Unified Transportation Sensor Data Format (UTSDF): Introduction

By Dr. Taek M. Kwon

Transportation Data Research Laboratories (TDRL)
Northland Advanced Transportation Systems Research Laboratories (NATSRL)
University of Minnesota Duluth

Feb, 2004

1. Introduction

Today's transportation systems have been increasingly utilizing a wide range of sensors to monitor, control, and analyze many parts of transportation system components. The sensor usage has been further accelerated by the US DOT's emphasis on Intelligent Transportation Systems (ITS) technologies in recent years. On the other hand, while the usage of sensors has been increased, archiving (or saving) of the sensor data has not been increased at the same rate [1]. In fact, only a small fraction of sensor data from the today's transportation systems has been archived. For example, each intersection typically includes a number of vehicle detecting sensors to optimize the timing of the traffic controller, but the data is rarely archived.

There are a number of reasons that archiving of transportation sensor data (TSD) has not been eagerly pursued by transportation departments. First and the most influential factor is the cost, i.e., while the cost of sensors is low, the cost of archiving their data is expensive. As a result, archiving has been frequently unwelcome by maintenance engineers and managers. Second, continuous flow of data from transportation sensors adds an additional burden to the management of data and archiving. Sensors continuously generate data once they are activated, and the amount of data can be quickly accumulated to a large amount. Moreover, all parts of the data acquisition system must continuously work without faults. In effect, archiving often reveals the weakness or reliability of the system, which sometimes is not a pleasant thing. For maintenance personnel, archiving is an additional work but it can also lead to a substantial pressure because missing or lost data can be a responsibility. Third, when data is collected from many different types of sensors, which would be the case in Road Weather Information Systems (RWIS), management of data is complicated. To date there exists no uniform and efficient data format that can be used for archiving all types of sensor data. As a result, management of data from various sensors and data acquisition systems developed by many different types of manufacturers is in itself a challenge. For example, it requires a large amount of work to keep up with the data format differences and modifications, incompatible file formats, version changes of software tools and operating systems, etc. Therefore, acquisition, archiving, and maintenance of data from a large number of sensors used in today's transportation systems are often more difficult than what individual sensor shows.

The next question is then "Why do we need archiving?" or "Do we really need archiving?" The answer would depend upon the needs. However, if we assume that system analysis (performance, reliability, etc.) is needed at some point in the future, archiving of data would be a necessary part since analyzing a system without data is difficult and unreliable. Therefore, the more important question on archiving is not in the need of or not, but in what extent, i.e., which selected locations, what sensors, and how long the data should be archived. The Unified Transportation Sensor Data Format (UTSDF) introduced in this documentation is an

attempt towards making TSD collection and archiving simple and easy regardless of the number of sensors, sensor types, and variability such as location change or removal.

Our main objective of developing UTSDF was to simplify the archiving task of TSD. An important step in this process is to create a uniform and efficient data format that is simple and independent of operating systems and programming languages. The users of transportation sensors should only need to learn a single type of data format for archiving and for the use of the archived data. Based on the UTSDF we also intend to build reliable data acquisition models and efficient methodologies for archiving large-scaled TSD such as a statewide system.

At our Transportation Data Research Laboratory (TDRL), the need for the development of UTSDF was born out of the needs in developing statewide archives of TSD that have characteristics of large scaled data and variety of data types including non-numeric data. In developing UTSDF, the following list was set as the objectives.

- A single unified data format for all types of transportation sensors
- Simple to understand and use
- Easy to manage
- Compatible with all types of computers, OS, and programming languages
- Easy to distribute or share large amount of data
- Compact, compressed form
- No or low cost in adopting the technology
- Fast and easy retrieval of a large amount of data from the archived data
- Adaptable for changes in sensor locations or configuration
- Inclusion of description of data (meta data)

This documentation describes the format of UTSDF and archiving methodologies that could be applied for statewide TSD archives.

2. Assumption on Transportation Sensor Data (TSD)

We refer all types of sensors (electrical, magnetic, mechanical, optical, chemical, etc) that are used in transportation systems as transportation sensors. Transportation sensors are typically used in monitoring the state or conditions of a transportation system component and often placed under the pavement or near the roadways. The digitized values or decision results of the sensor state comprise the sensor data. We assumed that all sensor readings are obtained from a fixed sampling rate, which would be the case for the most of the real-world TSD. For example, if traffic counting was done at every 30-second interval, we expect 2,880 data points per day. We further assume that the sampling rate is determined based on the sampling theorem, i.e. twice the bandwidth (also called a Nyquist rate) of the original signal [2]. If sensor readings are sampled at a Nyquist rate, the sampling theorem guarantees that the complete original signal can be reconstructed from the sampled data [2]. Consequently, it is assumed that re-sampling is possible from the reconstructed signal without loss of information.

Some sensors do not produce numerical values but descriptive conditions. For example, pavement sensors produce pavement conditions such as wet, dry, ice, etc. As long as those readings are recorded at a fixed rate, the data can be stored in UTSDF. A single sensor may

produce multiple types of values. For example, a single inductive loop detector produces two types of values, volume and occupancy. In order to differentiate between the sensor and values produced, we refer each type of sensor values as parameter, i.e., volume and occupancy are parameters of inductive loop detectors. These parameters are the final data (or variables) that are stored as sensor values in UTSDF.

3. Basic UTSDF Archive File

A single UTSDF archive file (or simply called UTSDF file) is a zip-compressed archive file of many small data files called *daylets* (described in the Section 4). More specifically, a single UTSDF file is created based on the time unit of a single day, in which it is a collection of *deaylets* from the same day and adheres the following name convention:

yyyymmdd.Class_Name

where the date of the archived data is encoded as the file name with eight digits, i.e., yyyy is the year, mm is the month, and dd is the day. The Class_Name is the name of the sensor class such as RWIS, traffic, or WIM (Weigh-in-Motion). For example, an RWIS archive file on Feb 23, 2003 would have the name *20030223.rwis*. Similarly, the traffic file on the same day would have the name *20030223.traffic*. As a result, when the archived files are viewed as a sorted list, it should be in a chronological order. In general, different classes of the archived files are stored in separate directories, and one year of complete RWIS or traffic archive would consist of 365 UTSDF files. The structure of a single UTSDF file is illustrated as a block diagram in Figure 1. The size of a daylet would depend on the type of parameters it stores and described in Section 4.

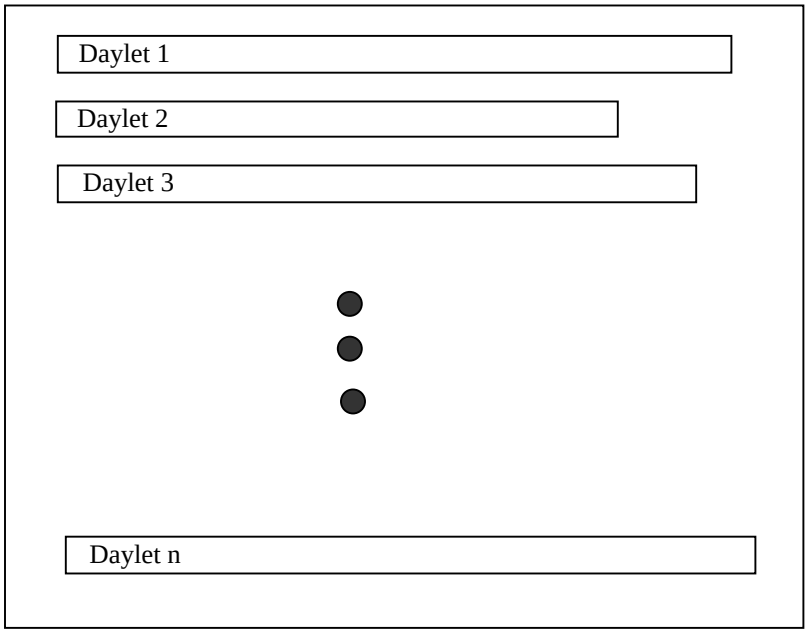

Figure 1: UTSDF archive consisting of n daylets. The number of daylets in an archive file would depend on the different types of parameters and the number of sensor locations.

4. Daylets

Daylets are the basic components of UTSDF file and contain the actual sensor data. The name of each daylet is assigned based on the spatial information (i.e., location) and the parameter type of the sensor, while the name of UTSDF file is assigned using the temporal information (date) and the sensor class name. The reason behind the choice of this name convention is to utilize the temporal and spatial properties of TSD, which are discussed in [3]. The basic name format consists of several fields separated by dots and is shown below:

SysID.SiteID.SensorID.ParaName

SysID: System ID. It is a unique number assigned for system characteristics such as a manufacturer type.

SiteID: Site ID. It is a unique numeric number assigned to each site based on the geographical location of sensor.

SensorID: Sensor ID. It is a unique number assigned for multiple sensors of identical types within a site (same location). For example, if three pavement temperature sensors are installed at a site, the sensors are assigned with SensorID, 0, 1, and 2.

ParaName: Parameter Name. It is a shortened parameter name without any space. For example, air temperature is shortened as "atemp" in this field. Please refer to Appendix A and B for RWIS and traffic data.

UTSDF itself does not define each field. The four-field name convention is provided as a recommendation. It is the archive provider's responsibility that each field is defined, documented, and provided along with the archive. An example documentation of theses definitions are provided in Appendix A and B based on the actual UTSDF archives of the Mn/DOT (Minnesota Department of Transportation) RWIS and traffic data used in TDRL.

For illustration, the name convention of daylets for statewide RWIS used in Mn/DOT data is used. Say, we wish to create a daylet for air temperature at System ID=330, Site ID=17, and Sensor ID=0, then the daylet's name would be assigned as "330.17.0.atemp".

If the statewide system consists of only one type of systems and no duplicated sensors exist in a single site, the first three fields can be combined as a single field of site ID number, but this will limit the flexibility and future extensibility. Again, a proper documentation must be provided for the definition of the daylet's name. At a minimum two fields must exist, i.e. the Site ID and the Parameter Name to be qualified as UTSDF.

The content of daylet files is simply a long string of ASCII characters that represent a single day data for the parameter it stores. Use of ASCII string provides excellent portability and allows storage of both numeric and non-numeric data. Each datum within a daylet must have the same length (the same number of characters per datum), so that the string length of a daylet is always computed by the datum length multiplied by the number of data items in the daylet. If a null datum exists, repetition of "N" characters for the allocated datum length is entered. Repetition of the same characters for null data is later efficiently compressed by the compression process. Since each datum has the same length, sampling period is precisely determined by dividing 24 hours by the number of data entries in the daylet, or vice versa. For example, if wind direction data is sampled at every 10 minutes and three digits are allocated for each datum to represent an angle in clockwise degrees from north, then the total string length of the wind-direction daylet would be 432 and it contains 144 data entries. Time stamp is not entered for each

datum since we assume that all data is sampled at a fixed sampling rate within that day. We assume that data can be always reconstructed from the sampled data and resampled to produce data for any time of the day based on the sampling theorem [1]. For the negative numbers, a single character " – " is used as a prefix, but positive numbers do not use any prefix character.

Example) Suppose that air temperature was sampled from a sensor for a single day with a 10 minute sampling period. The data collected from the sensor are degrees in Celsius and shown below:

```
00:00  27.5
00:10  10.5
00:20  5.8
00:30  N/A        ; missing data
00:40  0.5
   .
   .
   .
23:40  -13.5
23:50  -10.5
23:50  -5.5
```

Suppose that four digits are allocated for each datum representing a unit of one-tenth degrees in Celsius. Then, the string for the above data is packed as an ASCII string by simply concatenating four digit numbers in chronological order, i.e.,

```
027501050058NNNN0005…-135-100-055
```

When this string is saved as a file with the predefined daylet name, it becomes the daylet for the air temperature for the given date of the UTSDF file. □

In the daylet ASCII string, it is important to notice that no line breaks, commas, or spaces are used to separate the data. Such data separators are not needed, since we are using the same number of ASCII characters for each datum within a daylet. One may concern about the increased size of the data due to the use of ASCII string and fixed length. However, since the daylets are later zip-compressed to an archive file, the size of the initial data should not be of concern. According to our study, zip compression algorithm efficiently compresses the ASCII strings with fixed data length. The size was often smaller than the zip-compressed result of the equivalent size of binary data [3].

One advantage of using daylets in archiving is that since each daylets are independent each other in terms of storage, they can be easily added or removed without any modification in overall data structure. For example, as more sensors are installed at new locations or removed from old location, daylets can be simply added or removed. This parallelism with hardware makes the overall management of the archive simple.

5. Log and missing information

Each UTSDF archived file includes two special files. They are yyyymmdd.missing and yyyymmdd.log files where yyyymmdd is the numerical values of the year, month, and day of the archived date. The yyyymmdd.missing file includes the list of names of missing daylets (null data for the entire day) on that day. The daylet list is separated by a comma.

The yyyymmdd.log file serves as meta data and includes information about the data in the archive file. The format of this file is not defined except that ASCII characters should be used.

The archive provider should supply the documentation on the *.log file where the detailed format should be defined. Any information the user of the archive must know, should be stored in this file such as the number of active sites, the sites in out of service, special events, etc.

6. Data Compression

A UTSDF archive file is simply a zip-compressed file of daylets. When a single UTSDF file is uncompressed (unzipped), it should reproduce all of the original daylets that were compressed into a single archive file. Since most unzip tools allow unzipping of a single or just few files, retrieval of only needed daylets can be easily done from a UTSDF file.

Zip compression uses a compression algorithm referred to as the Deflate. Deflate combines the LZ77 algorithm [4] for marking common sub-strings and Huffman coding [5] to take an advantage of the different frequencies of occurrence of byte sequences in the file. Deflate does have an important advantage in that it is **not patented** (no need to obtain licenses). Thus, it is presently the most widely used file compression method. It is used in the WinZip™ freeware in Windows™ and the gzip program in Unix, and the jar files in Java. The Deflate algorithm is also a standard for the Internet IP payload compression (RFC 2394). Today, the term, zip or unzip, is commonly used replacing the algorithm name Deflate. For programmers, free source codes are available from Internet for zip and unzip. Also, many convenient commercial software tools, such as dynaZip, Sax.net, Xceed, ComponentOne, etc., are available for embedding unzip or zip function into application programs. At TDRL, a freeware WinZip™ and DynaZip™ utilities have been used as the basic tool for compression and decompression.

7. Organization of Archive Directories

The recommended organization of UTSDF archives is a hierarchical organization based on a file directory structure. File directory structure (or system) has been successfully used in storing all types of data since the beginning of the computer age and has proven very effective in handling large complicated data. There are a number of benefits in using a file system as the structure of archive organization. First, file system is such a familiar form to any computer users that it is probably one of the easiest structures to understand and manage. Second, it is one of the most stable and reliable parts of any computer operating system. Third, temporal, spatial, and computational hierarchies of the TSD properties nicely fit into the hierarchical nature of the file directory structure [3].

The organization of archives should be based on clarity and efficiency in retrieval of the data. We will consider organization of two types of common transportation sensor data, i.e., RWIS and traffic data. First, consider that we wish to build a statewide archive for traffic data. Since the number of traffic detectors used in a state is such a huge number, it is convenient to divide the data into districts to form a reasonable size of the archive files. Within each district, the sensors can then be given unique ID numbers or can be organized using dot separated fields as shown in Section 4. Each district directory is then further divided into year directories where daily UTSDF archives are stored. This directory structure is simply and utilizes the division of data that we are familiar with, which has benefits of clarity. This directory structure is illustrated in Figure 2. In this case, if the district, year, date, and the detector ID number are known, the data can be quickly searched. Notice that the spatial and temporal hierarchies of traffic data properties are alternatively utilize in the directory tree.

Next, we consider a statewide RWIS archive. Since the number of RWIS stations in a state is typically less than 1,000 and the stations are centrally managed, dividing them into districts can result in small fragmented archived files. Too many fragmented archived files are

less efficient in sharing data. Therefore, it is more logical to organize the archive directories into simply year directories as shown in Figure 2. In this case, each UTSDF archive file would contain RWIS daylets for the entire state for a single day. TDRL presently uses this organization to archive the Mn/DOT's statewide RWIS data. However, if the number of stations in a state were very large such as exceeding 5,000s, then dividing the directories into Districts would be more sensible. Again, the overall structure utilizes temporal and spatial relations since daylet's names are organized based on spatial relations.

One important part of the UTSDF directory structure is the inclusion of /docs directory at the next to the root level as shown in Figure 2. In the /docs directory, the archive provider should include all documentations necessary to understand the archive. It helps the users of the archive as well as the maintenance. The documentation could include daylet field name definitions, string length allocated for each parameter, basic units, sensor locations, sensor manufacturer information, maintenance history, addition or removal of sensors, etc. Inclusion of the /docs directory follows the sprit of the inclusion of a log file inside the daily UTSDF archive file, i.e., description of the data is provided at multiple levels, directory level and daylet level.

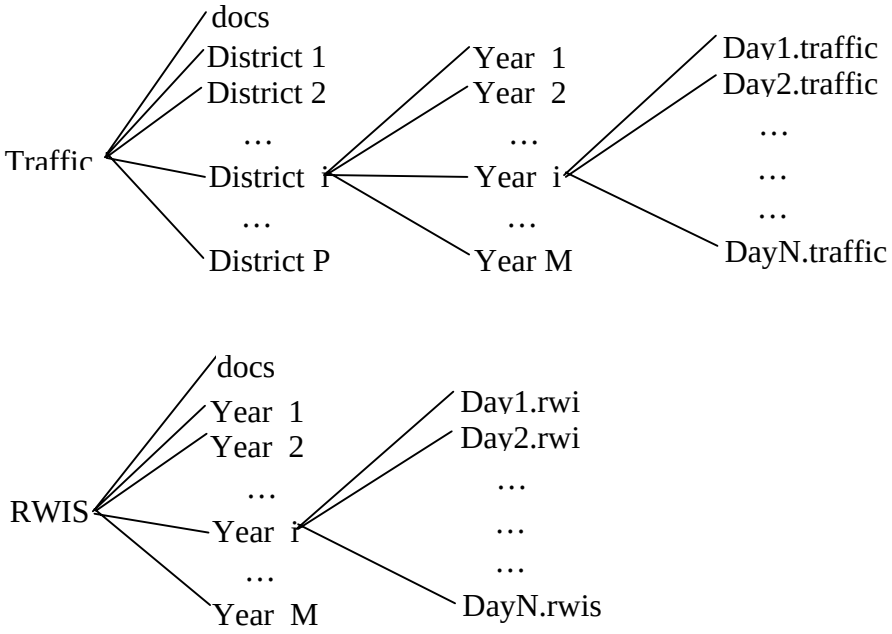

Figure 2: Directory structure of statewide UTSDF archives: RWIS and traffic data example

8. More Complex Structure of UTSDF: *Monthlets* and *Yearlets*

Until now, we only discussed archiving of raw sensor data in which daily operation of archiving is assumed, where daylets are utilized. However, we frequently need processed data such as Average Annual Daily Traffic (AADT) or daily average/low/high of pavement temperatures. For those processed data, expressing the data in a larger time scale is necessary such as a year rather than a single day. This type of needs can be met using *monthlets* and *yearlets*, which are similar to daylets except that they contain a whole month of data or a whole year of data.

Unlike daylets, *monthlets* and *yearlets* would require multiple parameters in a single file. For example, a *yearlet* storing daily average/low/high air temperatures for the entire year requires three parameters. In such a case, the yearlet should contain three strings one for average, one for low, and one for high air temperatures. Each string should follow the same principle used in daylets, i.e., fixed length for each datum. Each string should be separated by a pair of carriage return and line feed ASCII characters for distinction. Figure 3 illustrates a *yearlet* with three strings.

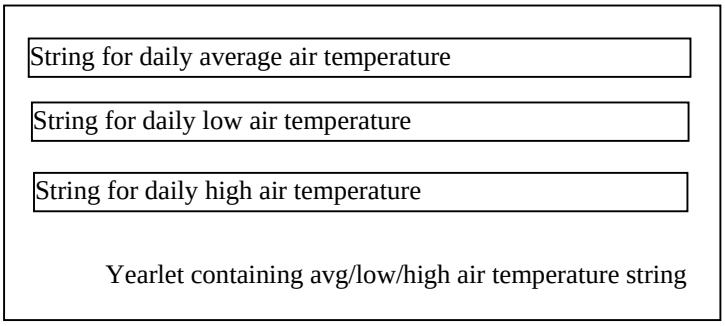

Figure 3: Yearlet example of daily avg/low/high temperature for the entire year.

In reference [3], we described a computational hierarchy in which processed data can be organized and archived as a hierarchical directory structure. In a very large system such as a statewide network of sensors, archiving processed data for sharing is considered highly beneficial. Examples include AADT or daily average of RWIS parameters. For developing a directory structure for processed data, we recommend to follow the structure of the examples given in Figure 2. More specifically, additional root directory can be created for the computational hierarchies for each class of data. In the above example, we could create one directory for processed RWIS data and another directory for processed traffic data. The children directories of the computational hierarchy would depend on the type of computation and outcome, so it would require a development of subdirectories for the specific needs.

9. Binary UTSDF

In a standard UTSDF, all of the sensor data in *daylets*, *monthlets*, and *yearlets* are stored using ASCII characters. However, if the data consists of only numerical values and fixed sizes, binaries could be used instead of ASCII characters. When the data in daylets are stored in binary form, we refer the archive as *binary UTSDF*. In general, we do not recommend binary UTSDF

since they are less portable between different operating systems and programming languages. Binary data can create a compatibility problem of byte orders known as Little-Endian and Big-Endian as well as the size definition in integers and floating points. Since the benefits of using binary for a smaller file size is diminished after zip-compression as demonstrated in [3], binary UTSDF is not recommended for archiving TSD.

10. Conclusion

We presented UTSDF in this introductory documentation. UTSDF was developed for archiving a very large set of transportation sensor data that include many different types of sensors. It is a simple and easy to use, and can be used in developing well-organized large archives. It is our hope that UTSDF is adopted in other states so that transportation sensor data can be easily archived and shared. At TDRL, we are continuously working on developing data visualization and analysis tools for UTSDF data we are providing. These software tools are presently distributed through http://www.d.umn.edu/~tkwon/TDRLSoftware/Download.html or links from http://tdrl1.d.umn.edu/services.htm.

References

1. U.S. DOT ITS, Archived Data User Service (ADUS), “ITS Data Archiving: Five-Year Program Description,” March 2000, Published by U.S. DOT, ADUS Program.

2. Alan V. Oppenheim, Alan S. Willsky, Ian T. Young, *Signals and Systems*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1983.

3. T.M. Kwon and N. Dhruv, “Unified Transportation Sensor Data Format (UTSDF) for Efficient Archiving and Sharing of Statewide Transportation Sensor Data,” *Proc. of the Transportation Research Board 83nd Annual Meeting*, Washington D.C., Jan. 2004.

4. Ziv J. and Lempel A., "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337-343.

5. Huffman, D. A., "A Method for the Construction of Minimum Redundancy Codes", Proceedings of the Institute of Radio Engineers, September 1952, Volume 40, Number 9, pp. 1098-1101.

Appendix A: RWIS Daylet Definitions

Table A.1 Parameter Definitions

Index	Parameter	File extension	Digits	Values
Atmospheric Parameters				
0	Air Temp	atemp	4	Tenths of degree Celsius
1	Dew Temp	dtemp	4	Tenths of degree Celsius
2	RH	relhum	2	Percent, 100=PP
3	Wind Speed Avg.	avgspd	4	Tenths of meters/sec
4	Wind Spd Gust	gstspd	4	Tenths of meters/sec
5	Wind Direction Avg.	avgdir	3	Clockwise degrees from North
6	Wind Direction Max.	maxdir	3	Clockwise degrees from North
7	Precip Intensity	pinten	1	See Table A.2*
8	Precip Type	ptype	1	See Table A.3*
9	Visibility	visib	5	Tenths of meter
10	Air Pressure	apress	5	Tenths of millibar
11	Precip Rate	prate	3	Tenths of Cm/hr.
12	Precip Accum	paccum	4	Tenths Cm over 24 hr starting at Midnight local time
13	10 min Solar	10msol	5	Tenths Joule/sq. meter
14	24 hr Solar	24hsol	6	Tenths Joule/sq. meter
15	24 hr Sun	24hsun	4	Minutes over 24hr
16	Air Temp Max	amaxtemp	4	Tenths of degree Celsius
17	Air Temp Min	amintemp	4	Tenths of degree Celsius
18	Wet Bulb Temp	Wbtemp	4	Tenths of degree Celsius
19	Last Precip start	Pstart	14	yyyymmddHHMMSS
20	Last Precip end	Pend	14	yyyymmddHHMMSS
21	1 hr Precip Accum	1hpaccum	4	Tenths Cm
22	3 hr Precip Accum	3hpaccum	4	Tenths Cm
23	6 hr Precip Accum	6hpaccum	4	Tenths Cm
24	12 hr Precip Accum	12hpaccum	4	Tenths Cm
25	24 hr Precip Accum	24hpaccum	4	Tenths Cm
Surface Parameters				
0	Surface condition	surcond	1	See Table A.4*
1	Surface Temp	surtemp	4	Tenths of degree Celsius
2	Freeze Temp	frztemp	4	Tenths of degree Celsius
3	Chemical Pct.	chmpct	2	Percent, 100=PP
4	Depth	dpth	3	Hundredth of millimeter
5	Ice Pct.	Icepct	2	Percent, 100=PP
6	Salinity	Salin	5	Parts/100,000
7	Conductivity	Conduc	4	Mhos
Sub-Surface Parameters				
0	Surface Sensor Id	surid	2	Integer
1	Subsurface temp	subtemp	4	Tenths of degree Celsius
2	Subsurface moisture	submoist	2	Percent
3	Delta-t	delta	5	Picoseconds

Table A.2 Precipitation Intensity

Classification	Code
None	0
Light	1
Slight	2
Moderate	3
Heavy	4
Other	5
Unknown	6
Anything else	7

Table A.3 Precipitation Type

Classification	Code
None	0
Yes	1
Rain	2
Snow	3
Mixed	4
Light	5
Light Freezing	6
Freezing Rain	7
Sleet	8
Hail	9
Frozen	A
Unidentified	B
Unknown	C
Other	D
Anything else	E

Table A.4 Surface Condition

Classification	Code
Dry	0
Wet	1
Chemically Wet	2
Snow/Ice Watch	3
Snow/Ice Warning	4
Damp	5
Frost	6
Wet Above Freezing	7
Wet Below Freezing	8
Absorption	9
Absorption at Dewpoint	A
Dew	B
Black Ice Warning	C
Other Slush	D

Daylet field definition:

SysID.SiteID.SensorID.ParaName

SysID: A unique number assigned for the system characteristics of the RWIS stations based on manufacturer or specially categorized group.
SiteID: A unique numeric number assigned to each site based on geographical location.
SensorID: A unique number assigned within multiple sensors of identical types in a site. For example, if three pavement temperature sensors are installed at a site, the sensors are assigned with SensorID, 0, 1, and 2.
ParaName: Shortened parameter name described in Table A.1

Appendix B: Traffic Daylet Definitions

Table B: Traffic daylet definitions and parameters

Index	Parameter	File Extension	Digits	Values
1	30sec volume	v30s	2	Veh counts/30 sec
2	30sec occupancy	o30s	3	Tenths of percent, 100.0=PPP
3	30sec speed	s30s	3	Tenths of Miles/hr
4	1min volume	v1m	2	Veh counts/1min
5	1min occupancy	o1m	3	Tenths of percent, 100.0=PPP
6	1min speed	s1m	2	Tenths of Miles/hr
7	5min volume	v5m	3	Veh counts/5min
8	5min occupancy	o5m	3	Thenths of percent, 100.0=PPP
9	5min speed	s5m	3	Tenths of Miles/hr
10	1hr volume	v1h	4	Veh counts/1hr
11	1hr occupancy	o1h	3	Thenths of percent, 100.0=PPP
12	1hr speed	s1h	3	Tenths of Miles/hr