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/
```