

dBase (DBF) Reader/Writer

The dBase Format (DBF) Reader and Writer modules provide the Feature Manipulation Engine (FME) with access to data in the DBF format.

All DBF files are formatted according to the dBase III specification. The DBF Reader/Writer reuses the Relational Table Reader/Writer's CSV to expand its capabilities from being usable only in the FME Universal Translator to also being usable in FME Workbench and Universal Viewer.

Note: Any single DBF file can have a maximum file size of 2 GB, a limit imposed by the dBase III specification. Files larger than 2 GB may be readable, but not officially supported. Files larger than 2 GB are not writable, and will produce an error message.

Overview

A DBF file defines a single table within a database. The feature attribution of a FME feature are the columns and values of the DBF database table. There is no geometry or dimension to the features created from the DBF files. They are all undefined. Therefore, none of the features created are viewable because there is no graphical component to the features.

DBF files store only feature attribution. The DBF format has one physical file. The extension is added to the basename of the DBF file.

DBF Quick Facts

Format Type Identifier	DBF
Reader/Writer	Both
Licensing Level	Base
Dependencies	None
Dataset Type	Directory or File
Feature Type	File base name
Typical File Extensions	.dbf
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	dbf_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 DBF reader module produces FME feature for each table entry held in the DBF files residing in the given directory. The DBF reader first scans the directory for all DBF files which are defined in the mapping file. It processes only the specified files if IDs lines are available. Otherwise, all files in the directory are read. The DBF reader extracts data from the file one row at a time, producing FME features before passing them on to the rest of the FME for further processing. When the file is exhausted, the DBF reader moves on to the next file in the directory. Optionally a single DBF file can be given as the dataset. In this case, only the single file is read. The reader supports dBASE III, dBASE IV and FoxPro files.

Reader Directives

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

DATASET

Required/Optional: *Required*

This is the name of a directory containing one or more DBF files, or a single DBF file. The extension for DBF files is `.dbf`.

Example:

```
DBF_DATASET /usr/data/dbf/input
```

Workbench Parameter: [<WorkbenchParameter>](#)

DEF

Required/Optional: *Required*

Each DBF file may optionally be defined before it can be read. The definition specifies the base name of the file, and the names and the types of all attributes.

Example:

```
<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 width 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 strings with the format YYYYMMDD.
number(<width>, <decimals>)	Number fields store single and double precision floating point values. The width parameter is the total number of characters allocated to the field, including the decimal point. The decimals parameter controls the precision of the data and is the number of digits to the right of the decimal.
logical	Logical fields store TRUE/FALSE data. Data read or written from and to such fields must always have a value of either true or false.
memo	The reader can read dBASE III, IV and FoxPro memo fields. When writing, only dBASE III format memo fields are supported.

The example below is a DEF line for the trees DBF file that has the attributes name and id_number:

```
DBF_DEF trees \
  name char(30) \
  id_number number(11,0)
```

Workbench Parameter: <WorkbenchParameter>

IDs

Required/Optional: *Optional*

This optional specification limits the available and defined DBF files read. If no IDs are specified, then all defined and available DBF files are read.

The syntax of the IDs keyword is:

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

The basenames must match those used on the DEF lines.

The example below selects only the pipeline DBF file for input during a translation:

```
DBF_IDS pipeline
```

Workbench Parameter: *<WorkbenchParameter>*ENCODING******Required/Optional:** *Optional*

This optional specification controls which character encoding is used to interpret text attributes from the DBF file. If the value is not set, then the character encoding will be automatically detected from the source DBF file. If the value is set, it will take precedence over the automatically detected character encoding.

This directive is useful when the character encoding information stored in the DBF file is missing or incorrect.

Example:

```
<ReaderKeyword>_ENCODING <character encoding>
```

Workbench Parameter: *<WorkbenchParameter>*

Parameter	Description
<character encoding>	The character encoding to use when interpreting text attributes. Must be set to any of the following values: ANSI - this means use the "current OS language" BIG5 EUC ISO OEM SJIS UTF-8 CP437 CP708 CP720 CP737 CP775 CP850 CP852 CP855 CP857 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 ISO8859-15

TRIM_PRECEDING_SPACES

Required/Optional: *Optional*

This option specifies whether the reader should trim preceding spaces of attribute values. If the option is set to `YES`, then preceding spaces in attribute values will be discarded. If the option is set to `NO`, then preceding spaces will be left intact. The default value is `YES`.

[Workbench Parameter: Trim Preceding Spaces](#)

Writer Overview

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

Writer Directives

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

The DBF writer processes the `DATASET` and `DEF` keywords as described in the *Reader Keywords* section above. However, it does not make use of the `IDS` keywords.

Unlike the reader, the writer requires a `DEF` line for each file being written.

The `ENCODING` directive is used to specify which character encoding should be used when writing text attributes into DBF files. If the value of this directive is not set, the current OS language is used. The syntax of the `ENCODING` writer directive is the same as the `ENCODING` reader directive, as described in the *Reader Directives* section.

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 DBF feature attributes consists of the column name that were in the DBF table. All DBF features contain a `dbf_type` attribute, which is always set to `dbf_none` as there is no geometry to DBF features. This shows that the feature was generated from a DBF file.

Attribute Name	Contents
<code>dbf_type</code>	The DBF geometric type of this entity. Range: <code>dbf_none</code> Default: <code>dbf_none</code>