Designation: E2560 – 23

Standard Specification for Data Format for Pavement Profile[1]

This standard is issued under the fixed designation E2560; the number immediately following the designation indicates the year of original adoption or, in the case of revision, the year of last revision. A number in parentheses indicates the year of last reapproval. A superscript epsilon (ε) indicates an editorial change since the last revision or reapproval.

1. Scope

1.1 This specification describes a data file format for pavement profile.

1.2 This specification describes the variables and sizes of all data that will be stored in the file. The file is in binary format and is fully documented in this specification.

1.3 This specification is designed to be independent of hardware platforms, computer languages, and operating systems (OS).

1.4 *This standard does not purport to address all of the safety concerns, if any, associated with its use. It is the responsibility of the user of this standard to establish appropriate safety, health, and environmental practices and determine the applicability of regulatory limitations prior to use.*

1.5 *This international standard was developed in accordance with internationally recognized principles on standardization established in the Decision on Principles for the Development of International Standards, Guides and Recommendations issued by the World Trade Organization Technical Barriers to Trade (TBT) Committee.*

2. Referenced Documents

2.1 *ASTM Standards:*[2]
E867 Terminology Relating to Vehicle-Pavement Systems

2.2 *IEEE Standards:*[3]
IEEE 754–2008 (2008) Floating-Point Arithmetic

3. Terminology

3.1 *Definitions:*

3.1.1 Terminology used in this specification conforms to the definitions included in Terminology E867.

3.2 *Definitions of Terms Specific to This Standard:*

3.2.1 *signed*—integer capable of representing negative values.

3.2.2 *unsigned*—integer only capable of representing non-negative values.

3.2.3 *int8*—data type for an 8-bit, unsigned integer.

3.2.4 *int32*—data type for a 32-bit, signed integer.

3.2.5 *single*—data type for a 32-bit, signed real number, such as single-precision IEEE floating point.

3.2.6 *string*—data type for a variable-length ASCII string. No null character is included at the end of the string. A separate field defines the length of the string.

3.2.7 *3-byte string*—an ASCII string of three characters in length. No null character is included at the end of the string.

3.2.8 *4-byte string*—an ASCII string of four characters in length. No null character is included at the end of the string.

3.2.9 *8-byte string*—an ASCII string of eight characters in length. No null character is included at the end of the string.

3.2.10 *array (numeric data type)*—sequence of data of the specified numeric data type. Only the values are stored; no information about the array is stored.

3.2.11 *array (string)*—ASCII strings separated by a tab. There is no tab after the last string.

3.2.12 *double*—data type for a 64-bit, signed real number, such as double-precision IEEE floating point.

3.3 *Symbols:*

3.3.1 *n*—total channels of elevation data.

3.3.2 *m*—total number of test locations (that is, data points).

4. Profile Data Specifications

4.1 *File Structure:*

4.1.1 The general file structure is divided into five sections: (*1*) File Header, (*2*) Metadata, (*3*) Longitudinal Profile Data, (*4*) Transverse Profile Data, and (*5*) File Trailer. The five sections are stored sequentially. (See Fig. 1.)

4.1.2 Each of these portions of the file is described in the following sections, as well as the data types and other descriptors that will be required by the file. The data will be written to the file sequentially, with the offsets listed in the file header as guides to find various portions of the file. It is

[1] This specification is under the jurisdiction of ASTM Committee E17 on Vehicle - Pavement Systems and is the direct responsibility of Subcommittee E17.31 on Methods for Measuring Profile and Roughness.

Current edition approved May 1, 2023. Published May 2023. Originally approved in 2007. Last previous edition approved in 2017 as E2560 – 17. DOI: 10.1520/E2560-23.

[2] For referenced ASTM standards, visit the ASTM website, www.astm.org, or contact ASTM Customer Service at service@astm.org. For *Annual Book of ASTM Standards* volume information, refer to the standard's Document Summary page on the ASTM website.

[3] Available from Institute of Electrical and Electronics Engineers, Inc. (IEEE), 445 Hoes Ln., P.O. Box 1331, Piscataway, NJ 08854-1331, http://www.ieee.org.

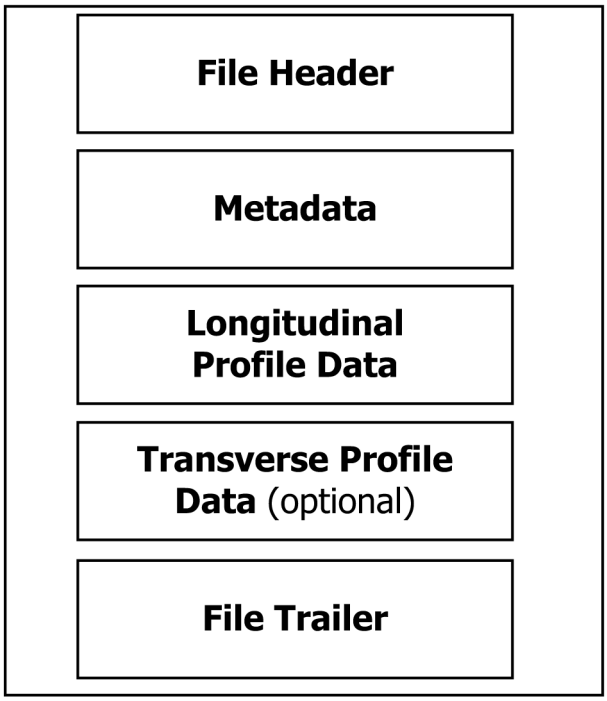

FIG. 1 Layout of the File Structure

important to note that all offsets are relative to the beginning of the file. Because offset values may not be known at the time of writing the file header, these values need not be written. However, spare space must still be reserved for the offsets so that values can be updated when known.

4.2 *File Header*—The file header contains information pertaining to the data file type, software version information, and information about the data contained (Table 1).

4.2.1 Version 1.06 introduces support for double-precision numbers. If a file does not use double-precision numbers, then a version of 1.05 should be used for better compatibility.

4.3 *Metadata:*

4.3.1 Metadata is structured, descriptive information about a resource or data about data. Using metadata in the binary file format will allow generic operation on the data information about which the reader software has no prior knowledge. Also, metadata will allow scalable evolution of the data description without requiring simultaneous upgrades to all reader software.

4.3.2 The first value in the metadata portion will provide the number of metadata entries (MDE) (Table 2).

TABLE 2 Metadata

Variable Name	Data Type	Data
Number of MDEs	Int32	Number of MDEs

Table 3 shows the information required to construct an appropriate MDE.

4.3.3 The metadata tags are listed in Table 4, and can be used in any number or order. If no metadata tags exist, the number of MDEs = 0.

4.3.4 The names of the standard metadata entries (see Table 3) are not stored in the metadata entry to conserve space and, more importantly, to allow for localization. That is, the file is not tied to one written language. User-defined metadata entries cannot be arrays, and the data type is always String.

4.3.5 The storage convention for empty arrays is to store a one-byte value of the same data type as the array. For example, an array of singles with no elements would store a value of 0.

4.4 *Longitudinal Profile Data:*

4.4.1 There are two ways to store the profile data: location-wise and array-wise. The first method is appropriate for data recording during profile data collection to prevent data loss, while the other is appropriate for post-processing to speed up software reading and writing.

4.4.2 If the data storage format, from metadata tag number 522 specifies location-wise storage, the longitudinal data will be stored as a sequence of current longitudinal distance followed by corresponding elevations of longitudinal sensors at this location, beginning at the left side of the vehicle. The next block of storage will store longitudinal distance and all elevation data for the next location, and so on. However, the location may not need to be stored if a specific data interval is given. (See Fig. 2.)

4.4.2.1 In general, if a location and elevation channels are recorded for each test location, every set of $n+1$ Singles (one distance data and n channels of elevation data) will be read as one profile location. If a specific data interval is included in the metadata, only n Singles will be read for each location. For example, if a standard interval exists and a single channel of

TABLE 1 File Header

Variable Name	Data Type	Data	Default Value
Signature	4-byte String	Identifies file as being written in the Standard Pavement Profile Format	"SPPF"
Version	4-byte String	Identifies the version number of the file format. This number is incremented if a change breaks compatibility with previous versions of the format.	"1.06"
SW version	8-byte String	Identifier of the software that produced the file	for example, "TGPA1.00"
Metadata offset	Int32	Offset in bytes from the beginning of the file to the beginning of the metadata	N/A
Longitudinal offset	Int32	Offset in bytes from the beginning of the file to the beginning of the longitudinal profile data	N/A
Transverse data offset	Int32	Offset in bytes from the beginning of the file to the beginning of the transverse profile data	N/A

TABLE 3 Metadata Entries

Variable name	Data type	Data
Tag of MDE	Int32	Metadata tag (see Table 4)
Data type of MDE	Int32	Data type index of MDE (see Table 5)
Array size	Int32	"-1" if not an array. "0" if the array is empty. Numbers greater than 0 specify the number of elements in the array. Even though arrays of strings are stored differently than other types of arrays, an array size should still be specified here.
Count	Int32	For data types "String" and "Array (String)," count = the number of bytes in the string. For other data types, count = 1.
Name length	Int32	For metadata entries listed in Table 4, this is 0. For user-defined entries, this value is the length of the name.
Name	String	Name of the metadata
MDE	varies	Information associated with the tag of MDE

profile data is present, only one Single will be read for each location. If two are present, then two Singles will be read per point.

4.4.2.2 The location-wise format is recommended for profiler data acquisition software. Storing the data after every sampling location allows immediate writing to protect against data loss and reduce memory requirements.

4.4.3 If the data storage format from metadata tag number 522 specifies array-wise storage, then the longitudinal data will be stored as a sequence of the longitudinal distance array followed by the elevation array of each longitudinal sensor, beginning at the left side of the vehicle for all locations. (See Fig. 3.)

4.4.3.1 In general, if distance and elevation channels are recorded for each test location, *n+1* sets (one distance channel and *n* channels of elevation data) of *m* Singles will be stored in sequence, where *m* is the number of points. If a specific data interval is included in the metadata, only *n* sets of *m* Singles will be stored sequentially, with distance being calculated from the beginning of the test location by the software. For example, if a standard interval exists and a single channel of profile data is present, only one set of Singles will be stored. If two are present, then two sets of *m* Singles will be stored sequentially.

4.4.3.2 This data format is recommended for software that reads and writes the data during post-processing. Data stored as one continuous array (array-wise) can be read and processed much faster than the location-wise storage format.

4.5 *Transverse Profile Data*—The transverse elevation readings are treated the same as the longitudinal data.

4.6 *File Trailer*—The file trailer is used to signal the end of the file. (See Table 15.)

4.7 *Event Markers*—Event markers are defined by tags 528 to 531. These four arrays must all be of the same length.

4.8 *Sections:*

4.8.1 A section is defined by the use of two event markers. The first event marker is the start location of the section and the second event marker is the stop location. Special attention should be paid to lead-in and lead-out event markers. These two markers define the section that is bounded by the lead-in and lead-out. Please note that they do not define the lead-in and lead-out but the section between them. An example of this follows:

4.8.1.1 A 1000-point profile has a lead-in of 50 points and a lead-out of 40 points. Points 0 to 49 will constitute the lead-in, so the event marker index for lead-in will be 50. Points 960 to 999 constitute the lead-out, so the event marker index for lead-out will be 959.

4.8.2 Tags 526 and 527 were defined before the use of event markers to define sections. These tags are no longer used.

4.8.3 Tag 531 is a recent addition to help ensure the integrity of the sections, even if the event markers are not in order. The tag is not required, but if present, it must be the same length as tags 529 and 530. As this tag is new, there is no guarantee that file readers will use it. When writing a file, ensure that events are not sorted but rather stored in the order created. Use tag 531 only to verify the order. An example follows, containing two sections and one event marker. The key for tags 311 and 531 are random ASCII strings that can be of any length. The only constraint is that the values cannot be duplicated for a given tag.

311 (Section keys)	57A9, GD89
312 (Section names)	Section 1, Section 2
528 (Event marker index)	300, 350, 699, 800, 1000
529 (Event marker text)	(blank), (blank), Event 1, (blank), (blank),
530 (Event marker type)	2, 3, 1, 2, 3
531 (Event marker section-related key)	65UW, 65UW, 7H89, 8GJK

4.9 *Geographical Data*—Geographical coordinates can be provided for the profile and events. Beginning with version 1.06 of this standard, coordinate data can be stored as low precision or high precision. It is recommended to store data with high precision once readers are readily available to process this data. For brevity, this section refers only to the tags that store low-precision data.

4.9.1 Tags 532, 533, and 534 are used to record the geographical location of events. These values are not required to be set, but the arrays must be the same size as the other event marker arrays.

4.9.2 There are two ways to define the original geographical coordinates associated with a profile. Tags 318 to 323 define the start and stop coordinates for a profile. Or, tags 535 and 536 can be used by the profiler to record the geographical location of multiple points in the profile. Tag 538 is also required to associate each coordinate with a distance on the profile. Tag 541 is optional. Tags 539, 540, and 542 provide storage for a processed route. With either of these methods, there may be too little or too many data to create a route suitable for use. Because converting these methods into a usable route can take time, these tags allow for storing the processed route.