ACFT (2.4%)
BLACK BUTTE = 20727 ACFT (2.6%)
COYOTE = 30294 ACFT (3.8%)
ENGLEBRIGHT = 63891 ACFT (7.9%)
ISABELLA = 107487 ACFT (13.3%)
NEW HOGAN = 118945 ACFT (14.8%)
PINE FLAT = 216822 ACFT (26.9%)
WARM SPRINGS = 193966 ACFT (24.1%)
● PINE FLAT ● WARM SPRINGS ● NEW HOGAN ● ISABELLA ● ENGLEBRIGHT ● COYOTE ● BLACK BUTTE ● TERMINUS ● BUCHANAN ● HIDDEN
● SUCCESS ● MARTIS CR ● MARIPOSA ● BURNS ● BEAR ● OWENS ● FARMINGTON

Date/Time: 01/01/1991 24:00:00

Options

Dimension: ● 2D ○ 3D

Sort Type: ○ By Name ● By Value ○ Unsorted

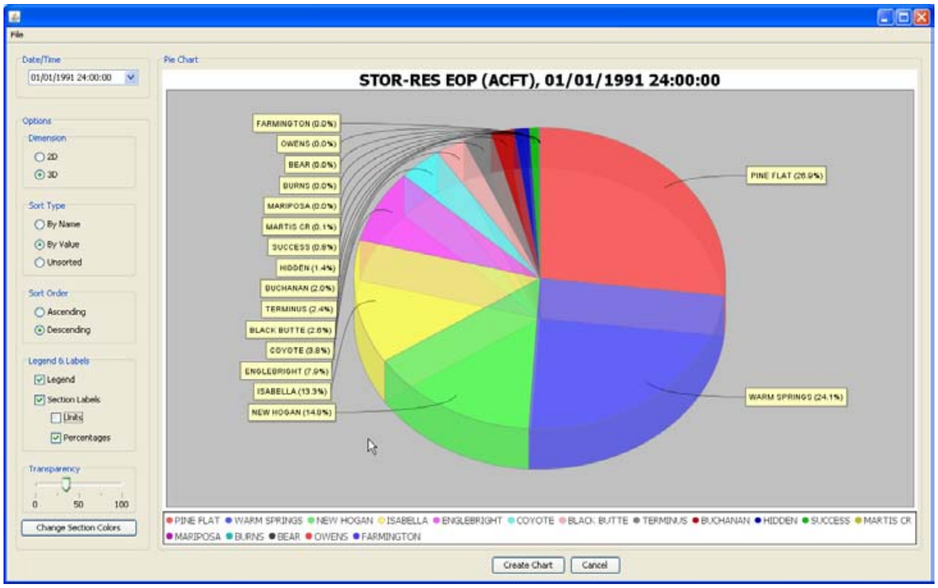
Sort Order: ○ Ascending ● Descending

Legend & Labels: ☑ Legend ☑ Section Labels ☑ Units ☑ Percentages

Transparency: 0 50 100

Change Section Colors
Create Chart
Cancel

Figure A.3 2D Pie Chart, Transparency Turned Off



File

Date/Time: 01/01/1991 24:00:00

Options

Dimension: ○ 2D ● 3D

Sort Type: ○ By Name ● By Value ○ Unsorted

Sort Order: ○ Ascending ● Descending

Legend & Labels: ☑ Legend ☑ Section Labels □ Units ☑ Percentages

Transparency: 0 50 100

Figure A.4 3D Pie Chart, Transparency Set to Thirty-Five Percent

A.2.4 Sort Type

The **Sort Type** options to the left of the plot panel allow you to specify whether the data should be sorted by site name, by data value, or by the order the data is listed in the DSS file ("unsorted"). Click **By Name** to sort the plot alphabetically, starting with the first site name at the top of the plot. This option allows quick identification of particular sites, particularly if a large number of sites were included.

Select **By Value** to sort the data by ascending or descending data values. This option creates neat and visually appealing plots that can be used to compare sites and quickly identify important differences between sites.

Select **Unsorted** to plot the data in the order in which it appears in the DSS file. This option respects the order of the DSS file but usually will not produce charts that are as clear or useful as those created using the **By Name** or **By Value** options.

A.2.5 Sort Order

The **Sort Order** options at the left of the plot window allow you to specify whether to sort the data in ascending or descending order. The **Ascending** option sorts the data around the chart clockwise, with the first value plotted at the top of the chart. **Descending** reverses the sort order by plotting the data in increasing order counter-clockwise around the plot. If **Unsorted** is selected for **Sort Type**, the **Sort Order** option will be disabled.

A.2.6 Transparency

The transparency of the entire pie chart can be changed to enhance the clarity or visual appeal of the plot. Drag the slider in the **Transparency** option box to the left to decrease the transparency and drag to the right to increase transparency. The transparency is dynamically updated as the slider is dragged. To turn transparency off and create an opaque plot, drag the slider to the far left. The 2D plot in Figure A-3 (page A-3) has the transparency set to zero, while the 3D plot in Figure A-4 (page A-4) has the transparency set to thirty-five percent.

A.2.7 Section Colors

The color of each individual slice in the pie chart can be altered by clicking the **Change Section Colors** button. A color selection window will appear (Figure A-5). Select the site name from the drop-down list to the right of the window, and then click on a color. The color of the selected slice is dynamically updated on the pie chart. Drag the color chooser window to one side to view the changes in the main chart window. Figure A-6 (page A-6) shows an example of the chart after changing the colors of the slices representing the Pine Flat and Warm Springs sites.

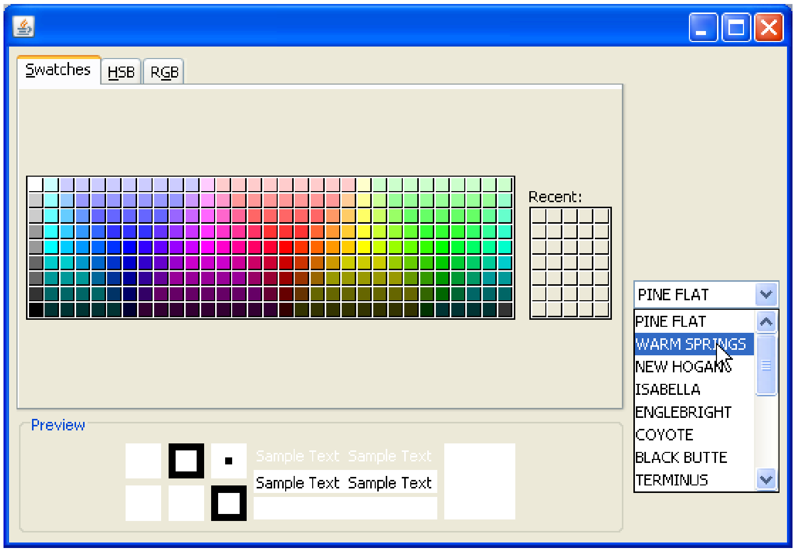
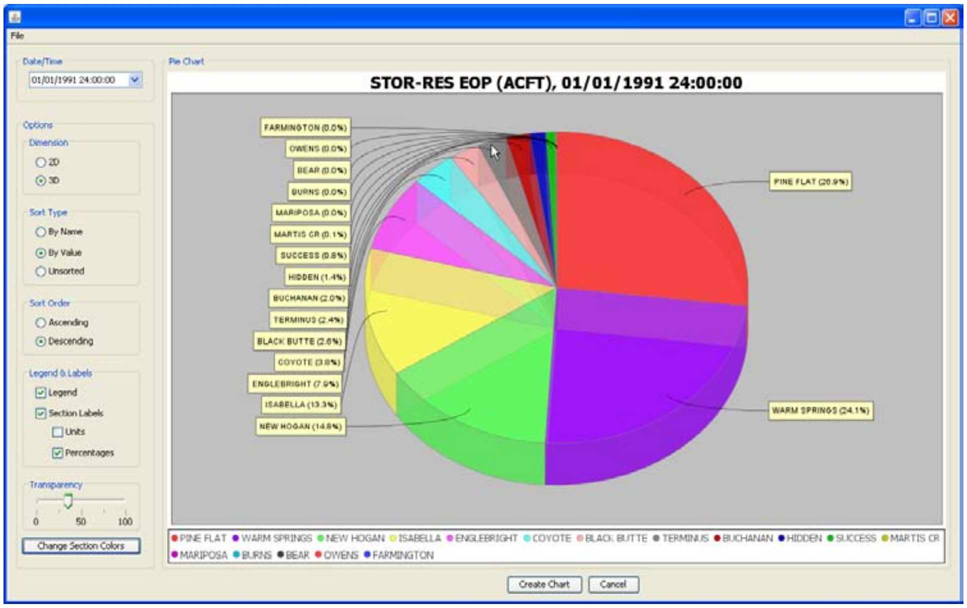


Figure A.5 Color Chooser

A.2.8 Printing and Saving

To print the pie chart, from the **File** menu, click **Print** of the pie chart window.



Date/Time: 01/01/1991 24:00:00

Options

Dimension: ○ 2D ● 3D

Sort Type: ○ By Name ● By Value ○ Unsorted

Sort Order: ○ Ascending ● Descending

Legend & Labels: ☑ Legend ☑ Section Labels □ Units ☑ Percentages

Transparency

Figure A.6 Slice Color Changed (Pine Flats and Warm Springs)

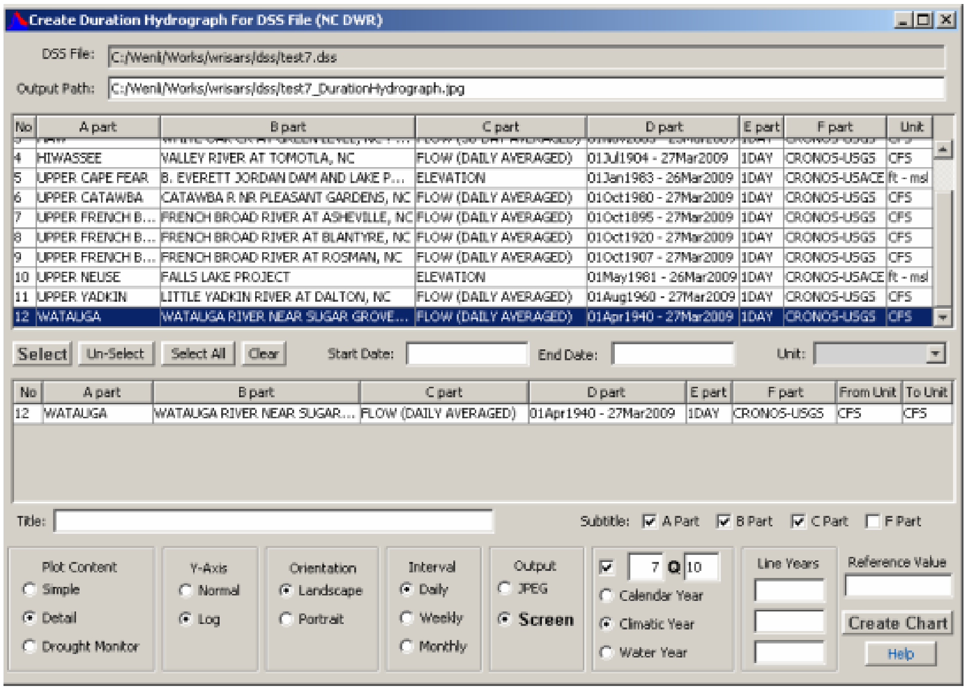
To save the pie chart to a graphics file, choose **File** → **Save As** from the pie chart menu. The chart can be saved as either a JPEG or PNG file. Select the desired extension in the **Files of type** field, and then type a name in the **File name** field, or browse for an existing file to replace with the current one. Then click **Save**. The selected extension will be appended to the file name automatically.

A.3 North Carolina DWR Duration Hydrograph

The North Carolina Department of Water Resources developed a plug-in to develop duration hydrographs for data in HEC-DSS. This is an alternative approach to the built-in capability in HEC-DSSVue in the Math functions Statistics tab. This plug-in is supported by North Carolina DWR. For questions, please contact them directly. The web site is http://www.ncwater.org/wrisars/dss.

A.3.1 Introduction

The North Carolina Department of Water Resources Duration Hydrograph Plug-in can create Daily, Weekly, or Monthly Duration Hydrograph plots with Simple, Detail, or Drought Monitor style for DSS data in HEC-DSSVue. The window that will appear is shown in Figure A.7 (page A-7. Low-Flow frequency data dQr (e.g. 7Q10, 30Q2) for water, climatic, or water year can be calculated on-the-fly and shown in the Duration Hydrograph plot (see Figure A.8, page A-7).



DSS File: C:/Wenli/Works/wrisars/dss/test7.dss

Output Path: C:/Wenli/Works/wrisars/dss/test7_DurationHydrograph.jpg

No	A part	B part	C part	D part	E part	F part	Unit
4	HIWASSEE	VALLEY RIVER AT TOMOTLA, NC	FLOW (DAILY AVERAGED)	01Jul1904 - 27Mar2009	1DAY	CRONOS-USGS	CFS
5	UPPER CAPE FEAR	B. EVERETT JORDAN DAM AND LAKE P...	ELEVATION	01Jan1983 - 26Mar2009	1DAY	CRONOS-USACE	ft - msl
6	UPPER CATAWBA	CATAWBA R NR PLEASANT GARDENS, NC	FLOW (DAILY AVERAGED)	01Oct1980 - 27Mar2009	1DAY	CRONOS-USGS	CFS
7	UPPER FRENCH B...	FRENCH BROAD RIVER AT ASHEVILLE, NC	FLOW (DAILY AVERAGED)	01Oct1895 - 27Mar2009	1DAY	CRONOS-USGS	CFS
8	UPPER FRENCH B...	FRENCH BROAD RIVER AT BLANTYRE, NC	FLOW (DAILY AVERAGED)	01Oct1920 - 27Mar2009	1DAY	CRONOS-USGS	CFS
9	UPPER FRENCH B...	FRENCH BROAD RIVER AT ROSMAN, NC	FLOW (DAILY AVERAGED)	01Oct1907 - 27Mar2009	1DAY	CRONOS-USGS	CFS
10	UPPER NEUSE	FALLS LAKE PROJECT	ELEVATION	01May1981 - 26Mar2009	1DAY	CRONOS-USACE	ft - msl
11	UPPER YADKIN	LITTLE YADKIN RIVER AT DALTON, NC	FLOW (DAILY AVERAGED)	01Aug1960 - 27Mar2009	1DAY	CRONOS-USGS	CFS
12	WATAUGA	WATAUGA RIVER NEAR SUGAR GROVE...	FLOW (DAILY AVERAGED)	01Apr1940 - 27Mar2009	1DAY	CRONOS-USGS	CFS

Start Date: End Date: Unit:

No	A part	B part	C part	D part	E part	F part	From Unit	To Unit
12	WATAUGA	WATAUGA RIVER NEAR SUGAR...	FLOW (DAILY AVERAGED)	01Apr1940 - 27Mar2009	1DAY	CRONOS-USGS	CFS	CFS

Title:

Subtitle: ☑ A Part ☑ B Part ☑ C Part ☐ F Part

Plot Content: ○ Simple ● Detail ○ Drought Monitor

Y-Axis: ○ Normal ● Log

Orientation: ● Landscape ○ Portrait

Interval: ● Daily ○ Weekly ○ Monthly

Output: ○ JPEG ● Screen

☑ 7 Q 10 ○ Calendar Year ● Climatic Year ○ Water Year

Line Years

Reference Value

Figure A.7 Create Duration Hydrograph for DSS File (NC DWR)

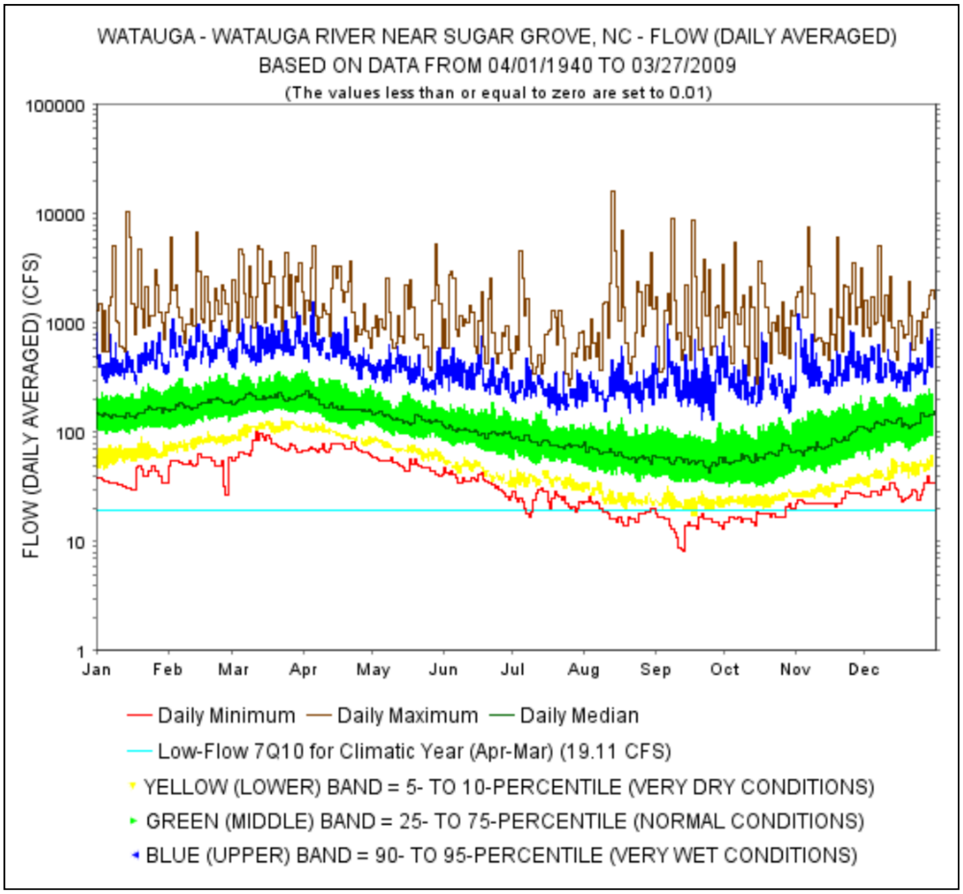
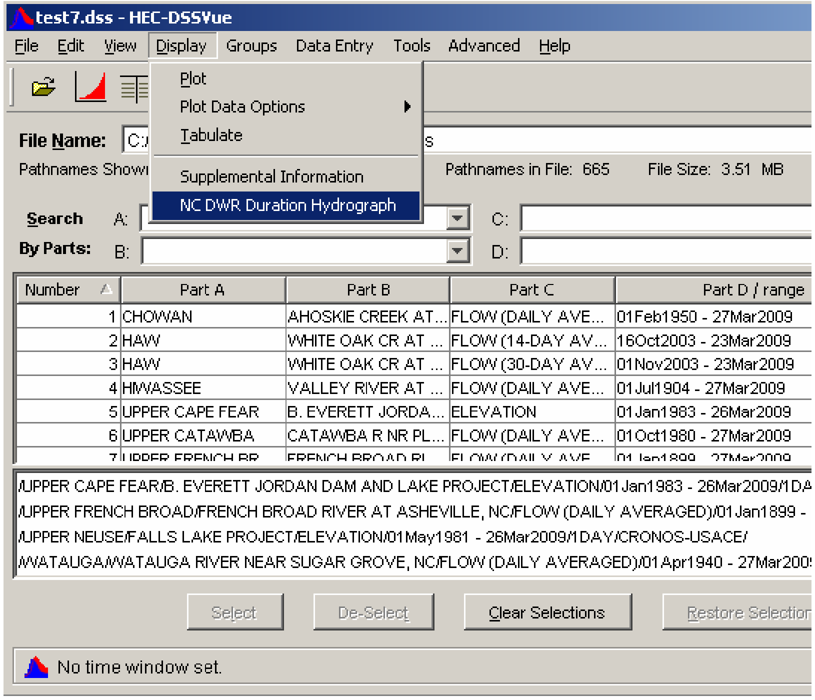


Figure A.8 Create Duration Hydrograph for DSS File (NC DWR) - Plot

A.3.2 Use

1. In HEC-DSSVue, create a new DSS file or open an existing one.
2. In the HEC-DSSVue main window, highlight or select the DSS path(s) you are interested in (if no path is selected, all paths in the DSS file will be considered as the interested paths), from the **Display** menu, click**NC DWR Duration Hydrograph** (Figure A.9).



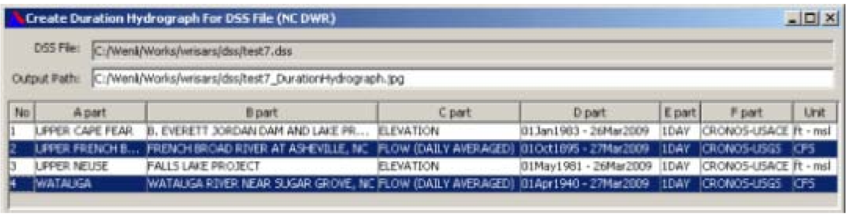
Number	Part A	Part B	Part C	Part D / range
1	CHOWAN	AHOSKIE CREEK AT...	FLOW (DAILY AVE...	01Feb1950 - 27Mar2009
2	HAW	WHITE OAK CR AT ...	FLOW (14-DAY AV...	16Oct2003 - 23Mar2009
3	HAW	WHITE OAK CR AT ...	FLOW (30-DAY AV...	01Nov2003 - 23Mar2009
4	HIWASSEE	VALLEY RIVER AT ...	FLOW (DAILY AVE...	01Jul1904 - 27Mar2009
5	UPPER CAPE FEAR	B. EVERETT JORDA...	ELEVATION	01Jan1983 - 26Mar2009
6	UPPER CATAWBA	CATAWBA R NR PL...	FLOW (DAILY AVE...	01Oct1980 - 27Mar2009
7	UPPER FRENCH BR	FRENCH BROAD RI	FLOW (DAILY AVE	01Jan1899 - 27Mar2009

/UPPER CAPE FEAR/B. EVERETT JORDAN DAM AND LAKE PROJECT/ELEVATION/01Jan1983 - 26Mar2009/1DA
/UPPER FRENCH BROAD/FRENCH BROAD RIVER AT ASHEVILLE, NC/FLOW (DAILY AVERAGED)/01Jan1899 -
/UPPER NEUSE/FALLS LAKE PROJECT/ELEVATION/01May1981 - 26Mar2009/1DAY/CRONOS-USACE/
/WATAUGA/WATAUGA RIVER NEAR SUGAR GROVE, NC/FLOW (DAILY AVERAGED)/01Apr1940 - 27Mar2009

No time window set.

Figure A.9 Open NC DWR Duration Hydrograph

3. In the newly opened **Create Duration Hydrograph For DSS File (NC DWR)** panel, click the path(s) you want to create Duration Hydrograph this time (Figure A.10).



DSS File: C:/Wenli/Works/wrisars/dss/test7.dss

Output Path: C:/Wenli/Works/wrisars/dss/test7_DurationHydrograph.jpg

No	A part	B part	C part	D part	E part	F part	Unit
1	UPPER CAPE FEAR	B. EVERETT JORDAN DAM AND LAKE PR...	ELEVATION	01Jan1983 - 26Mar2009	1DAY	CRONOS-USACE	ft - msl
2	UPPER FRENCH B...	FRENCH BROAD RIVER AT ASHEVILLE, NC	FLOW (DAILY AVERAGED)	01Oct1895 - 27Mar2009	1DAY	CRONOS-USGS	CFS
3	UPPER NEUSE	FALLS LAKE PROJECT	ELEVATION	01May1981 - 26Mar2009	1DAY	CRONOS-USACE	ft - msl
4	WATAUGA	WATAUGA RIVER NEAR SUGAR GROVE, NC	FLOW (DAILY AVERAGED)	01Apr1940 - 27Mar2009	1DAY	CRONOS-USGS	CFS

Figure A.10 Choose Pathnames to Create Duration Hydrograph

4. Select/customize the data for your Duration Hydrograph chart.
   - If you want limit the data to be used for creating the chart, enter the Start and/or End Date with yyyy-mm-dd format.
   - If you want to convert the unit for stream flow or reservoir, select one conversion method from the unit drop down list box (see Figure A.11, page A-9).

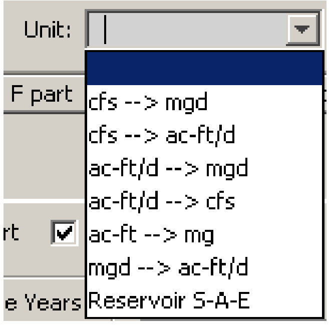
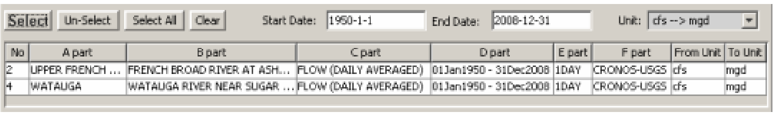


Figure A.11 Select Units for Conversion

- Click "Select" to select the path(s) you selected with the date limitations and the unit conversion.
- Clicking "Select All" will put all the path(s) in the upper table to the lower selected list table.
- You can click "Un-Select" to remove the path(s) from the selected list.
- Clicking "Clear" will clear all the selected path(s) in the lower table. The toolbar containing the buttons mentioned above is shown in Figure A.12.



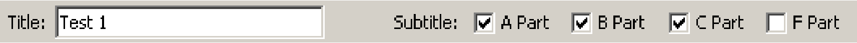
Select | Un-Select | Select All | Clear

Start Date: 1950-1-1 End Date: 2008-12-31 Unit: cfs --> mgd

No	A part	B part	C part	D part	E part	F part	From Unit	To Unit
2	UPPER FRENCH ...	FRENCH BROAD RIVER AT ASH...	FLOW (DAILY AVERAGED)	01Jan1950 - 31Dec2008	1DAY	CRONOS-USGS	cfs	mgd
4	WATAUGA	WATAUGA RIVER NEAR SUGAR ...	FLOW (DAILY AVERAGED)	01Jan1950 - 31Dec2008	1DAY	CRONOS-USGS	cfs	mgd

Figure A.12 Select, Un-Select, Select All, and Clear buttons

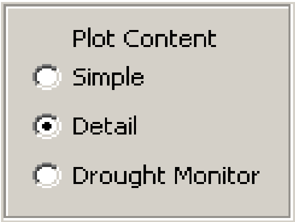
5. By default, the title for the chart will be constructed from the A, B, C, and D parts of the paths as below (Figure A.13).



Title: Test 1 Subtitle: ☑ A Part ☑ B Part ☑ C Part □ F Part

Figure A.13 Chart Title

- If you want to include the F Part, check the check box for "F Part". F Part information will be added after the C Part in the title.
- You can also populate the "Title" textbox to add your own title before the default title.



Plot Content
○ Simple
● Detail
○ Drought Monitor

Figure A.14 Plot Content

6. This plug-in can create Duration Hydrograph with three styles. You can choose one of them in the "Plot Content" group. The default is "Detail" style. (Figure A.14).
   - Simple: Plot the very dry, normal, and very wet three bands (min to percentile 25, percentile 25 to 75, and percentile 90 to max), see Figure A.15 (page A-10).
   - Detail: Plot Minimum, maximum, and median lines, in addition to the very dry, normal, and very wet three bands (percentile 5 to 10, percentile, 25 to 75, and percentile 90 to 95), see Figure A.16 (page A-10).
   - Drought Monitor: Use Drought Monitor classification and color schema to create Duration Hydrograph, see Figure A.17 (page A-10).

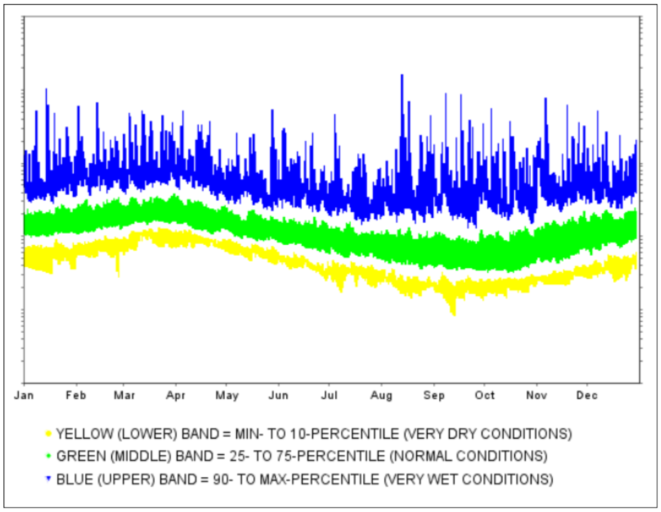


Figure A.15 Simple Plot

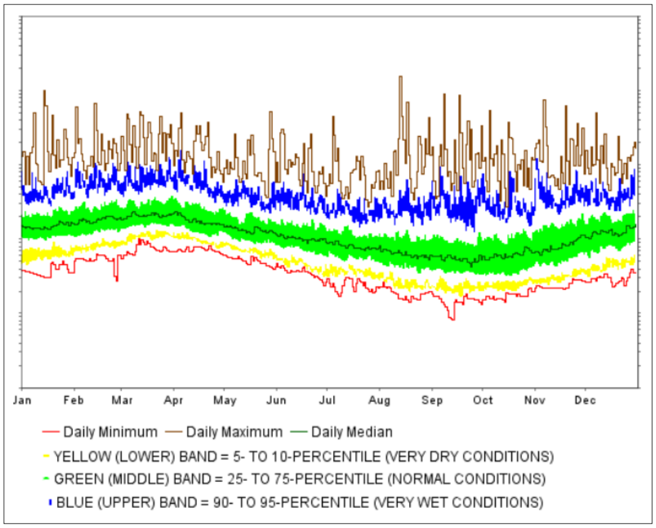


Figure A.16 Detailed Plot

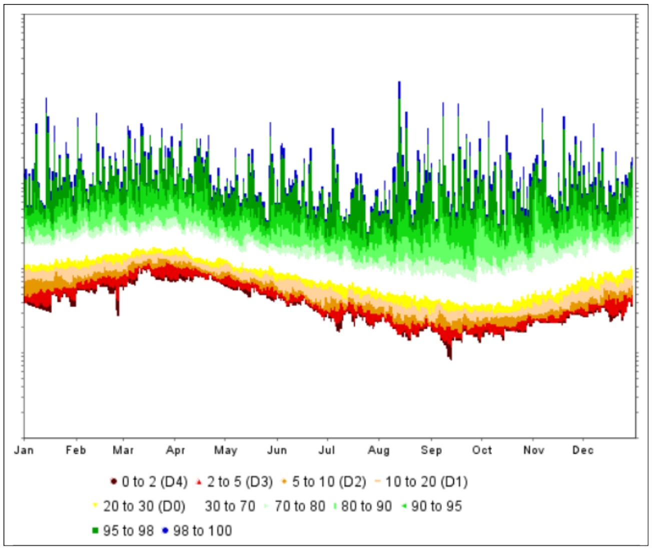
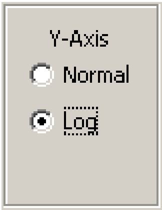


Figure A.17 Drought Monitor Plot

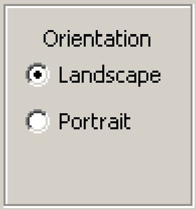
7. You have the option to create your chart with normal Y-Axis or Log Y-Axis. The default is "Normal". But generally it is better to choose "Log" for streamflows (Figure A.18).



Y-Axis
○ Normal
● Log

Figure A.18 Y-Axis

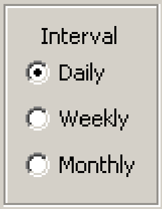
8. You can control the image orientation by choosing the Landscape or Portrait. (Figure A.19).



Orientation
● Landscape
○ Portrait

Figure A.19 Image Orientation

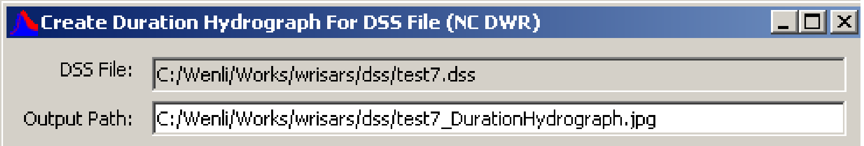
9. By default, the Duration Hydrograph will be created on the daily values. The Plug-in can also summarize the weekly or monthly averages and then create weekly or monthly chart. (Figure A.20).



Interval
● Daily
○ Weekly
○ Monthly

Figure A.20 Chart Intervals

10. By default, the Duration Hydrograph image will be created to jpg files, but not show on the screen. The file path and name is shown in the "Output Path" textbox on the top of the whole Plug-in panel, see Figure A.21. You can change the "Output Path". If multiple images will be output, sequential numbers will be added before the .jpg file extension.



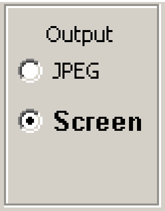
Create Duration Hydrograph For DSS File (NC DWR)

DSS File: C:/Wenli/Works/wrisars/dss/test7.dss

Output Path: C:/Wenli/Works/wrisars/dss/test7_DurationHydrograph.jpg

Figure A.21 Output Path

11. If you want to see the Duration Hydrograph image on the screen directly, choose the "Screen" output option. In this case, no jpg files will be created. (Figure A.22) You are not allowed to create more than 20 Duration Hydrograph on the screen in one run.



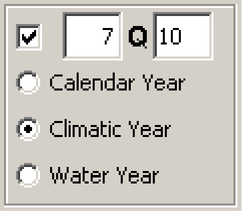
Output

○ JPEG

● Screen

Figure A.22 View Duration Hydrograph Image

12. You can choose to add Low-Flow dQr (e.g. 7Q10, or 30Q2, etc.) line to the Duration Hydrograph image, see Figure A.23.



☑ 7 **Q** 10

○ Calendar Year

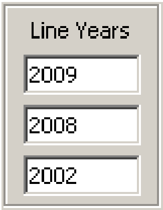
● Climatic Year

○ Water Year

Figure A.23 Add Low Flow dQr

- The textbox before the bold Q is how many consecutive days. It must be an integer between 1 and 365.
- The textbox after the bold Q is how many years of recurrence interval period. It must be an integer between 2 and 100.
- You can use three types of year for your calculation.
  Calendar Year - January 1 to December 31
  Climatic Year - April 1 to March 31. This is used by default.
  Water Year - October 1 to September 31
- dQr calculation only uses the years that have at least 365 daily data.

13. If you want to overlay some real daily data over the Duration Hydrograph, put the year numbers in the textboxes of the "Line Years" group. You can overlay one to three years of data, see Figure A.24.



Line Years

2009

2008

2002

Figure A.24 Line Years

14. Put a numeric value in the textbox under the "Reference Value", a horizontal line represent that value will be shown in the Duration Hydrograph chart, see Figure A.25.

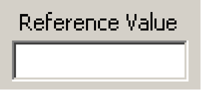


Figure A.25 Reference Value

15. Click **Create Chart** at the bottom right corner to create a Duration Hydrograph chart based on your customization.
16. If you need any help, you can click Help to open Help.

Appendix B

Writing Plug-Ins

B.1 Introduction

If you are experienced with Java, you can write a plug-in for HEC-DSSVue to provide custom capabilities for your office. Interfacing a plug-in is rather simple once you have written your own custom code. To do so, you copy existing plug-in source code and modify it so that it calls your custom code, call a registration function, compile and add the class files to a plug-in jar, and then change the manifest to show your class. To access your plug-in, simply copy the jar file to the Plugins directory and run HEC-DSSVue.

B.2 Code Overview

The "main" function of a plug-in is relatively simple; it just creates a menu or toolbar item, adds an action associated with that item, and registers it with the main class, *ListSelection*. We'll first work with the skeleton plug-in GenericDssVuePlugin.

```
public class GenericDssVuePlugin {

    public static void main(Object[] args)
    {
        final GenericDssVuePlugin plugin = new GenericDssVuePlugin();
        final ListSelection listSelection = (ListSelection) args[0];

        JMenuItem genericMenuItem = new JMenuItem("Generic Plugin");
        genericMenuItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e){
                plugin.process(listSelection);
            }
        });
       listSelection.registerPlugin(ListSelection.TOOLS, genericMenuItem);
     }
```

In process, you tell ListSelection to read the selected data:

```
protected void process(ListSelection listSelection) {
   List[] dataList = listSelection.getSelectedDataContainers();
   List tsContainers = dataList[0];
   boolean success = processData(listSelection, tsContainers);
}
```

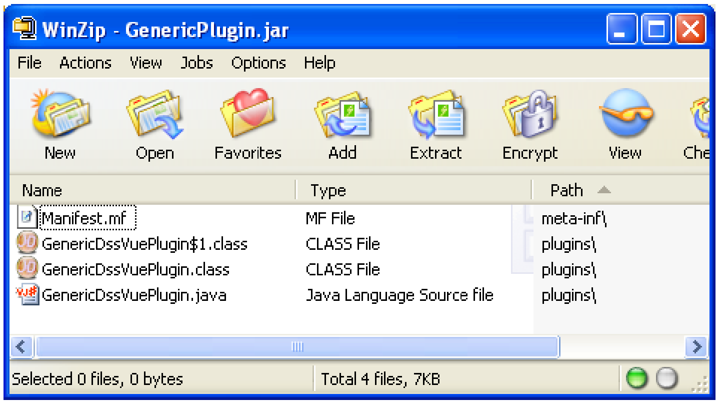
B.3 Procedure

To write a plug-in you need Java and a sample plug-in to start with. The GenericDssVuePlugin is included with the program in the Samples directory. The source code is located in that jar, along with the class files and manifest.

You will need to refer to the functions listed in the Scripting Chapter. These include documentation of what is in a TimeSeriesContainer class, how to use HecTime, and various ListSelection calls. All of the scriptable calls are available for use in plug-ins.

1. You need a Java IDE (Integrated Development Environment), but the examples shown will just be using the J2SE (Java 2 Standard Edition) SDK (Software Development Kit). You can download the SDK from sun at: http://java.sun.com/j2se/. You can get an excellent Java IDE for free named Eclipse from: http://www.eclipse.org/.
2. From the GenericDssVuePlugin.jar file, extract "GenericDssVuePlugin.java" to a java file named according to your task. You can put it in any directory you desire if you are not going to use a package name. If you use a package name, then you have to put it into a directory that is the same as the package name.
3. Edit the .java file you copied:
   a). Change the class name and the name in the main to the task name you gave the .java file. The examples do not specify a package or directory, but you may want to.
   b). Change the JMenuItem text name to reflect your task.
   c). Register the plug-in with the appropriate HEC-DSSVue menu, or you can make a toolbar button and add it to the main toolbar.
   d). Modify the process function(s) to do what you want with the data.
   e). Comment or uncomment code to save changed data with the same pathname or with a different pathname or allow the user to change the pathname, as desired.
   f). File you changes.
4. Compile your code. Add hec.jar, heclib.jar and rma.jar in your class path:
```
javac -classpath "C:\Program Files\Hec\HEC-DSSVue\
jar\hec.jar;C:\Program Files\Hec\HEC-DSSVue\jar
\heclib.jar;C:\Program Files\Hec\HEC-DSSVue\jar\
rma.jar" MyPlugin.java
```
5. Copy the "GenericDssVuePlugin.jar" to a jar file named according to your task in the HEC-DSSVue\Plugins directory.
6. Use pkzip or other zip utility to open your .jar file. Add in the class files that you generated in the compile (maybe more than one). Don’t include the directory names, unless you are using

packages, where you will have to use a directory name that matches the package. (Figure B.1).

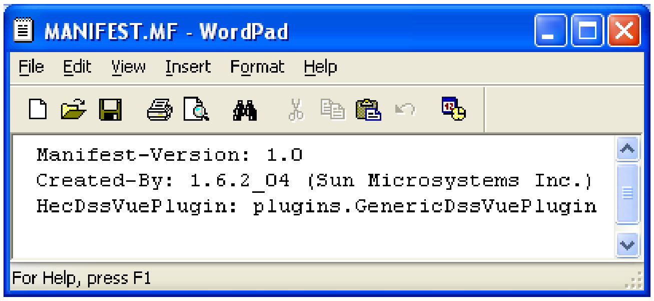

WinZip - GenericPlugin.jar
File Actions View Jobs Options Help
New
Open
Favorites
Add
Extract
Encrypt
View
Che

Name	Type	Path
Manifest.mf	MF File	meta-inf\
GenericDssVuePlugin$1.class	CLASS File	plugins\
GenericDssVuePlugin.class	CLASS File	plugins\
GenericDssVuePlugin.java	Java Language Source file	plugins\

Selected 0 files, 0 bytes Total 4 files, 7KB

Figure B.1 GenericPlugin.jar Directory

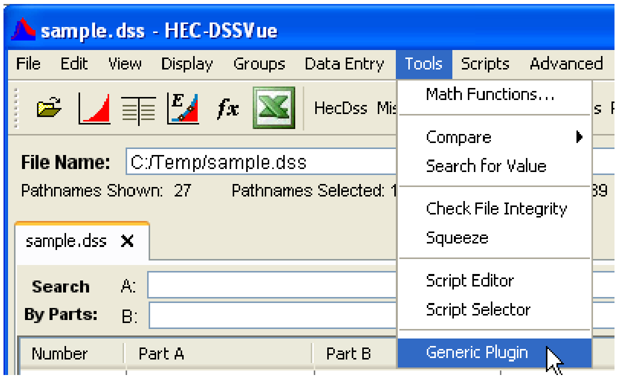
7. Extract the Manifest.mf file with the path. Edit this file with Wordpad and change the GenericDssVuePlugin name to the name of your class. Add the modified file back to the .jar file and be sure that you keep the path specified (meta-inf\). (Figure B.2).



```
Manifest-Version: 1.0
Created-By: 1.6.2_04 (Sun Microsystems Inc.)
HecDssVuePlugin: plugins.GenericDssVuePlugin
```

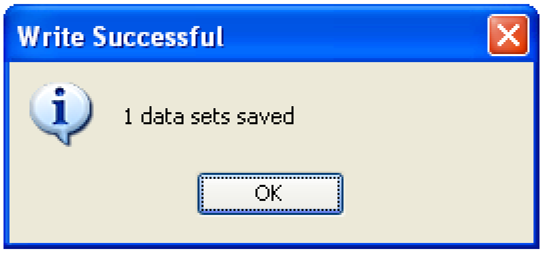
Figure B.2 Manifest *.mf* File

8. You should be good to go at this point. Run HEC-DSSVue and test your plug-in. (Figure B.3). If you have problems, compare what you have to one of the simple example plug-ins. If the run is successful, you will receive a confirmation as is shown in Figure B.4 (page B-4).



File Name: C:/Temp/sample.dss
Pathnames Shown: 27 Pathnames Selected: 1

Figure B.3 Run Generic Plugin



1 data sets saved

Figure B.4 Successful Run Confirmation

B.4 Generic DssVuePlugin Source

```
package plugins;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JOptionPane;
import java.util.*;
import hec.dssgui.ListSelection;
import hec.io.TimeSeriesContainer;
import hec.heclib.dss.DSSPathname;
import javax.swing.JMenuItem;

public class GenericDssVuePlugin
{
   public static void main(Object[] args)
   {
      final GenericDssVuePlugin plugin = new GenericDssVuePlugin();
      final ListSelection listSelection = (ListSelection) args[0];

      JMenuItem genericMenuItem = new JMenuItem("Generic Plugin");
      genericMenuItem.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e){
            plugin.process(listSelection);
         }
      });
      listSelection.registerPlugin(ListSelection.TOOLS, genericMenuItem);
   }

   protected void process(ListSelection listSelection)
   {
      //  Reads data from DSS and returns it in data containers in a list
of lists
      List[] dataList = listSelection.getSelectedDataContainers();
      if (dataList == null) {
         JOptionPane.showMessageDialog(listSelection, "No records
selected",
                                "Cannot read data",
JOptionPane.WARNING_MESSAGE);
         return;
      }
      // The first list contains a list of TimeSeriesContainers
      // The second is PairedDataContainers, third is text, fourth is
gridded
      List tsContainers = dataList[0];
      if ( (tsContainers != null) && (tsContainers.size() > 0)) {
         boolean success = processData(listSelection, tsContainers);
         if (!success) {
            JOptionPane.showMessageDialog(listSelection, "Error in
processing data",
                                "Error", JOptionPane.WARNING_MESSAGE);
         }
      }
      else {
         JOptionPane.showMessageDialog(listSelection, "No time series data
read",
                                "No data found",
JOptionPane.WARNING_MESSAGE);
      }
   }

   protected boolean processData(ListSelection listSelection, List
tsContainers)
   {
      for (int i = 0; i < tsContainers.size(); i++) {
         TimeSeriesContainer tsc = (TimeSeriesContainer)
tsContainers.get(i);
         //  Do something with the data here
         //  For this example, we will add "10" to each value
         for (int j=0; j<tsc.values.length; j++) {
            tsc.values[j] += 10.0;
         }
         //  Save the data, if needed
         DSSPathname path = new DSSPathname(tsc.fullName);
         path.setFPart(path.fPart() + " + MOD");
         tsc.fullName = path.pathname();
         //  You can save or saveAs individual containers as you modify
them
         //listSelection.save(tsc);
         //listSelection.saveAs(tsc);
      }
      //  OR you can save the list of containers.
      listSelection.save((Vector)tsContainers);
      //  Update the catalog if you added records
      listSelection.refreshCatalog();
      //  Display a confirmation message
      JOptionPane.showMessageDialog(listSelection, tsContainers.size() + "
data sets saved",
                             "Write Successful",
JOptionPane.INFORMATION_MESSAGE);
      return true;
   }
}
```

B.5 Primary Functions

B.5.1 ListSelection Functions

To register your plug-in with HEC-DSSVue, the following functions are available:

```
public static void main(Object[] args) {
    final ListSelection listSelection = (ListSelection) args[0];
```

1. Register your plug-in with:
   a. `public void registerPlugin(int menuNumber, JMenuItem menuItem)`

   where *menuNumber* is:

```
FILE = 0;
EDIT = 1;
VIEW = 2;
DISPLAY = 3;
GROUPS = 4;
DATA_ENTRY = 5;
TOOLS = 6;
CUSTOM = 7;
```

```
        SCRIPTS = 8;
        ADVANCED = 9;
        HELP = 10;
```

b. `public void registerImportPlugin(ImportPlugin importPlugin, JMenuItem menuItem, String dropAndDragExtension)`

c. `public void registerExportPlugin(JMenuItem menuItem)`

d. `public void addToolBarButton(JButton button)`

2. You can put your plug-ins in a custom menu that you can name (e.g., listSelection.setCustomMenu(Our Math Functions): `public void setCustomMenu(String menuText)`

3. Read data for selected pathnames with getSelectedData Containers():

```
List[0] will be a list of TimeSeriesContainer, or null if
none read.
List[1] will be a list of PairedDataContainer or null
List[2] will be a list of TextContainer
```

4. Write data to HEC-DSS:

```
boolean save(DataContainer dataContainer);
boolean saveAs(DataContainer dataContainer);
int save(Vector DataContainer);
```

5. Others:

```
DataReferenceSet getSelectedPathnames()
void updateCatalog();
boolean setDSSFilename(String DSSFileName);
String getDSSFilename();
void addToSelection(String pathname)
public void setTimeWindow(String timeWindow)
void setTimeWindow(String start, String end, boolean applyAll)
void setTimeWindow(HecTime start, HecTime end, boolean applyAll)
void tabulate(List selectedDataContainers);
void plot(List selectedDataContainers);
```

B.5.2 DSSPathname

hec.heclib.dss.DSSPathname

Table B.1 DSSPathname

Field	Type	Description
getPathname ();	String	Returns the pathname
getAPart();	String	Gets the A part
getBPart();	String	Gets the B part
getCPart();	String	Gets the C part
getDPart();	String	Gets the D part
getEPart();	String	Gets the E part
getFPart();	String	Gets the F part
setAPart (String aPart);	void	Overrides the default A part
setBPart (String bPart);	void	Overrides the default B part
setCPart (String cPart);	void	Overrides the default C part

Field	Type	Description
setDPart (String dPart);	void	Overrides the default D part
setEPart (String ePart);	void	Overrides the default E part
setFPart (String fPart);	void	Overrides the default F part
setPathname (String pathname);	int	sets the pathname and returns 0 if successful
static getDefaultAPart();	String	Gets the default A part
static getDefaultBPart();	String	Gets the default B part
static getDefaultCPart();	String	Gets the default C part
static getDefaultDPart();	String	Gets the default D part
static getDefaultEPart();	String	Gets the default E part
static getDefaultFPart();	String	Gets the default F part
static setDefaultAPart (String aPart);	void	Sets the default A part
static setDefaultBPart (String bPart);	void	Sets the default B part
static setDefaultCPart (String cPart);	void	Sets the default C part
static setDefaultDPart (String dPart);	void	Sets the default D part
static setDefaultEPart (String ePart);	void	Sets the default E part
static setDefaultFPart (String fPart);	void	Sets the default F part

B.5.3 TimeSeriesContainer

.io.TimeSeriesContainer - contains the data read or to be written:

Table B.2 TimeSeriesContainter

Field	Type	Description
endTime;	int	The end time of the time window. If the data were retrieved, the end time may be later than the last time in the times list.
fullName;	String	Pathname
fileName;	String	Dss filename
interval;	int	The interval, in minutes, between each set of consecutive times in the times list. For irregular-interval times, the interval field is set to -1.
location	String	The location associated with the data (DSS pathname B-part if the DataContainer is associated with the DSS file).
numberValues;	int	The length of values and times lists.
parameter	String	The parameter associated with the data
precision = -1;	int	Digits to the right of the decimal
quality	int[ ]	The optional list of quality flags. If this list is present, there must be a quality for each value
startTime;	int	The start time of the time window. If the data were retrieved, the start time may be earlier than the first time in the times list.
subLocation	String	The sub-location associated with the data.

Field	Type	Description
subParameter	String	The sub-parameter associated with the data.
subVersion	String	The sub-version associated with the data.
times;	int[]	The array of times. There is a one-to-one correspondence between the times array and the values array .
timeZoneID	String	The time-zone for this object. Set to “” if none.
timeZoneRawOffset	int	The offset, in milliseconds, from UTC to the time zone.
type	String	Field containing the data type (PER-AVER, etc)
units	String	The units of the data (CFS, etc)
values	double[]	The data values, each of which has a corresponding time in the times array and optionally a corresponding quality in the quality array. All arrays must have the same length.
version	String	The version associated with the data (DSS pathname F-part if the DataContainer is associated with a DSS file).
watershed	String	The watershed associated with the data (DSS pathname A-part if the DataContainer is associated with a DSS file).

B.5.4 PairedDataContainer

Table B.3 PairedDataContainter

Field	Type	Description
endTime;	int	The end time of the time window. If the data were retrieved, the end time may be later than the last time in the times list.
fullName;	String	Pathname
fileName;	String	Dss filename
interval;	int	The interval, in minutes, between each set of consecutive times in the times list. For irregular-interval times, the interval field is set to -1.
location	String	The location associated with the data (DSS pathname B-part if the DataContainer is associated with the DSS file).
numberValues;	int	The length of values and times lists.
parameter	String	The parameter associated with the data
precision = -1;	int	Digits to the right of the decimal
quality	int[ ]	The optional list of quality flags. If this list is present, there must be a quality for each value
startTime;	int	The start time of the time window. If the data were retrieved, the start time may be earlier than the first time in the times list.
subLocation	String	The sub-location associated with the data.
subParameter	String	The sub-parameter associated with the data.

Field	Type	Description
subVersion	String	The sub-version associated with the data.
times;	int[ ]	The array of times. There is a one-to-one correspondence between the times array and the values array .
timeZoneID	String	The time-zone for this object. Set to “” if none.
timeZoneRawOffset	int	The offset, in milliseconds, from UTC to the time zone.
type	String	Field containing the data type (PER-AVER, etc)
units	String	The units of the data (CFS, etc)
values	double[ ]	The data values, each of which has a corresponding time in the times array and optionally a corresponding quality in the quality array. All arrays must have the same length.
version	String	The version associated with the data (DSS pathname F-part if the DataContainer is associated with a DSS file).
watershed	String	The watershed associated with the data (DSS pathname A-part if the DataContainer is associated with a DSS file).

B.5.5 HecTime

Table B.4 HecTime

Field	Type	Description
endTime;	int	The end time of the time window. If the data were retrieved, the end time may be later than the last time in the times list.
fullName;	String	Pathname
fileName;	String	Dss filename
interval;	int	The interval, in minutes, between each set of consecutive times in the times list. For irregular-interval times, the interval field is set to -1.
location	String	The location associated with the data (DSS pathname B-part if the DataContainer is associated with the DSS file).
numberValues;	int	The length of values and times lists.
parameter	String	The parameter associated with the data
precision = -1;	int	Digits to the right of the decimal
quality	int[ ]	The optional list of quality flags. If this list is present, there must be a quality for each value
startTime;	int	The start time of the time window. If the data were retrieved, the start time may be earlier than the first time in the times list.
subLocation	String	The sub-location associated with the data.
subParameter	String	The sub-parameter associated with the data.
subVersion	String	The sub-version associated with the data.

Field	Type	Description
times;	int[ ]	The array of times. There is a one-to-one correspondence between the times array and the values array.
timeZoneID	String	The time-zone for this object. Set to “” if none.
timeZoneRawOffset	int	The offset, in milliseconds, from UTC to the time zone.
type	String	Field containing the data type (PER-AVER, etc)
units	String	The units of the data (CFS, etc)
values	double[ ]	The data values, each of which has a corresponding time in the times array and optionally a corresponding quality in the quality array. All arrays must have the same length.
version	String	The version associated with the data (DSS pathname F-part if the DataContainer is associated with a DSS file).
watershed	String	The watershed associated with the data (DSS pathname A-part if the DataContainer is associated with a DSS file).

B.5.6 HecDouble

Table B.5 HecDouble

Field	Type	Description
isDefined();	boolean	false if missing or not set
set(double value);	void	Sets the value
set(double value, int precision);	void	Sets the value and precision (the number of values past the decimal place to display)
set(String value);		Sets the value and precision
setPrecision (int precision);	int	Sets the precision – the number of values past the decimal place to display. (does not change what is stored)
string();	String	Returns a String of the value
string(boolean printCommas);	String	Returns a String of the value. If printCommas is true, commas are inserted in the appropriate locations in the string.
string(boolean printCommas, DecimalFormatSymbols symbols);	String	Returns a String of the value, formatted according to the format given.
value();	double	returns the actual number

B.5.7 HecDoubleArray

hec.heclib.util.HecDoubleArray - an array of HecDouble

Table B.6 HecDoubleArray

Method	Returns	Description
element(int elementNumber);	HecDouble	Returns the HecDouble object at that location.
set (double values[]);	void	Sets the values.
setPrecision(int precision);	int	Sets the number of values to display past the decimal point for all values in the array.
string(int elementNumber);	String	Returns the value at the element given as a String
string(int elementNumber, boolean printCommas);	String	Returns the value at the element given as a String. If printCommas is true, commas are inserted at the appropriate locations.

B.6 ToTextPlugin

The ToTextPlugin takes selected time series data sets and writes the data out to a text file. You can use this to create an input file in a certain format for other programs.

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JOptionPane;
import java.util.*;
import java.io.*;

import hec.dssgui.ListSelection;
import hec.io.TimeSeriesContainer;
import hec.heclib.util.HecTime;
import rma.awt.FlatPanelButton;
import javax.swing.JFileChooser;
import hec.heclib.util.HecDoubleArray;
import hec.dataTable.*;

public class ToTextPlugin
{
   public static void main(Object[] args)
   {
      final ToTextPlugin plugin = new ToTextPlugin();
      final ListSelection listSelection = (ListSelection) args[0];

      FlatPanelButton button = new FlatPanelButton("To Text");
      button.addActionListener(new ActionListener()
      {
         public void actionPerformed(ActionEvent e)
         {
               plugin.process(listSelection);
         }
      });
      listSelection.getToolBar().add(button);
   }

    protected void process(ListSelection listSelection)
    {
        //  Reads data from DSS and returns it in data containers in a
list of lists
        List[] dataList = listSelection.getSelectedDataContainers();
        if (dataList == null) {
            JOptionPane.showMessageDialog(listSelection, "No records
selected",
                                          "Cannot read data",
JOptionPane.WARNING_MESSAGE);
            return;
        }
        // The first list contains a list of TimeSeriesContainers
        // The second is PairedDataContainers, third is text, fourth is
gridded
        List tsContainers = dataList[0];
        if ( (tsContainers != null) && (tsContainers.size()
0)) {
            processData(listSelection, tsContainers);
        }
        else {
            JOptionPane.showMessageDialog(listSelection, "No time series
data read",
                                          "No data found",
JOptionPane.WARNING_MESSAGE);
        }
    }


    protected boolean processData(ListSelection listSelection,
                                  List tsContainers) {
        try {
            //  Get the name of the text file to write to.
            JFileChooser chooser = new JFileChooser();
            chooser.setAcceptAllFileFilterUsed(false);
            chooser.setDialogTitle("Enter file to save data to");
            chooser.setFileFilter(new rma.util.RMAFilenameFilter("txt",
"*.txt"));
            chooser.showOpenDialog(listSelection);
            java.io.File file = chooser.getSelectedFile();
            if (file == null)
                return false;
            String fileName = file.getAbsolutePath();
            if (fileName.endsWith(".txt") == false)
                fileName = fileName + ".txt";
            PrintWriter textOut = new PrintWriter(new
FileWriter(fileName));
            //  Write either as either data in a single column or data
sets
            //  in multi-column.  The table software organizes data sets
with times.
            boolean tableStyle = true;
            if (tableStyle) {
                writeTableFormat(textOut, tsContainers);
            }
            else {
                writeSingleSets(textOut, tsContainers);
            }
            textOut.close();
            JOptionPane.showMessageDialog(listSelection,
                                          tsContainers.size() + " data
sets written to " +
                                             fileName,
                                          "Write Successful",
ptionPane.INFORMATION_MESSAGE);
            return true;
        }
        catch (Exception e) {
            JOptionPane.showMessageDialog(listSelection, e.toString(),
                                                  "Cannot write to file",

               JOptionPane.WARNING_MESSAGE);
            return false;
        }
    }

    protected void writeSingleSets(PrintWriter textOut, List tsContainers)
{
        HecTime time = new HecTime();
        for (int i = 0; i < tsContainers.size(); i++) {
            TimeSeriesContainer tsc = (TimeSeriesContainer)
```

```
tsContainers.get(i);
            textOut.println(tsc.fullName);
            //  HecDouble will take care of missing data and the precision
            HecDoubleArray values = new HecDoubleArray();
            values.set(tsc.values);
            values.setPrecision(tsc.precision);
            for (int j = 0; j < tsc.numberValues; j++) {
                time.set(tsc.times[j]);
                textOut.println(time.toString(4) + ";  " +
values.element(j));
            }
        }
    }

    protected void writeTableFormat(PrintWriter textOut, List
tsContainers) {
        HecDataTable dataTable = new HecDataTable(null);
        dataTable.setData(tsContainers, false, 0);
        // UndefinedStyle: 0 = ""; 1 = -901.0; 2 = "M"; 3 = "-M-"
        dataTable.setUndefinedStyle(2);
        //  See HecTime (or heclib juldat) for date styles
        dataTable.setDateStyle(4);
        int numberColumns = dataTable.getColumnCount();
        int numberRows = dataTable.getRowCount();
        int startingRow = dataTable.getNumberHeaderRows();
        for (int i=startingRow; i<numberRows; i++) {
            StringBuffer sb = new StringBuffer();
            for (int j=0; j<numberColumns; j++) {
                sb.append((dataTable.getValueAt(i, j)).toString());
                sb.append("   ");
            }
            textOut.println(sb.toString());
        }
    }
}
```

Appendix C

Example Scripts

C.1 Introduction

This appendix provides several example scripts to illustrate the use of scripting with HEC-DSSVue. Documentation for the calls in these scripts is provided in Chapter 8 of the HEC-DSSSVue User Manual. The intent of this appendix is to show variations in scripts and how to implement them.

Usually it is easiest to write a script by taking an example and modifying it to meet your needs. The scripts in this appendix are provided with the program in the directory "Samples". The data for the scripts is provided in the *sample.dss* file. Most scripts access this file in the "*C:\temp*" directory. To execute the scripts, you will need to copy sample.dss to the C:\temp directory, and the sample scripts to "*HecDssVue\Scripts*" directory, or you may import them with the Script Editor.

This section covers math scripts, export/import scripts, simple graphics, "headless" operation (graphics in a background mode), complex graphics and calling scripts from scripts.

C.2 Sample Math Scripts

Math scripts use the HecMath package, so no direct access to HEC-DSSVue or ListSelection is needed. You can also get a DataContainer and access the data directly. The two approaches are shown in the following two scripts, which both add the number "10" to each value.

AddTenTsc.py

```
#  Add 10 to each value using TimeSeriesContainer
#
from hec.script import *
from hec.heclib.dss import *
from hec.dataTable import *
from java import *

#  Open the file and get the data
try:
  dssFile = HecDss.open("C:/temp/sample.dss", "10MAR2006 2400,
09APR2006 2400")
  outflow = dssFile.get("/AMERICAN/FOLSOM/FLOW-RES
OUT/01JAN2006/1DAY/OBS/")
  i = 0
  for value in outflow.values :
    outflow.values[i] += 10.
    i += 1

  path = DSSPathname(outflow.fullName)
  fPart = path.fPart() + " + 10"
  path.setFPart(fPart)
  outflow.fullName = path.getPathname()

  dssFile.put(outflow)

except java.lang.Exception, e :
  #  Take care of any missing data or errors
MessageBox.showError(e.getMessage(), "Error reading data")
```

AddTenHecMath.py

```
#  Add 10 to each value using HecMath
#
from hec.script import *
from hec.hecmath import *
from java import *

#  Open the file and get the data
try:
  dssFile = DSS.open("C:/temp/sample.dss", "10MAR2006 2400,
09APR2006 2400")
  outflow = dssFile.read("/AMERICAN/FOLSOM/FLOW-RES
OUT/01JAN2006/1DAY/OBS/")
  newOutflow = outflow.add(10.)

  path = DSSPathname(newOutflow.getPath())
  fPart = path.fPart() + " + 10 HecMath"
  path.setFPart(fPart)
  newOutflow.setPathname(path.getPathname())

  dssFile.write(newOutflow)

except java.lang.Exception, e :
  #  Take care of any missing data or errors
   MessageBox.showError(e.getMessage(), "Error reading data")
```

Unless the function you are using is not something not supported by HecMath, the HecMath functions are easier to use than manipulating the data directly in a DataContainer.

A routine functions suitable for scripting is estimating missing values. This is done with the HecMath function *estimateForMissingValues()*. An example is show for **EstimateMissing.py**.

```
#  Fill in missing values using HecMath
#
from hec.script import *
from hec.hecmath import *
from java import *
```

```
#  Open the file and get the data
try:
  dssFile = DSS.open("C:/temp/sample.dss", "10MAR2006 2400,
09APR2006 2400")
  flow = dssFile.read("/AMERICAN/FOLSOM/FLOW-RES
IN/01JAN2006/1DAY/OBS/")
  newflow = flow.estimateForMissingValues(10)
  path = DSSPathname(newflow.getPath())
  fPart = path.fPart() + " + FILLED"
  path.setFPart(fPart)
  newflow.setPathname(path.getPathname())
  dssFile.write( newflow)
except java.lang.Exception, e :
  #  Take care of any missing data or errors
   MessageBox.showError(e.getMessage(), "Error reading data")
```

Another useful function is smoothing, typically used for reservoir data. Of course this could have been just an additional line added to the EstimateMissing.py script.

Smoothing.py

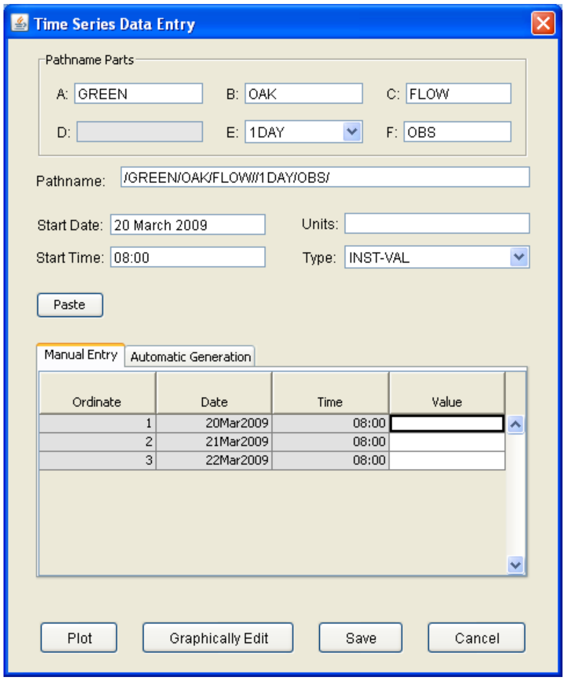
```
#  Smooth values using HecMath
from hec.script import *
from hec.hecmath import *
from hec.heclib.dss import *
from java import *

#  Open the file and get the data
try:
  dssFile = DSS.open("C:/temp/sample.dss", "10MAR2006 2400,
09APR2006 2400")
  flow = dssFile.read("/AMERICAN/FOLSOM/FLOW-RES
IN/01JAN2006/1DAY/OBS/")
  newflow = flow.centeredMovingAverage(7, 1, 0)
  path = DSSPathname(newflow.getPath())
  fPart = path.fPart() + " + SMOOTH"
  path.setFPart(fPart)
  newflow.setPathname(path.getPathname())
  dssFile.write( newflow)
except java.lang.Exception, e :
  #  Take care of any missing data or errors
MessageBox.showError(e.getMessage(), "Error reading data")
```

C.3 Sample Import and Export Scripts

You can write scripts to import or export data with various formats. You can also write scripts to display a data entry table that can be used to enter data manually.

C.3.1 Manually Entry Scripts

Both the manual time series data entry screen and paired data entry screen can be brought up from a script. They both can be passed a pathname to preset the dialog, and the time series dialog can be given a start time. For example, to display an initialized time series data entry screen (Figure C.1), the following **DataEntry.py** script can be used:



A: GREEN B: OAK C: FLOW
D: E: 1DAY F: OBS

Pathname: /GREEN/OAK/FLOW//1DAY/OBS/

Start Date: 20 March 2009 Units:
Start Time: 08:00 Type: INST-VAL

Ordinate	Date	Time	Value
1	20Mar2009	08:00	
2	21Mar2009	08:00	
3	22Mar2009	08:00	

Figure C.1 Time Series Data Entry Screen

```
from hec.script import *
from hec.dssgui import *
from hec.heclib.util import *
import java

ls = ListSelection.getMainWindow()
ls.setIsInteractive(1,0)  # Turn off popups
ls.open("C:/temp/mydb.dss")

time = HecTime()
time.setCurrent()
time.setTime("0800")
time.addDays(-5)

ls.timeSeriesDataEntry("/GREEN/OAK/FLOW//1DAY/OBS",
time.dateAndTime(4))
# ls.finish()  # Batch mode only
```

C.3.2 SHEF Import/Export Scripts

If the SHEF parameter files are setup correctly, a simple script can be written to import or export SHEF data. Essentially, the same steps are followed in the script as would be done interactively through the ListSelection class. For example, **FolsomToShef.py**:

```
from hec.script import *
from hec.heclib.dss import *
from hec.dssgui import *
import java
```

```
#  Open the file and get the data
try:
  dssFile = HecDss.open("C:/temp/sample.dss", "10MAR2006 2400, 09APR2006
2400")
  precip = dssFile.get("/AMERICAN/FOLSOM/PRECIP-
BASIN/01JAN2006/1DAY/OBS/")
  stor = dssFile.get("/AMERICAN/FOLSOM/ STOR-RES EOP/01JAN2006/1DAY/OBS/")
  topcon = dssFile.get("/AMERICAN/FOLSOM/TOP CON
STOR/01JAN2006/1DAY/OBS/")
  sagca = dssFile.get("/AMERICAN/FOLSOM-SAGCA/TOP CON
STOR/01JAN2006/1DAY/OBS/")
  inflow = dssFile.get("/AMERICAN/FOLSOM/FLOW-RES IN/01JAN2006/1DAY/OBS/")
  outflow = dssFile.get("/AMERICAN/FOLSOM/FLOW-RES
OUT/01JAN2006/1DAY/OBS/")
except java.lang.Exception, e :
  #  Take care of any missing data or errors
   MessageBox.showError(e.getMessage(), "Error reading data")

#  Add Data
datasets = java.util.Vector()
datasets.add(precip)
datasets.add(stor)
datasets.add(topcon)
datasets.add(sagca)
datasets.add(inflow)
datasets.add(outflow)

ls = ListSelection.getMainWindow()
ls.setIsInteractive(1,0)  # Turn off popups
ls.exportShef("C:/temp/FolsomShefData.shef", datasets)

#ls.finish()  # Batch mode only
```

The file *FolsomShefData.shef* will contain:

```
  .E FOLSOM 200906 Z DY0603102400/PR/DID1/0.79000002/0.31999999/0.25999999
  .E FOLSOM 200906 Z DY0603132400/PR/DID1/0.20999999/1.08/0.050000000/0.30000001
  .E FOLSOM 200906 Z DY0603172400/PR/DID1/0.76999998/0.039999990/0.020000000
  .E FOLSOM 200906 Z DY0603202400/PR/DID1/0.38999999/0.14/0.020000000/0.020000000
```

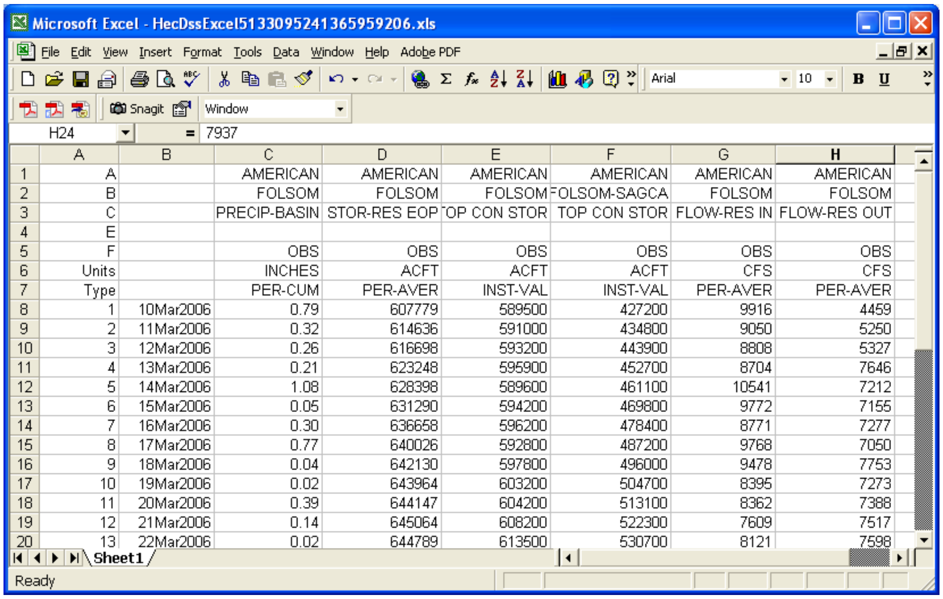
Instead of reading the data and passing to exportShef(), the time window could have been set and the pathnames set selected in ListSelection just as well.

Importing SHEF data is easy; you essentially call the function *importShef(String fileName)*, passing in the name of the file to import.

```
  from hec.script import *
  from hec.dssgui import *
  import java

  ls = ListSelection.getMainWindow()
  ls.setIsInteractive(1,0)  # Turn off popups
  ls.open("C:/temp/myDb.dss")
  ls.importShef("C:/temp/FolsomShefData.shef")
  #ls.finish()  # Batch mode only
```

C.3.3 Exporting to Excel

The export to Excel function uses code from HecDataTable; writing to Excel is similar to creating a table. The script **FolsomExcel.py** not only puts the data in an Excel file, but runs Excel too (Figure C.2):


Microsoft Excel - HecDssExcel5133095241365959206.xls
File Edit View Insert Format Tools Data Window Help Adobe PDF
Arial 10 B U
Snagit Window
Sheet1
Ready

H24 = 7937

	A	B	C	D	E	F	G	H
1	A		AMERICAN	AMERICAN	AMERICAN	AMERICAN	AMERICAN	AMERICAN
2	B		FOLSOM	FOLSOM	FOLSOM	FOLSOM-SAGCA	FOLSOM	FOLSOM
3	C		PRECIP-BASIN	STOR-RES EOP	TOP CON STOR	TOP CON STOR	FLOW-RES IN	FLOW-RES OUT
4	E							
5	F		OBS	OBS	OBS	OBS	OBS	OBS
6	Units		INCHES	ACFT	ACFT	ACFT	CFS	CFS
7	Type		PER-CUM	PER-AVER	INST-VAL	INST-VAL	PER-AVER	PER-AVER
8	1	10Mar2006	0.79	607779	589500	427200	9916	4459
9	2	11Mar2006	0.32	614636	591000	434800	9050	5250
10	3	12Mar2006	0.26	616698	593200	443900	8808	5327
11	4	13Mar2006	0.21	623248	595900	452700	8704	7646
12	5	14Mar2006	1.08	628398	589600	461100	10541	7212
13	6	15Mar2006	0.05	631290	594200	469800	9772	7155
14	7	16Mar2006	0.30	636658	596200	478400	8771	7277
15	8	17Mar2006	0.77	640026	592800	487200	9768	7050
16	9	18Mar2006	0.04	642130	597800	496000	9478	7753
17	10	19Mar2006	0.02	643964	603200	504700	8395	7273
18	11	20Mar2006	0.39	644147	604200	513100	8362	7388
19	12	21Mar2006	0.14	645064	608200	522300	7609	7517
20	13	22Mar2006	0.02	644789	613500	530700	8121	7598

Figure C.2 Export to Excel

If you would rather specify the Excel file name, you can do that by giving the name of the file following the datasets. You can also just create an Excel file, without running it, by using the function createExcelFile. This is shown in the script **FolsomSaveExcel.py**:

```
from hec.script import *
from hec.heclib.dss import *
from hec.dataTable import *
import java


#  Open the file and get the data
try:
  dssFile = HecDss.open("C:/temp/sample.dss", "10MAR2006 2400, 09APR2006
2400")
  precip = dssFile.get("/AMERICAN/FOLSOM/PRECIP-
BASIN/01JAN2006/1DAY/OBS/")
  stor = dssFile.get("/AMERICAN/FOLSOM/ STOR-RES EOP/01JAN2006/1DAY/OBS/")
  topcon = dssFile.get("/AMERICAN/FOLSOM/TOP CON
STOR/01JAN2006/1DAY/OBS/")
  sagca = dssFile.get("/AMERICAN/FOLSOM-SAGCA/TOP CON
STOR/01JAN2006/1DAY/OBS/")
  inflow = dssFile.get("/AMERICAN/FOLSOM/FLOW-RES IN/01JAN2006/1DAY/OBS/")
  outflow = dssFile.get("/AMERICAN/FOLSOM/FLOW-RES
OUT/01JAN2006/1DAY/OBS/")
except java.lang.Exception, e :
  #  Take care of any missing data or errors
   MessageBox.showError(e.getMessage(), "Error reading data")

#  Add Data
datasets = java.util.Vector()
datasets.add(precip)
datasets.add(stor)
datasets.add(topcon)
datasets.add(sagca)
datasets.add(inflow)
datasets.add(outflow)

#  For this code, jython sees a List before a Vector
#list = java.awt.List()
list = []
list.append(datasets)

table = HecDataTableToExcel.newTable()
```

```
#  If you want to run Excel with a specific name and not a temp name:
table.runExcel(list, "myWorkbook.xls")
#  Or, if you would just rather create an Excel file, but not run it:
#table.createExcelFile(datasets, "myWorkbook.xls")
```

C.3.4 Importing Other Formats

If you can parse a format, you can write a script to enter your data into HEC-DSS by using a DataContainer. DataContainers are generic database independent classes that hold data used by the various functions in HEC-DSSVue.

The following example, WriteDss.py, shows the use of a TimeSeriesContainer with made-up data. In this example, HecTime is used to generate the times needed and the TimeSeriesContainer is imported from hec.io:

```
from hec.script import *
from hec.heclib.dss import *
from hec.heclib.util import *
from hec.io import *
import java

try :
  try :
    myDss = HecDss.open("C:/temp/test.dss")
    tsc = TimeSeriesContainer()
    tsc.fullName = "/BASIN/LOC/FLOW//1HOUR/OBS/"
    start = HecTime("04Sep1996", "1330")
    tsc.interval = 60
    flows = [0.0,2.0,1.0,4.0,3.0,6.0,5.0,8.0,7.0,9.0]
    times = []
    for value in flows :
      times.append(start.value())
      start.add(tsc.interval)
    tsc.times = times
    tsc.values = flows
    tsc.numberValues = len(flows)
    tsc.units = "CFS"
    tsc.type = "PER-AVER"
    myDss.put(tsc)

  except Exception, e :
    MessageBox.showError(' '.join(e.args), "Python Error")
  except java.lang.Exception, e :
    MessageBox.showError(e.getMessage(), "Error")
finally :
  myDss.close()
```

C.3.5 Sample Graphics Scripts

Generally, graphics scripts may be broken into five segments; 1) Retrieve the data; 2) Initialize the plot; 3) Bring the plot into existence with *showPlot()*; 4) Change the plot; and 5) Save the plot to file and close.

Most people are tripped up with the function *showPlot()*. This function puts all the parts together and creates the plot. One would think that this would be called at the end of the script, but to the contrary, it needs to be called near the beginning. For example, you can not change or set a curve's color until the curve exists, and the showPlot() function is what creates curves. Scripting is emulating the steps that you would do interactively; it is not a *command* language.

HecDss uses *exceptions* for error processing, such as indicating missing data. You need to use *try: except* loops to catch errors, otherwise an exception message will be written to the output and that exception may not be very clear.

A simple plot (Figure C.3) example that illustrates these points is one for Oakville:

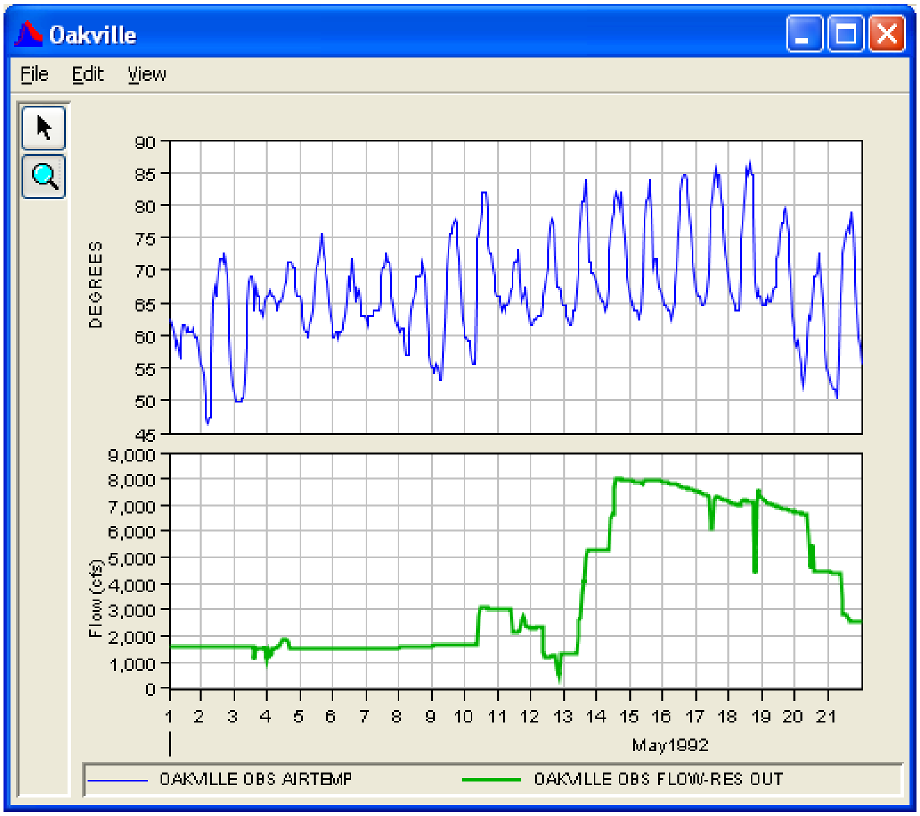


Figure C.3 Oakville Output - Plot

```
from hec.script import *       # always needed
from hec.heclib.dss import *   # Used to access HEC-DSS
import java                    #  Provide access to java

#  Open the file and get the data
try :
  dssFile = HecDss.open("C:/temp/sample.dss")
  airTemp = dssFile.get("/GREEN
RIVER/OAKVILLE/AIRTEMP/01MAY1992/1HOUR/OBS/")
  outflow = dssFile.get("/GREEN RIVER/OAKVILLE/FLOW-RES
OUT/01MAY1992/1HOUR/OBS/")
```

```
  dssFile.done()
except java.lang.Exception, e :
  #  Take care of any missing data or errors
   MessageBox.showError(e.getMessage(), "Error reading data")

#  Initialize the plot and add data
plot = Plot.newPlot("Oakville")
plot.addData(airTemp)
plot.addData(outflow)

#  Create plot
plot.setSize(600,600)
plot.setLocation(100,100)  #  Move off screen if you don’t want
to see it
plot.showPlot()

#  Change the plot
outCurve = plot.getCurve(outflow)
outCurve.setLineColor("darkgreen")

#  Save the plot and close
plot.saveToPng("C:/temp/Oakville.png")
#plot.close()  #  Do this if you only want the plot
#                 as a .png and not on screen
```

This example shows a more complex plot where we break the rule of *showPlot()* of creating the plot prior to changing it. In this case, *configurePlotLayout()* will also create the plot. In this plot of Folsom Lake (Figure C.4), we create three separate viewports and size them according to the different data types. Without too much work, we produce a pretty fancy plot.

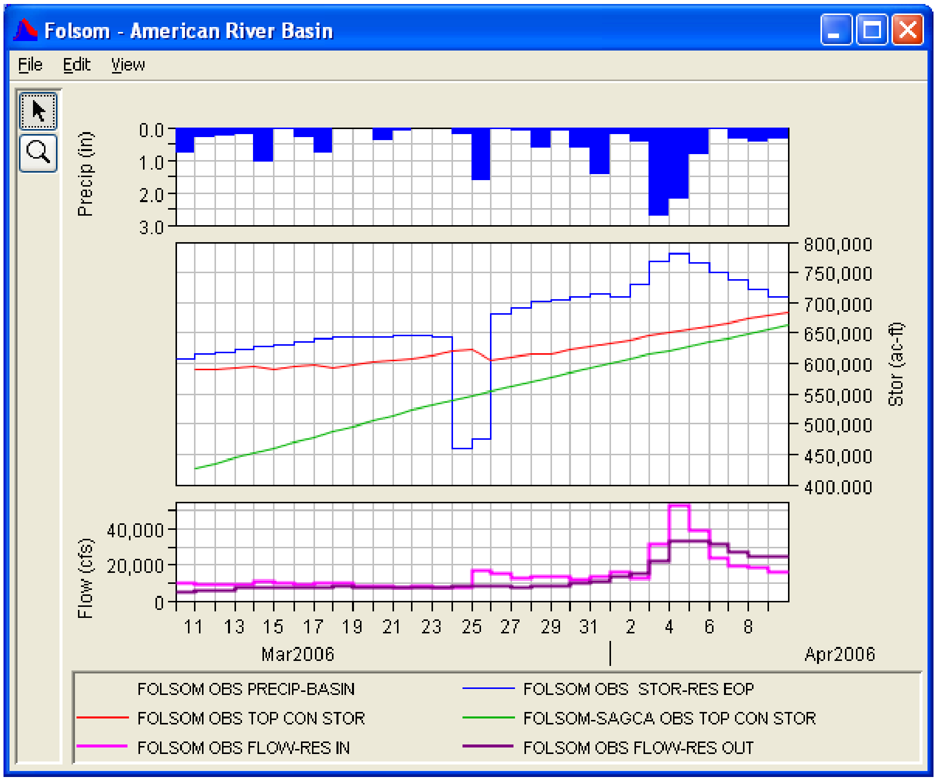


Figure C.4 Folsom Lake Plot

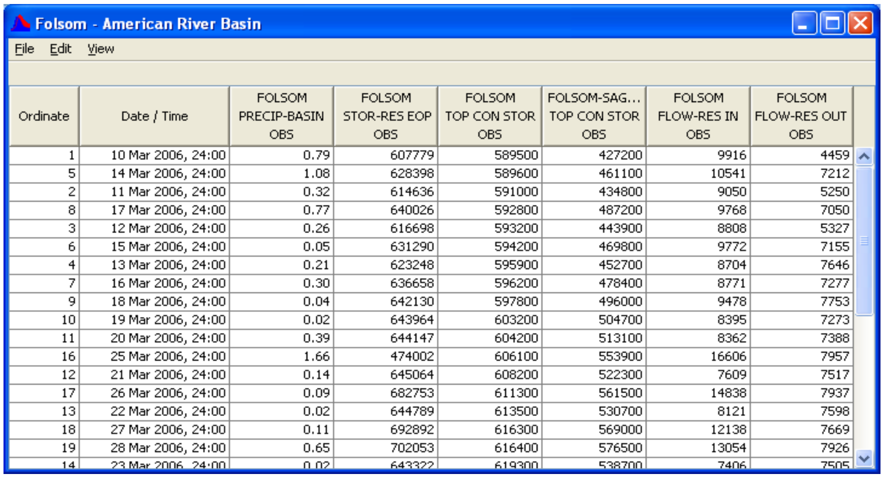
Folsom.py

```
from hec.script import *
from hec.script.Constants import TRUE, FALSE
from hec.heclib.dss import *
import java

#  Open the file and get the data
try:
  dssFile = HecDss.open("C:/temp/sample.dss", "10MAR2006 2400,
09APR2006 2400")
  precip = dssFile.get("/AMERICAN/FOLSOM/PRECIP-
BASIN/01JAN2006/1DAY/OBS/")
  stor = dssFile.get("/AMERICAN/FOLSOM/ STOR-RES
EOP/01JAN2006/1DAY/OBS/")
  topcon = dssFile.get("/AMERICAN/FOLSOM/TOP CON
STOR/01JAN2006/1DAY/OBS/")
  sagca = dssFile.get("/AMERICAN/FOLSOM-SAGCA/TOP CON
STOR/01JAN2006/1DAY/OBS/")
  inflow = dssFile.get("/AMERICAN/FOLSOM/FLOW-RES
IN/01JAN2006/1DAY/OBS/")
  outflow = dssFile.get("/AMERICAN/FOLSOM/FLOW-RES
OUT/01JAN2006/1DAY/OBS/")
except java.lang.Exception, e :
  #  Take care of any missing data or errors
   MessageBox.showError(e.getMessage(), "Error reading data")

#  Initialize the plot and set viewport size in precent
plot = Plot.newPlot("Folsom - American River Basin")
layout = Plot.newPlotLayout()
topView = layout.addViewport(10.)
middleView = layout.addViewport(60.)
bottomView = layout.addViewport(30.)

#  Add Data in specific viewports
topView.addCurve("Y1", precip)
middleView.addCurve("Y2", stor)
middleView.addCurve("Y2", topcon)
middleView.addCurve("Y2", sagca)
bottomView.addCurve("Y1", inflow)
bottomView.addCurve("Y1", outflow)

panel = plot.getPlotpanel()
prop = panel.getProperties()
prop.setViewportSpaceSize(0)

#  Break our first rule - actually this creates the plot to
change
plot.configurePlotLayout(layout)

panel = plot.getPlotpanel()
prop = panel.getProperties()
prop.setViewportSpaceSize(0)

#  Invert the precip and make pretty
view0 = plot.getViewport(0)
yaxis = view0.getAxis("Y1")
yaxis.setReversed(FALSE)
precipCurve = plot.getCurve(precip)
precipCurve.setFillColor("blue")
precipCurve.setFillType("Above")
precipCurve.setLineVisible(FALSE)
```

```
#  Set the inflow and outflow colors
inflowCurve = plot.getCurve(inflow)
inflowCurve.setLineColor("magenta")
outflowCurve = plot.getCurve(outflow)
outflowCurve.setLineColor("purple")

plot.showPlot()
```

C.4 Sample Table Script

Creating a table from a script is easier than plots as there is less to do. Like plots, you retrieve the data, add the data to a *Vector*, create a table and set the data, and then show the table. The same data shown in the plot is easily displayed in tabular form (Figure C.5) by the following **FolsomTable.py** script:



Ordinate	Date / Time	FOLSOM PRECIP-BASIN OBS	FOLSOM STOR-RES EOP OBS	FOLSOM TOP CON STOR OBS	FOLSOM-SAG... TOP CON STOR OBS	FOLSOM FLOW-RES IN OBS	FOLSOM FLOW-RES OUT OBS
1	10 Mar 2006, 24:00	0.79	607779	589500	427200	9916	4459
5	14 Mar 2006, 24:00	1.08	628398	589600	461100	10541	7212
2	11 Mar 2006, 24:00	0.32	614636	591000	434800	9050	5250
8	17 Mar 2006, 24:00	0.77	640026	592800	487200	9768	7050
3	12 Mar 2006, 24:00	0.26	616698	593200	443900	8808	5327
6	15 Mar 2006, 24:00	0.05	631290	594200	469800	9772	7155
4	13 Mar 2006, 24:00	0.21	623248	595900	452700	8704	7646
7	16 Mar 2006, 24:00	0.30	636658	596200	478400	8771	7277
9	18 Mar 2006, 24:00	0.04	642130	597800	496000	9478	7753
10	19 Mar 2006, 24:00	0.02	643964	603200	504700	8395	7273
11	20 Mar 2006, 24:00	0.39	644147	604200	513100	8362	7388
16	25 Mar 2006, 24:00	1.66	474002	606100	553900	16606	7957
12	21 Mar 2006, 24:00	0.14	645064	608200	522300	7609	7517
17	26 Mar 2006, 24:00	0.09	682753	611300	561500	14838	7937
13	22 Mar 2006, 24:00	0.02	644789	613500	530700	8121	7598
18	27 Mar 2006, 24:00	0.11	692892	616300	569000	12138	7669
19	28 Mar 2006, 24:00	0.65	702053	616400	576500	13054	7926
14	23 Mar 2006, 24:00	0.02	643322	619300	538700	7406	7505

Figure C.5 Folsom Lake Tabular Display

```
from from hec.script import *
from hec.heclib.dss import *
from hec.dataTable import *
import java

#  Open the file and get the data
try:
  dssFile = HecDss.open("C:/temp/sample.dss", "10MAR2006 2400,
09APR2006 2400")
  precip = dssFile.get("/AMERICAN/FOLSOM/PRECIP-
BASIN/01JAN2006/1DAY/OBS/")
  stor = dssFile.get("/AMERICAN/FOLSOM/ STOR-RES
EOP/01JAN2006/1DAY/OBS/")
  topcon = dssFile.get("/AMERICAN/FOLSOM/TOP CON
STOR/01JAN2006/1DAY/OBS/")
  sagca = dssFile.get("/AMERICAN/FOLSOM-SAGCA/TOP CON
STOR/01JAN2006/1DAY/OBS/")
  inflow = dssFile.get("/AMERICAN/FOLSOM/FLOW-RES
IN/01JAN2006/1DAY/OBS/")
  outflow = dssFile.get("/AMERICAN/FOLSOM/FLOW-RES
OUT/01JAN2006/1DAY/OBS/")
except java.lang.Exception, e :
  #  Take care of any missing data or errors
   MessageBox.showError(e.getMessage(), "Error reading data")
```

```
       #  Add Data
datasets = java.util.Vector()
datasets.add(precip)
datasets.add(stor)
datasets.add(topcon)
datasets.add(sagca)
datasets.add(inflow)
datasets.add(outflow)

table = HecDataTableFrame.newTable("Folsom - American River Basin")
table.setData(datasets)
table.showTable()
```

C.5 Complex Graphics Scripts

A report-ready script for displaying monthly reservoir data and a script that uses arguments to define what is to be plotted are presented in the following sections.

Note: *An important detail to keep in mind is that you need to call showPlot() early on. Scripting emulates what you would do interactively; it is not a "command driven" procedure. You cannot change features on a plot until they exist and the showPlot() function causes the plot object to come into existence.*

C.5.1 Coyote Valley Dam Reservoir Plot

This is a plot showing reservoir data for the month of March (Figure C.6). The plot shows precipitation, storage, top of conservation, inflow, outflow and downstream flows.

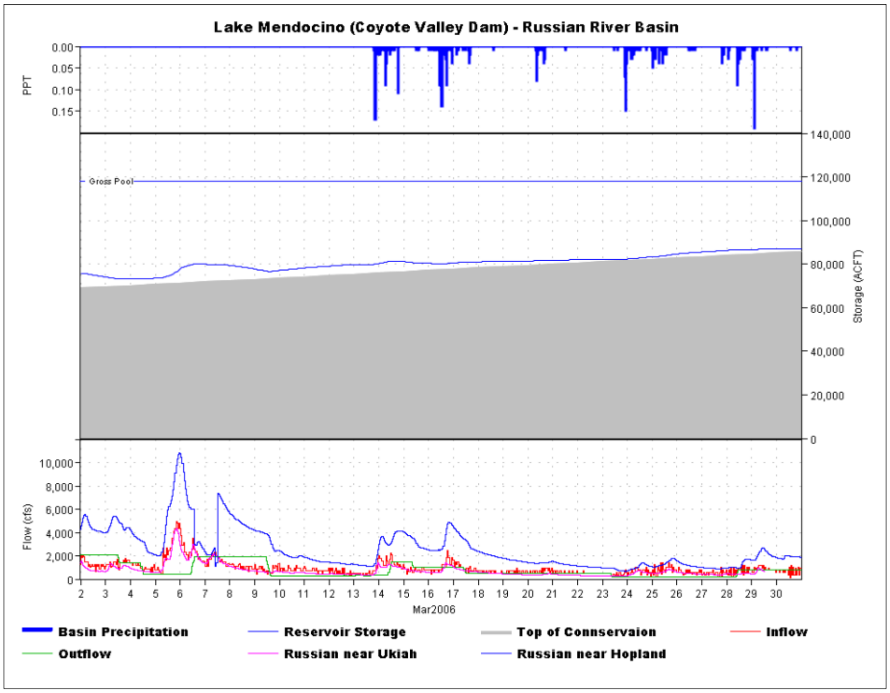


Figure C.6 Mendocino Reservoir Data for March

The script for Coyote Valley is:

```
from hec.script import *
from hec.script.Constants import TRUE, FALSE
from hec.heclib.dss import *
import java

#  grid pattern - 2 solid pixels followed by 8 blank ones
dotPat = [2., 8.]
datasets = []

#  Get the data
Try :
  dssFile = HecDss.open("C:/temp/sample.dss", "01MAR2006 2400,
30MAR2006 2400")
  precip = dssFile.get("/EF RUSSIAN/COYOTE/PRECIP-
INC/01MAR2006/1HOUR/TB/")
  datasets.append(precip)
  stor = dssFile.get("/EF RUSSIAN/COYOTE/STOR-RES
EOP/01MAR2006/1HOUR//")
  datasets.append(stor)
  topcon = dssFile.get("/EF RUSSIAN/COYOTE/TOP CON
STOR/01JAN2006/1DAY//")
  datasets.append(topcon)
  inflow =dssFile.get("/EF RUSSIAN/COYOTE/FLOW-RES
IN/01MAR2006/1HOUR/SMOOTH/")
  datasets.append(inflow)
  outflow = dssFile.get("/EF RUSSIAN/COYOTE/FLOW-RES
OUT/01MAR2006/1HOUR//")
  datasets.append(outflow)
  ukiah = dssFile.get("/RUSSIAN/NR UKIAH/FLOW/01MAR2006/1HOUR//")
  datasets.append(ukiah)
  hopland = dssFile.get("/RUSSIAN/NR HOPLAND/FLOW/01MAR2006/1HOUR//")
  datasets.append(hopland)
      except java.lang.Exception, e :
       MessageBox.showError(e.getMessage(), "Error reading
data")

#  Create the view ports
plot = Plot.newPlot("Lake Mendocino")
layout = Plot.newPlotLayout()
topView = layout.addViewport(10.)
middleView = layout.addViewport(60.)
bottomView = layout.addViewport(30.)

#  Add the data
topView.addCurve("Y1", precip)
middleView.addCurve("Y2", stor)
middleView.addCurve("Y2", topcon)
bottomView.addCurve("Y1", inflow)
bottomView.addCurve("Y1", outflow)
bottomView.addCurve("Y1", ukiah)
bottomView.addCurve("Y1", hopland)
plot.configurePlotLayout(layout)

#  For headless operation (batch mode), draw the plot off the screen
plot.setLocation(-10000, -10000)
plot.setSize(1000, 800)

####### Important – showPlot() creates the plot objects ######
# (You cannot set or change things that do not exist yet)
plot.showPlot()
```

```
#  Make the legend labels look nice
for dataset in datasets:
label = plot.getLegendLabel(dataset)
label.setFontSize(16)
label.setFont("Arial Black,Plain,14")
label.setForeground("black")

# Set the label text
label = plot.getLegendLabel(precip)
label.setText( "Basin Precipitation")
label = plot.getLegendLabel(stor)
label.setText("Reservoir Storage")
label = plot.getLegendLabel(topcon)
label.setText("Top of Connservaion")
label = plot.getLegendLabel(inflow)
label.setText("Inflow")
label = plot.getLegendLabel(outflow)
label.setText("Outflow")
label = plot.getLegendLabel(ukiah)
label.setText("Russian near Ukiah")
label = plot.getLegendLabel(hopland)
label.setText("Russian near Hopland")

#  Set the plot title
plot.setPlotTitleText("Lake Mendocino (Coyote Valley Dam) - Russian
River Basin")
tit = plot.getPlotTitle()
tit.setFont("Arial Black")
tit.setFontSize(18)
plot.setPlotTitleVisible(TRUE)

#  Make the viewports right next to each other
panel = plot.getPlotpanel()
panel.setHorizontalViewportSpacing(0)

#  Set the Y axis labels
view0 = plot.getViewport(0)
yaxis = view0.getAxis("Y1")
yaxis.setReversed(FALSE)
yaxis.setLabel("PPT")
view1 = plot.getViewport(1)
yaxis1 = view1.getAxis("Y2")
yaxis1.setLabel("Storage (ACFT)")
yaxis1.setScaleLimits(0., 140000.)
yaxis1.setViewLimits(0., 140000.)

#  Mark the gross pool level
marker = AxisMarker()
marker.axis = "Y"
marker.value = "118000"
marker.labelText = "Gross Pool"
marker.lineColor = "Blue"
marker.labelPosition = "center"
view1.addAxisMarker(marker)

#  Set the curve colors and fill
precipCurve = plot.getCurve(precip)
precipCurve.setFillColor("blue")
precipCurve.setFillType("Above")
precipCurve.setLineColor("blue")

conCurve = plot.getCurve(topcon)
conCurve.setFillColor("lightGray")
conCurve.setFillType("Below")
conCurve.setLineColor("lightGray")

storCurve = plot.getCurve(stor)
storCurve.setLineColor("blue")
storCurve.setLineWidth(1.)


outCurve = plot.getCurve(outflow)
       outCurve.setLineColor("darkgreen")
       outCurve.setLineWidth(1.)

inCurve = plot.getCurve(inflow)
inCurve.setLineColor("red")
inCurve.setLineWidth(1.)

ukiahCurve = plot.getCurve(ukiah)
ukiahCurve.setLineColor("magenta")
ukiahCurve.setLineWidth(1.)

hoplandCurve = plot.getCurve(hopland)
hoplandCurve.setLineColor("blue")
hoplandCurve.setLineWidth(1.)

#  Set grid style
view0 = plot.getViewport(0)
prop = view0.getProperties()
prop.setMajorXGridStyle(dotPat)
prop.setMajorYGridStyle(dotPat)
view1 = plot.getViewport(1)
prop = view1.getProperties()
prop.setMajorXGridStyle(dotPat)
prop.setMajorYGridStyle(dotPat)
view2 = plot.getViewport(2)
prop = view2.getProperties()
prop.setMajorXGridStyle(dotPat)
prop.setMajorYGridStyle(dotPat)

#  Now that it is complete, save to a png and close it
plot.saveToPng("C:/temp/Coyote.png")
plot.close()
```

C.5.2 Scripts with Arguments

Often you will want to setup a plot or other function and apply it to different locations, variables or times. Instead of duplicating the script for each instance, which leads to maintenance and manageability issues, it is usually advantageous to write one script and pass in arguments that may vary.

To access arguments passed into the script you must include the following: import sys (sm = globals())

Arguments are passed into a python script using a "name-space dictionary", which gives the keyword followed by a colon and then the parameter. Inside the script, the keyword will be replaced with the parameter. For example, in the GagePlot script, the name-space dictionary could be:

```
"location" : "Glenfir", "version" : "OBS"
```

Wherever the string location or version is in the script will be substituted with GlenFir (Figure C.7) and Obs, respectively.

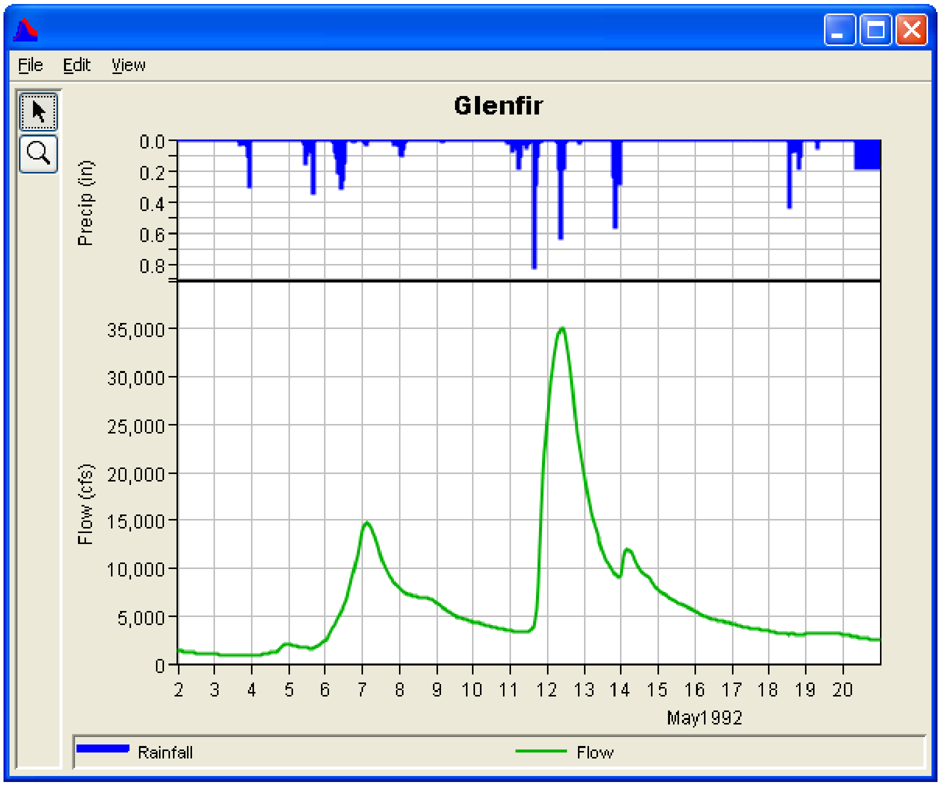


Figure C.7 Glenfir Plot

The GagePlot script is designed to produce a standard plot for a gage where there is both a stream flow gage and a precipitation gage. Here you can have one script that you call from another script multiple times, with a different argument for the location. The individual scripts may be run using the function "exec" for in-line scripts or "execFile" for scripts on disk. For example, a RunGages script can be written to run GagePlot for several locations:

Script RunGages

```
from hec.script import *

glenFirArgs  = {"location" : "Glenfir", "version" : "OBS"}
execfile("C:/HecDSSVueDev/HecDssVue/scripts/GagePlot.py", {},
glenFirArgs)

madronArgs   = {"location" : "Madron", "version" : "OBS"}
execfile("C:/HecDSSVueDev/HecDssVue/scripts/GagePlot.py", {},
madronArgs)

oakTreeArgs   = {"location" : "Madron", "version" : "OBS"}
execfile("C:/HecDSSVueDev/HecDssVue/scripts/GagePlot.py", {},
oakTreeArgs)
```

The **GagePlot** Script is:

```
# Both location=name version=ver must be defined as arguments
# To run:  GagePlot location=Glenfir version=OBS

from hec.script import *
from hec.heclib.dss import *
import sys
import java
from hec.script.Constants import TRUE, FALSE

# Access arguments
sm = globals()

#  "location" and "version" will be replaced with the arguments
passed in

#  Retrieve data
try :
  #  If you wanted to use a relative time window, you could do
something like:
  #  dssFile = HecDss.open("C:/temp/sample.dss", "T-30D, T")
  dssFile = HecDss.open("C:/temp/sample.dss","01MAY1992
2400",”20MAY1992 2400")
  flowPath = "/GREEN RIVER/" + location + "/FLOW//1HOUR/OBS/"
  precipPath = "/GREEN RIVER/" + location + "/PRECIP-
INC//1HOUR/OBS/"
  precip = dssFile.get(precipPath)
  flow = dssFile.get(flowPath)
except java.lang.Exception, e :
    MessageBox.showError(e.getMessage(), "Error reading data")

# Create plot and viewports
plot = Plot.newPlot()
layout = Plot.newPlotLayout()
topView = layout.addViewport(15.)
bottomView = layout.addViewport(85.)

#  Add data
topView.addCurve("Y1", precip)
bottomView.addCurve("Y1", flow)
plot.configurePlotLayout(layout)
plot.setSize(600, 500)

#  Add title
plot.setPlotTitleText(location)
tit = plot.getPlotTitle()
tit.setFont("Arial Black")
tit.setFontSize(18)
plot.setPlotTitleVisible(TRUE)

#  This actually creates the plot - cannot access any components
until done
plot.showPlot()

#  Now we can change components
panel = plot.getPlotpanel()
#  Remove space between viewports
panel.setHorizontalViewportSpacing(0)

#  Invert the precip
view0 = plot.getViewport(0)
yaxis = view0.getAxis("Y1")
yaxis.setReversed(FALSE)
```

```
#  Make the precip pretty
precipCurve = plot.getCurve(precip)
precipCurve.setFillColor("blue")
precipCurve.setFillType("Above")
precipCurve.setLineColor("blue")

#  Make the flow pretty
flowCurve = plot.getCurve(flow)
flowCurve.setLineColor("darkgreen")
flowCurve.setLineWidth(2.)

#  Set the legend labels
plot.setLegendLabelText(flow, "Flow")
plot.setLegendLabelText(precip, "Rainfall")

# Done (should look like we want now)

#  If you wanted this to run in the background, you could add
the following
#  plot.setLocation(-10000, -10000)
#  plot.saveToPng("C:/temp/GagePlot.png")
#  plot.close()
```

Another example of calling scripts from scripts is the **script tester.py**:

```
from hec.script import *
import time
import java

#---------------------#
# some code to eval() #
#---------------------#
getCurrentTimeSnippet = "time.ctime(time.time())"

#-----------------#
# a simple script #
#-----------------#
scriptText = '''
def myFunc() :
     global startTime, endTime
     startTime = time.time()
     for i in range(3) :
            print("Sleeping for 1 second at %f" % time.time())
            time.sleep(1)
     endTime = time.time()

myFunc()
print("Start time = %s" % time.ctime(startTime))
print("End time   = %s" % time.ctime(endTime))
'''

#---------------------------------#
# write the script out to a file #
#---------------------------------#
scriptFilename = "myFunc.py"
scriptFile = open(scriptFilename, "w")
scriptFile.write(scriptText)
scriptFile.close()

#----------------------------------#
# Use eval() to evaluate some code #
#----------------------------------#
```

```
print("\nCalling eval()")
print("The local time is %s" % eval(getCurrentTimeSnippet))

#---------------------------------------#
# set up a separate scope for execution #
#---------------------------------------#
otherGlobals = {"time" : time, "startTime" : None, "endTime" :
None}
otherLocals  = {}

#------------------------------------#
# exec some text in a separate scope #
#------------------------------------#
print("\nCalling exec in a separate scope")
exec scriptText in otherGlobals, otherLocals
try    : print("Global variable startTime = %f" % startTime)
except : print("Global variable  startTime is not defined.")
try    : print("Global variable endTime   = %f" % endTime)
except : print("Global variable  endTime is not defined.")

#---------------------------------------------#
#execfile the same script in a separate scope #
#---------------------------------------------#
print("\nCalling execfile in a separate scope")
execfile(scriptFilename, otherGlobals, otherLocals)
try    : print("Global variable startTime = %f" % startTime)
except : print("Global variable  startTime is not defined.")
try    : print("Global variable endTime   = %f" % endTime)
except : print("Global variable  endTime is not defined.")

#-------------------------------------#
# exec the same text in our own scope #
#-------------------------------------#
print("\nCalling exec in the current scope")
exec scriptText
try    : print("Global variable startTime = %f" % startTime)
except : print("Global variable  startTime is not defined.")
try    : print("Global variable endTime   = %f" % endTime)
except : print("Global variable  endTime is not defined.")

#------------------------------------------#
#execfile the same script in our own scope #
#------------------------------------------#
print("\nUndefining global variables startTime and endTime")
del globals()['startTime']
del globals()['endTime']
print("\nCalling execfile in the current scope")
execfile(scriptFilename)
try    : print("Global variable startTime = %f" % startTime)
except : print("Global variable  startTime is not defined.")
try    : print("Global variable endTime   = %f" % endTime)
except : print("Global variable  endTime is not defined.")
```