
Standard Specification for File Format of Intelligent Construction Data

AASHTO Designation: MP 39-22 (2023)¹

Technically Revised: 2022

Reviewed but Not Updated: 2023

Editorially Revised: 2022

Technical Subcommittee: 5c, Quality Assurance and Environmental



**American Association of State Highway and Transportation Officials
555 12th Street NW, Suite 1000
Washington, D.C. 20004**

Standard Specification for

File Format of Intelligent Construction Data

AASHTO Designation: MP 39-22 (2023)¹

AASHTO

Technically Revised: 2022

Reviewed but Not Updated: 2023

Editorially Revised: 2022

Technical Subcommittee: 5c, Quality Assurance and Environmental

1. SCOPE

- 1.1. This specification describes a tagged file storage format used for georeferenced intelligent construction data. The type of data includes, but is not limited to, intelligent compaction data and paver-mounted thermal profiling data.
- 1.2. This specification is designed to be independent of hardware platforms, computer languages, and operating system (OS).
-

2. REFERENCED STANDARDS

- 2.1. *IEEE Standard:*
- < IEEE 754–2008, Floating-Point Arithmetic
- 2.2. *Websites:*
- < EPSG Geodetic Parameter Dataset. <https://epsg.org/home.html>
 - < ESRI Projected Coordinate Systems. <https://desktop.arcgis.com/en/arcmap/latest/map/projections/what-are-map-projections.htm>
 - < Example Time Resolution for Programming Languages and Applications on the Wikipedia. https://en.wikipedia.org/wiki/System_time#Programming_languages_and_applications
 - < Intelligent Construction. <https://www.intelligentconstruction.com/>
 - < International Society for Intelligent Construction. <https://www.is-ic.org/>
 - < PROJ Projection Parameters. <https://proj.org/usage/projections.html>
 - < The Unified Code for Units of Measure–UCUM. <https://ucum.org/trac>
-

3. TERMINOLOGY

- 3.1. *Definitions:*
- 3.1.1. *array (numeric data type)*—Sequence of data of the specified numeric data type.
- 3.1.2. *array (string)*—Each element is stored as UTF-8 text and is preceded by length stored as an Int16.
- 3.1.3. *ASCII*—Method of encoding basic English printable characters and symbols.
- 3.1.4. *byte*—Data type for an 8-bit, unsigned integer.
- 3.1.5. *double*—Data type for a 64-bit, signed real number, e.g., double precision IEEE floating point.
-

- 3.1.6. *Int16*—Data type for a 16-bit, signed integer.
- 3.1.7. *Int32*—Data type for a 32-bit, signed integer.
- 3.1.8. *Int64*—Data type for a 64-bit, signed integer.
- 3.1.9. *lookup table (LUT)*—Array of numeric items, where each item is associated with a number or string value. A lookup table is typically used to conserve space or reduce computation time.
- 3.1.10. *signed*—Capable of representing negative and non-negative values.
- 3.1.11. *single*—Data type for a 32-bit, signed real number, e.g., single precision IEEE floating point.
- 3.1.12. *string*—Data type for variable-length text. The string is not terminated and must be preceded by the number of bytes used to represent the string, stored as an *Int16*. All strings are encoded using ASCII, except for data values, which are encoded using UTF-8.
- 3.1.13. *Unicode*—A set of characters capable of representing almost any writable character.
- 3.1.14. *unsigned*—Capable of representing only non-negative values.
- 3.1.15. *UTF-8*—Method of encoding Unicode. The first 128 characters encoded the same as ASCII.
- 3.1.16. *Veta*—A standardized intelligent construction data management (ICDM) software that stores, maps, and analyzes intelligent compaction and associated geospatial data (e.g., thermal profile data, spot test data). This software can perform standardized data processing, analysis, and reporting to provide project summary results quickly in the field from various IC manufacturers. The software can provide statistics, histograms, correlations for the IC measurements (e.g., speed, temperature, pass count, Intelligent Compaction Measurement Value (ICMV)), and document coverage area; and it can evaluate the uniformity of compaction as part of the project quality control operations. Veta can be downloaded from the www.intelligentconstruction.com website to validate data files written based on this standard.

4. DATA SPECIFICATION

- 4.1. Data value are stored in binary format for maximum performance, minimum file size, and protection against casual modification. The metadata or data descriptions are stored in text format for ease of parsing and understanding.
- 4.2. *Byte Order:*
- 4.2.1. All values that require more than one byte must be stored in Little Endian format with the least significant byte (LSB) first and the most significant byte (MSB) last.
- 4.3. *File Name:*
- 4.3.1. The file extension must be “tds”, which stands for “tagged data storage.”
- 4.4. *File Structure:*
- 4.4.1. The file structure uses a series of blocks. Each block contains a description of the data and the data itself, when applicable.

- 4.4.2. Blocks are grouped into sections. Sections are noted by text in brackets. The following sections are available and when used, must be present in this order: [file-headers], [lookup-tables], [data-headers], [data-source], and [log-data].
- 4.4.3. The format is designed to store one set of coordinates for a given timestamp. The file is limited to one machine per file, but multiple sensors are allowed. A geographical offset can be defined for each sensor, allowing for different coordinates for each sensor.
- 4.4.4. The format is generally defined to store measured, not calculated, data.
- 4.4.5. Measurement data can be stored in the data source section or the log data section. While it is possible to store data in both sections, it is not recommended because the two options are for different scenarios. If all measurement data is available, data should be stored in the data source section for best performance. If data is being stored as it is measured, the log data section is more appropriate for data transmission. After initial values are saved for data, subsequent values are stored only if they have changed from the previous value. This allows for a compact file but requires more time for the reading software.
- 4.4.6. Because each block is self-describing, processing software can and must preserve all data blocks, even when they are not recognized.
- 4.5. *File Header:*
- 4.5.1. The file header shall contain two data blocks that describe the file format: (1) signature, and (2) version.
- 4.5.1.1. *Signature*—The signature block shall contain the string data, “TaggedDataFormat”.
- 4.5.1.2. *Version*—The version for this specification is “1”. The version is only incremented if a file format change breaks compatibility with previous versions.
- 4.6. *Data:*
- 4.6.1. Two types of blocks are available: (1) data block, and (2) value block.
- 4.6.2. *Data Block:*
- 4.6.2.1. A data block describes and optionally includes the data. The block is defined in Table 1.

Table 1—Definition of Data Block

Variable	Data Type	Description
Name	String	Name of the data
Block ID	Int16	ID unique to this file (optional)
Measurement Unit	String	Unit of measure for the value
Data Type	String	Data type name (see Table 2)
Byte Length	Int32	Number of binary bytes used to store the value
Data	Varies	The data bytes appropriate to the block type (optional)

- 4.6.2.2. *Name:*
- 4.6.2.2.1. Block names must be encoded in ASCII. Names are not case-sensitive, but this specification lists names in lower-case. A list of recognized data is provided in Table 6. Names can contain numbers but cannot consist solely of numbers. Distinct words in the name are separated by a hyphen, “-”.

Multiple sensors of the same type should end with a pound or hash symbol ('#'), followed by a number. Examples: roller-speed, surface-temperature#1, surface-temperature#2

- 4.6.2.2.2. Custom data blocks (blocks not listed in this specification) are allowed. Custom blocks names must have a prefix in the format of:
-organization-. Example from company ABC: -abc-unique-measurement
- 4.6.2.3. *Block ID:*
- 4.6.2.3.1. The ID is a numeric value greater than zero that is unique to this file. When another block needs to refer to this block, it will use this ID rather than the name to conserve space. This value can be 0 if no other block refers to this block.
- 4.6.2.4. *Measurement Unit:*
- 4.6.2.4.1. The measurement unit is specified by ASCII, case-insensitive names in the Unified Code for Units of Measure. A value of “none” is used when a unit is not applicable. This provides a standard method for transmitting unit information in a file. When displaying the unit, more common text may be used.
- 4.6.2.5. *Data Type:*
- 4.6.2.5.1. The data types recognized by the standard file format are listed in Table 2. Custom data types are allowed, but arrays of custom data types are not allowed.
- 4.6.2.5.2. Integers are stored as an Int32 unless file size would be significantly affected.

Table 2—Recognized Data Types

Data Types	Byte Length
Byte	1
Byte-array	1 * length
Int16	2
Int16-array	2 * length
Int32	4
Int32-array	4 * length
Int64	8
Int64-array	8 * length
Single	4
Single-array	4 * length
Double	8
Double-array	8 * length
String	Varies
String-array	Varies
Timestamp	12
Timestamp-array	12 * length
Boolean	1
Boolean-array	1 * length

- 4.6.2.5.3. To conserve space, this specification defines a Boolean as a Byte. A value of 0 is False and a value of 1 is True.
- 4.6.2.5.4. A timestamp requires a time and UTC offset to ensure historical accuracy. It is insufficient to store one time zone for the file because time zone definitions change over time and the offset can change for daylight savings. When converted to binary storage for this specification, the value is stored as an Int64 for the time and an Int32 for the offset. The Int64 signifies the number of ticks

that have elapsed since a specified epoch, a common method of measuring time on computers. The number of seconds represented by a tick varies among programming languages. This specification uses the definition from Microsoft: The tick count is the number of 100 nanosecond intervals that have elapsed since the epoch date of 00:00:00.000, January 1, 0001 in the Gregorian calendar. The Int32 specifies the UTC offset in minutes, including daylight savings time.

4.6.2.6. *Byte Length:*

4.6.2.6.1. Number of binary bytes used to store the value. This value can be 0 to specify that the block does not contain data. The length for each data type is provided in Table 2.

4.6.2.7. *Data:*

4.6.2.7.1. Data for block. If the length is 0, this item will be empty. Some data use values obtained from lookup tables. A list of recognized lookup tables is provided in Table 7.

4.6.3. *Data Quality:*

4.6.3.1. For Single and Double data, values can be recorded as Not a Number (NaN) as specified in IEEE 754. This signifies the data is not valid. This is most likely to be found in arrays of data. GPS quality is specified using the GPS-quality block. The quality of other data sources is defined using the quality and threshold blocks.

4.6.4. *Value Block:*

4.6.4.1. A value block is used only by the Log Data section. It consists of an ID and one data value. The ID associates the data value with the previously defined data block. The value block is defined in Table 3.

4.6.5. *Gridding:*

4.6.5.1. Software using data from these files will often need to apply a gridding process for display and analysis purposes. For example, a data with a footprint of 3 ft by 1 ft will use at least 3 squares of a grid that is composed of 1-ft by 1-ft squares. Gridding usually entails processing the data on a computer, whereas ungridded data comes directly from the recording machine.

4.6.5.2. For the purposes of this specification, if gridding is used, offset and footprint blocks for the data sources should not be used. If data has been gridded, the offset and footprint of the data have already been used to determine which grid squares are used by the data. If the data is gridded, the coordinates must be the center of the grid.

Table 3—Definition of Value Block

Variable	Data Type	Description
Block ID	Int16	Unique ID defined in the associated data block
Data	Varies	The data bytes appropriate to the block type

4.7. *Recognized Data Blocks for Sections:*

4.7.1. *File Headers:*

Table 4—Required Blocks for the File Headers Section

Name	Data Type	Unit Category	Description
file-signature	String	None	Value must be “TaggedDataFormat”
file-version	Int32	None	Value must be 1

4.7.2. *Data Headers:*

Table 5—Recognized Blocks for the Data Headers Section

Name	Data Type	Unit Category	Description
data-lot-name	String	None	Name for the area covered by this file; naming convention specified by PP 80 and PP 81
gridding-height	Single	Length	Length in north–south direction
gridding-process	String	None	Ungridded, gridded
gridding-width	Single	Length	Length in east–west direction
lift-design-thickness	Single	Length	Thickness of lift
machine-id	String	None	Unique identifier for a machine
machine-name	String	None	Common or display name for a machine
machine-type	String	None	Recognized values: compactor paving thermal profiler inertial profiler rolling density meter laser test roller
spatial-reference-authority	String	None	Examples: ESRI, EPSG, PROJ4
spatial-reference-id	Int32	None	Well-known ID used by the authority to identify the spatial reference
spatial-reference-parameters	String	None	If an ID is not used, parameters that define the spatial reference can be provided using a format defined by PROJ4 or another authority

4.7.3. *Data Sources:*

4.7.3.1. The recognized blocks for a data source are listed in Table 6. Figure 1 illustrates the items in Table 6.