

Comma-Separated Value (CSV) Reader/Writer

The Comma-Separated Value (CSV) Reader and Writer modules provide the Feature Manipulation Engine (FME) with direct access to data in CSV format. It presents the user with a somewhat simpler interface to the FME's Relational Table Reader/Writer CSV module.

Overview

CSV files are ASCII database files, where each column in a row is separated by some separator character. The FME feature attributes for each line of the file are the columns values of that row. There is no geometry or dimension to the features created from the CSV files, but they may have attributes that can be turned into geometry via FME facilities such as @XValue, @YValue, and ConnectionFactory. Therefore, none of the features read from CSV are directly viewable.

By convention, these files use the .csv filename extension, but the CSV reader and writer can use any extension. CSV reader can also read from a gzipped file with that extension ".csv.gz" and the writer can write a gzipped file if the extension of destination file ends with ".gz"

Tip: When using RELATE statements that use a CSV file (where the CSV file is not part of the source or destination dataset), the Relational Table reader/writer keywords for CSV should be used instead of the CSV reader/writer keywords (see the chapter on Relational Table Reader/Writer). For example:

```
Relate TABLE_DEF roads CSV          \  
  CSV_OUTPUT_FIELDNAMES yes         \  
  MSLINK      number(10,0)           \  
  LENGTH     number(10,2)           \  
  PAVEMENT   char(20)                \  
_____  
_____
```

CSV Quick Facts

Format Type Identifier	CSV
Reader/Writer	Both
Licensing Level	Base
Dependencies	None
Dataset Type	Directory or File
Feature Type	File base name
Typical File Extensions	.csv, .csv.gz
Automated Translation Support	Yes
User-Defined Attributes	Yes
Coordinate System Support	No
Generic Color Support	No
Spatial Index	Never
Schema Required	Yes
Transaction Support	No
Geometry Type	csv_type
Encoding Support	Yes

Geometry Support			
Geometry	Supported?	Geometry	Supported?
aggregate	no	point	no
circles	no	polygon	no
circular arc	no	raster	no
donut polygon	no	solid	no
elliptical arc	no	surface	no
ellipses	no	text	no
line	no	z values	n/a
none	yes		

Reader Overview

The CSV reader module produces an FME feature for each line in each the CSV files residing in the given directory. The CSV reader first scans the directory for all CSV files that are defined in the mapping file. If IDs lines are specified, the CSV reader processes only the specified files; otherwise, it reads all files in the directory. Optionally a single CSV file can be given in the mapping file. In this case, only that CSV file is read.

Reader Directives

The suffixes shown are prefixed by the current <ReaderKeyword> in a mapping file. By default, the <ReaderKeyword> for the CSV reader is CSV.

DATASET**Required/Optional:** *Required*

This is the name of a directory containing one or more CSV files, or the name of a single CSV file. The default extension for CSV files is .csv.

Example:

```
CSV_DATASET /usr/data/csv/input
```

Workbench Parameter: [<WorkbenchParameter>](#)

DEF**Required/Optional:** *Required*

Each CSV file must be defined before it can be read. The definition contains the file's base name without any of the extensions, followed by the names and types of the attributes. There may be many DEF lines, one for each file to be read. If this is not specified, then all defined CSV files in the directory are read.

The syntax of a CSV DEF line is:

```
<ReaderKeyword>_DEF <baseName> \
  [<attrName> <attrType>]+
```

The following table shows the attribute types supported.

Field Type	Description
char (<width>)	Character fields store fixed length strings. The <code>width</code> parameter controls the maximum number of characters that can be stored by the field. No padding is required for strings shorter than this width.
date	Date fields store date as character string with the format YYYYMMDD.
number (<width>, <decimals>)	Number fields store single and double precision floating point values. The <code>width</code> parameter is the total number of characters allocated to the field, including the decimal point. The <code>decimals</code> parameter controls the precision of the data and is the number of digits to the right of the decimal.
float	Float fields store floating point values. There is no ability to specify the precision and width of the field.
integer	Integer fields store 32-bit signed integers.
smallint	Small integer fields store 16-bit signed integers and therefore have a range of -32767 to +32767.
logical	Logical fields store TRUE/FALSE data. Data read or written from and to such fields must always have a value of either <code>true</code> or <code>false</code> .

The following table shows all the DEF line directives that are supported by the CSV reader. Each of these directives has the same meaning as the global CSV reader keyword with the same suffix. Any value specified on a DEF line will override values defined for equivalent global directives, as they apply to the table being defined.

DEF Line Directives	Value	Required/Optional
CSV_FIELD_NAMES <yes no>	See FIELD_NAMES for details.	Optional
CSV_FIELD_NAMES_AFTER_HEADER <yes no>	See FIELD_NAMES_AFTER_HEADER for details.	Optional
CSV_SEPARATOR (<separator>)	See SEPARATOR for details.	Optional
CSV_SKIP_LINES <number>	See SKIP_LINES for details.	Optional
CSV_STRIP_QUOTES <yes no>	See STRIP_QUOTES for details.	Optional
CSV_DUPLICATE_DELIMS <yes no>	See DUPLICATE_DELIMS for details.	Optional
CSV_EXTENSION <extension>	See EXTENSION for details.	Optional
CSV_ENCODING <encoding>	See ENCODING for details.	Optional

The following mapping file fragment defines a CSV file called `roads`. Here we define the '?' as the separator character for columns in the file and we choose not to output the field names to the output file.

```

CSV_DEF roads                                     \
  CSV_SEPARATOR          (?)                       \
  CSV_FIELD_NAMES       no                        \
  id_num                number(11,0)             \
  type                  char(20)                  \

```

IDs

Required/Optional: *Optional*

This specification limits the available and defined CSV files read. If no IDs are specified, then all defined and available CSV files in the directory are read.

The syntax of the IDs keyword is:

```

<ReaderKeyword>_IDs<baseName>                   \
  <baseName1> ...                                 \
  <baseNameN>                                     \

```

The basenames must match those used in DEF lines.

Comma-Separated Value (CSV) Reader/Writer

The example below selects only the `roads` CSV file for input during a translation:

```
CSV_IDS roads
```

Workbench Parameter: [<WorkbenchParameter>](#)

FIELD_NAMES

Required/Optional: *Optional*

If the field or column names of the CSV table are specified in the file, then set this value to `yes` and the names will be extracted from the file. Otherwise, the columns of the CSV table are given default names (i.e. `col0`, `col1`, ... , `colN`) with the setting `no`. The default is `no`.

Note: If `FIELD_NAMES` is set to `yes`, `skip_lines` should also be set to skip at least one row, or the first row will be also be processed as a feature. You can also set `FIELD_NAMES_AFTER_HEADER` to `yes`. See `FIELD_NAMES_AFTER_HEADER` below for details.

Values: `<yes | no>`

Workbench Parameter: [<WorkbenchParameter>](#)

FIELD_NAMES_AFTER_HEADER

Required/Optional: *Optional*

If the column/field names is `AFTER` the header information instead of `BEFORE`, then you can set `FIELD_NAMES_AFTER_HEADER` to `yes`. Otherwise, by default, the first line of the file will be used as the column/field names.

Notes:

- This parameter is ignored if `FIELD_NAMES` is not set, or it is set to `no`.
 - If `FIELD_NAMES_AFTER_HEADER` is set to `yes`, `SKIP_LINES` should also be set to skip at least one row, or the first row will be also be processed as a feature.
-

Values: `<yes | no>`

Workbench Parameter: [<WorkbenchParameter>](#)

SEPARATOR

Required/Optional: *Optional*

A special field is listed to identify the separator used to divide the fields in the file. By default, a comma is used; however, different one-character separators can also be specified. Tab character separators are indicated by a backslash (`\`) followed by a "t"; for example:

```
CSV_SEPARATOR (\t)
```

Note: There must be a space between `CSV_SEPARATOR` and `(<separator>)`. The begin and end parentheses are optional.

Values: `(<separator>)`

Workbench Parameter: [<WorkbenchParameter>](#)

SKIP_LINES

Required/Optional: *Optional*

This field can be listed to indicate the number of lines to skip at the top of the file. By default, no lines are skipped. Each line skipped is logged to the log file. This is useful if the CSV file contains a header line of field names or other descriptive material that should be skipped.

Values: *<number>*

Workbench Parameter: [*<WorkbenchParameter>*](#)

SKIP_FOOTER

Required/Optional: *Optional*

This field can be listed to indicate the number of footer lines to skip at the bottom of the file. By default, no footer lines are skipped. Each footer line skipped is logged to the log file. This is useful if the CSV file contains a footer line of descriptive material that should be skipped.

Values: *<number>*

Workbench Parameter: [*<WorkbenchParameter>*](#)

STRIP_QUOTES

Required/Optional: *Optional*

Some CSV files place quotation marks around all values they contain. By setting this special field to *yes*, then these quotes can be stripped from each attribute. The default is *no*.

Values: *<yes|no>*

Workbench Parameter: [*<WorkbenchParameter>*](#)

DUPLICATE_DELIMS

Required/Optional: *Optional*

This field can be listed to indicate if duplicate delimiters are to be treated as a single delimiter. If set to *yes* then multiple contiguous delimiters are treated as a single delimiter; otherwise, each delimiter is treated as if it delimits a different field.

Values: *<yes|no>*

Workbench Parameter: [*<WorkbenchParameter>*](#)

EXTENSION

Required/Optional: *Optional*

This specifies the file extension to be read or written in. The default is *.csv*.

Values: *<.extension>* (Include the period (.) in front of the extension name.)

Default: *0*

Workbench Parameter: [<WorkbenchParameter>](#)

ENCODING

Required/Optional: *Optional*

This specifies the file encoding to use when reading.

Values: *<encoding>*

Encodings
UTF-8
UTF-16LE
UTF-16BE
ANSI
BIG5
SJIS
CP437
CP708
CP720
CP737
CP775
CP850
CP852
CP855
CP857
CP860
CP861
CP862
CP863
CP864
CP865
CP866
CP869
CP932
CP936

Encodings
CP950
CP1250
CP1251
CP1252
CP1253
CP1254
CP1255
CP1256
CP1257
CP1258
ISO8859-1
ISO8859-2
ISO8859-3
ISO8859-4
ISO8859-5
ISO8859-6
ISO8859-7
ISO8859-8
ISO8859-9
ISO8859-13
ISO8859-15

Workbench Parameter: Character Encoding

Writer Overview

The CSV Writer writes all attributes of a feature to an CSV file. Features of different types are written to different CSV files.

Writer Directives

The suffixes shown are prefixed by the current <WriterKeyword> in a mapping file. By default, the <WriterKeyword> for the CSV reader is CSV.

DATASET

Required/Optional: *Required*

Comma-Separated Value (CSV) Reader/Writer

This is the name of a directory containing one or more CSV files. The default extension for CSV files is `.csv`. To write gzipped files, use `.csv.gz` as the destination file extension.

An example of the `DATASET` keyword in use is:

```
CSV_DATASET /usr/data/csv/output
```

Workbench Parameter: [<WorkbenchParameter>](#)

DEF

Required/Optional: *Required*

Defines a CSV file. The definition contains the file's base name without any of the extensions, followed by the definitions of the attributes. There may be many `DEF` lines, one for each file to be written.

The syntax of a CSV `DEF` line is:

```
<WriterKeyword>_DEF <baseName> \
[<attrName> <attrType>]+
```

The attribute types supported by the CSV writer are the same as those listed in the Reader Keywords `DEF` section. The following `DEF` line directives are supported by the CSV writer:

DEF Line Directives	Value	Required/Optional
CSV_FIELD_NAMES <yes no>	See FIELD_NAMES below for details.	Optional
CSV_SEPARATOR (<separator>)	See SEPARATOR below for details.	Optional
CSV_EXTENSION <extension>	See EXTENSION below for details.	Optional
CSV_ENCODING <encoding>	See ENCODING for details.	Optional
CSV_END_OF_LINE <encoding>	See END_OF_LINE for details.	Optional

Each of these directives has the same meaning as the global CSV writer keyword with the same suffix. Any value specified on a `DEF` line will override values defined for equivalent global directives, as they apply to the table being defined.

Workbench Parameter: [<WorkbenchParameter>](#)

FIELD_NAMES

Required/Optional: *Optional*

If the field or column names of the CSV table are specified as the first row of the file, set this value to `yes` and the names will be written to the file. Otherwise, none of the column names will be written to file.

Values: <yes|no>

Default: no

Workbench Parameter: <WorkbenchParameter>

SEPARATOR

Required/Optional: *Optional*

A special field is listed to identify the separator used to divide the fields in the file. By default, a comma is used; however, different one-character separators can also be specified. Tab character separators are indicated by a backslash (\) followed by a "t"; for example:

```
CSV_SEPARATOR (\t)
```

Note: There must be a space between CSV_SEPARATOR and (<separator>). The begin and end parentheses are optional.

Values: (<separator>)

Workbench Parameter: <WorkbenchParameter>

EXTENSION

Required/Optional: *Optional*

This specifies the file extension to be written. The default is .csv.

Note: Include the period in front of the extension name. .csv.gz extension will output gzipped files.

Values: <extension>

Workbench Parameter: <WorkbenchParameter>

QUOTE_OUTPUT

Required/Optional: *Optional*

This specifies whether the fields written to the CSV file are quoted. If set to *yes*, then every field, including field names, will be quoted. If set to *no*, no fields will be quoted. If set to *if_needed*, fields will be quoted only if they contain a delimiter character.

Note: The meaning of a *yes* value differs slightly between the CSV format writer and the CSV mode of the Relational Table writer.

Values: *yes | no | if_needed*

Workbench Parameter: <WorkbenchParameter>

QUOTE_FIELD_NAMES

Required/Optional: *Optional*

This specifies whether the field names written on the first row of the CSV file are quoted. If set to *yes*, then field names will be quoted. If set to *no*, field names will not be quoted.

Values: *yes* | *no*

Default: *no*

Workbench Parameter: [<WorkbenchParameter>](#)

Workbench Parameter: [<WorkbenchParameter>](#) **Workbench Parameter:** [<WorkbenchParameter>](#)

APPEND

Required/Optional: *Optional*

This specifies whether rows will be appended to existing files if a matching CSV file was found in the destination directory.

Values: *yes* | *no*

Default: *no*

ENCODING

Required/Optional: *Optional*

This specifies the file encoding to use when writing.

Values: *<encoding>*

Encodings
UTF-8
UTF-16LE
UTF-16BE
ANSI
BIG5
SJIS
CP437
CP708
CP720
CP737
CP775
CP850
CP852
CP855
CP857

Encodings

CP860

CP861

CP862

CP863

CP864

CP865

CP866

CP869

CP932

CP936

CP950

CP1250

CP1251

CP1252

CP1253

CP1254

CP1255

CP1256

CP1257

CP1258

ISO8859-1

ISO8859-2

ISO8859-3

ISO8859-4

ISO8859-5

ISO8859-6

ISO8859-7

ISO8859-8

ISO8859-9

ISO8859-13

Encodings

ISO8859-15

[Workbench Parameter: Character Encoding](#)

END_OF_LINE

Required/Optional: *Optional*

This specifies the end of line character to use when writing.

Values: *Macintosh | Windows | Unix | System*

Default: *System*

[Workbench Parameter: Line Termination](#)

Feature Representation

In addition to the generic FME feature attributes that FME Workbench adds to all features (see *About Feature Attributes* on page 7), this format adds the format-specific attributes described in this section.

The CSV feature attributes consists of the columns that were in the CSV table. All CSV features contain a `csv_type` attribute, which is always set to `csv_none` as there is no geometry to CSV features. This represents that the feature was generated from a CSV file.

Attribute Name	Contents
<code>csv_type</code>	The CSV geometric type of this entity. Range: <code>csv_none</code> Default: <code>csv_none</code>

