

ESRI Shape Reader/Writer

The ESRI® Shape Reader and Writer module allows Feature Manipulation Engine (FME) to read and writer ESRI's Shape format. The Shape format is the native format of ESRI's ArcView product and has been made public by ESRI.

Overview

An ESRI shapefile consists of a main file, an index file, and a dBASE table. The main file is a direct access, variable-record-length file in which each record describes a shape with a list of its vertices. In the index file, each record contains the offset of the corresponding main file record from the beginning of the main file. The dBASE table contains feature attributes with one record per feature. The one-to-one relationship between geometry and attributes is based on record number. Attribute records in the dBASE file must be in the same order as records in the main file.

Shapefiles store both geometry and attributes for features. No topological information however is carried in a shapefile. A single logical shapefile consists of three physical files, each with one of the following file name extensions:

File Name Extension	Contents
.shp	Geometric data
.shx	Index to the geometric data
.dbf	Attributes for the geometric data
.sbn and .sbx	Spatial index for the geometric data. These two files will not exist unless you generate them with an ESRI product.

These extensions are added to the base name of the shapefile, creating separate physical files that must all reside in the same directory.

Point, multipoint, polyline, polygon, and multipatch geometric data can be stored in .shp files. However, a single .shp file can contain only one type of geometry. Each entity in a .shp file has a corresponding entry in the .shx index file and a corresponding row of attributes in the associated .dbf file. The order of the entries in each of these files is synchronized. For example, the third geometric entity in the .shp file is pointed to by the third entry in the .shx index file and has the attributes held in the third row of the .dbf file.

In the case of multipoint data, there is only one .dbf row for each set of points held in the file. This is in contrast with a point file where there is one .dbf row for each point. Polyline files contain linear features or aggregates of linear features, each having a sin-

gle attribute entry. Polygon files contain polygons or groups of disjoint or overlapped, in the case of holes, polygons each having a single attribute entity.

Tips:

- Aggregate linear features and aggregate polygonal features may be created using the *AggregateFactory*. They may be broken into their component pieces for output to formats that do not support aggregation using the *DeaggregateFactory*.
- If a polygon containing holes is written to a Shapefile, any adjacent holes will be merged into a single hole before the polygon is output.

The number and type of attributes associated with each entity is user-definable however, there must be at least one field in the .dbf file. As well, all features in the same shapefile will have the same number and type of attributes.

Note: Any single DBF (attribute) 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.

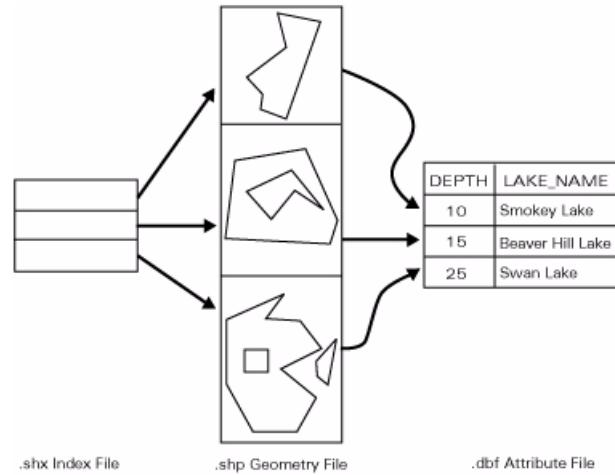
Shapefiles may hold both two- and three-dimensional geometry, as well as an optional measure value on each vertex. However, all features within a single shapefile will have the same dimensionality. Note that while older ESRI products may only support two-dimensional shapefiles, FME can read and write both two- and three-dimensional shapefiles. FME can also handle measure data associated with features.

Note: Measures are currently not supported when reading or the `shape_multipatch` geometry type.

In previous FME releases, when a feature with measures went through certain transformers, the measures would sometimes become out of sync with the feature that they were attached to. The addition of enhanced geometry support ensures that this will no longer happen, as long as the workspace/mapping file has been set to use enhanced geometry.

FME considers a Shape dataset to be a collection of shapefiles in a single directory. The geometry type and attribute definitions of each shapefile must be defined in the mapping file before being read or written.

The following diagram shows a Shape polygon file with three geometric entities in it. The index file has three entries, each of which refer to the vector data defining each polygon. Notice the second polygon contains a hole and the third polygon is an aggregate of two disjoint polygons, one of which contains a hole. Each geometric entity in turn corresponds with one record in the attribute table.



ESRI Shape Quick Facts

Format Type Identifier	SHAPE
Reader/Writer	Both
Licensing Level	Base
Dependencies	None
Dataset Type	Directory or File
Feature Type	File base name
Typical File Extensions	.shp (.shx, .dbf)
Automated Translation Support	Yes
User-Defined Attributes	Yes
Coordinate System Support	Optional
Generic Color Support	No
Spatial Index	Optional
Schema Required	Yes
Transaction Support	No
Enhanced Geometry	Yes
Encoding Support	Yes
Geometry Type	SHAPE_GEOMETRY

Geometry Support			
Geometry	Supported?	Geometry	Supported?
aggregate	yes	point	yes
circles	no	polygon	yes
circular arc	no	raster	no
donut polygon	yes	solid	no

Geometry Support			
Geometry	Supported?	Geometry	Supported?
elliptical arc	no	surface	yes
ellipses	no	text	no
line	yes	z values	yes
none	yes		

Reader Overview

The Shape reader produces FME features for all feature data held in shapefiles residing in a given directory. The Shape reader first scans the directory given for the shapefiles which have been defined in the mapping file. For each shapefile it finds, it checks to see if that file is requested by looking at the list of IDs specified in the mapping file. If a match is made or no IDs were specified in the mapping file, the shapefile is opened to be read. The Shape reader extracts features one at a time from the file and passes them on to the rest of the FME for further processing. When the file is exhausted, the Shape reader starts on the next file in the directory.

Reader Directives

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

DATASET

Required/Optional: *Required*

The value for this directive is the directory containing the shapefiles to be read, or a single shapefile. A typical mapping file fragment specifying an input Shape dataset looks like:

```
SHAPE_DATASET /usr/data/shape/92i080
```

Workbench Parameter: [<WorkbenchParameter>](#)

DEF

Required/Optional: *Optional*

The definition specifies only the base name of the file, the type of geometry it contains, and names and types of all attributes. The syntax of a Shape DEF line is:

```
<ReaderKeyword>_DEF <baseName> \
  SHAPE_GEOMETRY shape_point | \
    shape_multipoint | \
    shape_polyline | \
    shape_polygon | \
    shape_null | \
    shape_pointm | \
    shape_polylinem | \
    shape_polygonm | \
    shape_pointz | \
    shape_polylinez | \
```

```

shape_polygonz                               \ (
shape_multipatch|                             \
    [<attrName> <attrType>]+

```

Note: In older versions of FME, `shape_polyline` was called `shape_arc`. This has been changed to avoid confusion with mathematical arcs. However, FME still accepts `shape_arc` in place of `shape_polyline` to accommodate backwards compatibility.

Shapefiles with geometry of `shape_point`, `shape_multi_point`, `shape_polyline`, and `shape_polygon` contain two-dimensional features. A geometry of `shape_null` is used for shape files that contain no geometry. If the type of geometry has an `m` at the end, then each two-dimensional coordinate of a feature may optionally have an associated measure value. If the type of geometry has a `z` at the end or is `shape_multipatch`, the shapefile contains three-dimensional features, and each coordinate may optionally have an associated measure value.

The file name of the each of the physical shapefiles is constructed by adding their extension to the base name. The `SHAPE_GEOMETRY` clause specifies the geometry type for the entire file.

It is also possible to store features having no defined geometry. These features have their `SHAPE_GEOMETRY` attribute set to `shape_null`. These `shape_null` features may be stored or read from any type of shapefile.

Tip: When creating Shapefiles, no attributes need to be specified on the `SHAPE_DEF` line. When no attributes are defined on a Shapefile being written, FME automatically generates an `_ID` attribute for the Shapefile. This is useful if the Shapefile is to be imported into ArcInfo. If the Shapefile contained polygons, an `AREA` attribute is also generated. In both cases, the values of these attributes will be `NULL` for all features.

Shapefiles require at least one attribute be defined. The attribute definition given must match the definition of the file being read. If it does not, translation is halted and the true definition of the shapefile's attributes is logged to the log file. All shapefile attribute names must be uppercase and must not exceed 10 characters in length. The following table shows the attribute types that are supported.

Field Type	Description
<code>char (<width>)</code>	Character fields store fixed-length strings. The <code>width</code> parameter controls the maximum characters that can be stored by the field. When a character field is written, it is right-padded with blanks, or truncated, to fit the width. When a character field is retrieved, any padding blank characters are stripped away.
<code>date</code>	Date fields store dates as character strings with the format <code>YYYYMMDD</code> .
<code>logical</code>	Logical fields store <code>TRUE/FALSE</code> data. Data read to or written from such fields must always have a value of either <code>true</code> or <code>false</code> .

Field Type	Description
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.

The following mapping file fragment defines two shapefiles: one containing polygonal features possibly disjoint and with holes, and the other containing linear features:

```
SHAPE_DEF landcover SHAPE_GEOMETRY shape_polygon          \
  AREA number(12,3)                                       \
  TYPE char(11)                                           \
  PERIMETERnumber(12,3)                                   \
SHAPE_DEF roads SHAPE_GEOMETRY shape_arc                 \
  NUMOFLANES number(2,0)                                  \
  TYPE char(5)                                            \
  UNDERCNST logical                                     \
  DIVIDED logical                                        \
  TRVLDIR char(6)
```

Workbench Parameter: [<WorkbenchParameter>](#)

IDs

Required/Optional: *Optional*

This optional specification is used to limit the available and defined shapefiles read. If no `IDs` are specified, then all defined and available shapefiles are read. If more shapefiles were in the directory, they are ignored. The syntax of the `IDs` directive is:

```
<ReaderKeyword>_IDs<baseName1>                          \
                                     <baseName2> ...      \
                                     <baseNameN>
```

The base names must match those used in `DEF` lines.

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

```
SHAPE_IDS roads
```

Workbench Parameter: [<WorkbenchParameter>](#)

MEASURES_AS_Z

Required/Optional: *Optional*

This optional specification controls how measures data associated with geometric data is treated. If the value is `yes`, measures data is treated as elevations.

Workbench Parameter: [<WorkbenchParameter>](#)

SEARCH ENVELOPE

Required/Optional: *Optional*

Defines a rectangular search area. If the shapefiles have spatial indexes (.sbn & .sbx), then only features within the search area will be read.

This optional specification is used to restrict the returned features to those that intersect the defined envelope. The syntax of the `SEARCH_ENVELOPE` directive is:

```
<ReaderKeyword>_SEARCH_ENVELOPE \
    <xmin> <ymin> <xmax> <ymax>
```

This specification will only be used by the reader for datasets that have an associated spatial index (.sbn and .sbx file). If no arguments are given to the `SEARCH_ENVELOPE` directive, or if all the arguments are zero, the search envelope will be disabled.

Workbench Parameter: [<WorkbenchParameter>](#)

SEARCH_ENVELOPE_COORDINATE_SYSTEM

Required/Optional: *Optional*

This directive specifies the coordinate system of the search envelope if it is different than the coordinate system of the data. The `COORDINATE_SYSTEM` directive, which specifies the coordinate system associated with the data to be read, must always be set if the `SEARCH_ENVELOPE_COORDINATE_SYSTEM` directive is set.

If this directive is set, the minimum and maximum points of the search envelope are reprojected from the `SEARCH_ENVELOPE_COORDINATE_SYSTEM` to the reader `COORDINATE_SYSTEM` prior to applying the envelope.

The syntax of the `SEARCH_ENVELOPE_COORDINATE_SYSTEM` directive is:

```
<ReaderKeyword>_SEARCH_ENVELOPE_COORDINATE_SYSTEM <coordinate system>
```

Workbench Parameter: [Search Envelope Coordinate System](#)

DISSOLVE HOLES

Required/Optional: *Optional*

This optional specification controls whether the SHAPE reader dissolves adjacent holes in polygons read from shapefiles. If the value is set to `yes` or is not set, then the SHAPE reader will dissolve adjacent holes.

Workbench Parameter: [<WorkbenchParameter>](#)

REPORT BAD GEOMETRY

Required/Optional: *Optional*

This optional specification controls whether the SHAPE reader reports geometric anomalies in input shapefiles.

By default, the SHAPE reader will perform the following operations to ensure the validity of input features: close unclosed polygons, remove duplicate points, remove empty elements, dissolve holes (if `DISSOLVE_HOLES` is set to `YES` or is not set).

If `REPORT_BAD_GEOMETRY` is set to `YES`, then the `shape_geometry_error{}` list attribute will be set on input features, and will contain error messages as geometric anomalies are detected and/or fixed. The error messages are of the following format:

Closed Polygon at (x,y)
Duplicated Point at (x,y)
Removed Empty Element #n near (x,y)
Removed Duplicate Point at (x,y)
Invalid Polygon/Donut Orientation near (x,y)
Dissolved Holes

Workbench Parameter: [<WorkbenchParameter>](#)

ENCODING

Required/Optional: *Optional*

This optional specification controls which character encoding is used to interpret text attributes from the shapefile. If the value is not set, then the character encoding will be automatically detected from the source shapefile. 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 shapefile is missing or incorrect.

Example:

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

Workbench Parameter: [<WorkbenchParameter>](#)

Parameter	Description
<character encoding>	<p>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</p>

Workbench Parameter: [<WorkbenchParameter>](#)

UPPER_CASE_ATTR_NAMES**Required/Optional:** *Optional*

This option specifies whether the reader should upper case attribute names. If no, it will allow mixed case, otherwise attribute names will be uppercased. The default value is no; however, for backwards compatibility, when this keyword is not present, a value of yes will be used.

This keyword is used when generating workspaces & mapping files. As a result, it is not editable within workbench after the workspace has been generated.

Workbench Parameter: *NOT APPLICABLE***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 Shape writer creates and writes feature data to shapefiles in the directory specified by the DATASET directive. As with the reader, the directory must exist before the translation occurs. Any old shapefiles in the directory are overwritten with the new feature data. As features are routed to the Shape writer by the FME, it determines the file they are to be written to and outputs them according to the type of the file. Many shapefiles can be written during a single FME session.

Writing to shapefiles of type polygonz or polygonm

Writing to shapefiles of type polygonz or polygonm will result in a measures column, whether measures exist or not. If they don't, and the Universal Viewer displays in Classic mode, then geometry shows <1.#QNAN> for the M dimension.

Classic Geometry Handling

```
shape_measures NaN,NaN,NaN,NaN
```

```
1: (4526424.7720003976, 5690159.0588858956, 110.87090000000001)
```

Enhanced Geometry Handling

```
0: (4526424.7720003976,5690159.0588858956,110.87090000000001)<1.#QNAN>
```

Writer Directives

The Shape writer processes the DATASET, DEF, and MEASURES_AS_Z directives as described in the *Reader Directives* subsection. It does not make use of the IDs or SEARCH_ENVELOPE directives.

The `ENCODING` directive is used to specify which character encoding should be used when writing text attributes into shapefiles. 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.

UPPER_CASE_ATTR_NAMES

Required/Optional: *Optional*

This option specifies whether the writer should change attribute names to uppercase text. If set to `NO`, mixed case attribute names will be allowed. The default value is `YES`.

This directive is used when generating workspaces and mapping files. As a result, it is not editable within Workbench after the workspace has been generated.

Workbench Parameter: *NOT APPLICABLE*

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.

Shape features consist of geometry, a special predefined attribute, and a set of user-defined attributes. All shape features have one predefined attribute, `SHAPE_GEOMETRY`, which identifies the type of the features geometry. Geometry types can be two-dimensional (2D), 2D plus elevations, 2D plus measures, or 2D plus elevations and measures:

Attribute Name	Contents
<code>SHAPE_GEOMETRY</code>	The type of the geometry read from the shapefile. This attribute will contain one of: <code>shape_point</code> <code>shape_multipoint</code> <code>shape_polyline</code> <code>shape_polygon</code> <code>shape_null</code> <code>shape_pointm</code> <code>shape_multipointm</code> <code>shape_polylinem</code> <code>shape_polygonm</code> <code>shape_pointz</code> <code>shape_multipointz</code> <code>shape_polylinez</code> <code>shape_polygonz</code> <code>shape_multipatch</code> Default: No default

Attribute Name	Contents
shape_measures	<p>This is present for features that have measures when reading only if the reader directive <code>MEASURES_AS_Z</code> is not specified or is set to <code>no</code>. To write measures using this attribute, make sure the writer directive <code>MEASURES_AS_Z</code> is not specified or is set to <code>no</code>, and simply build this list with one value for each vertex in the feature being written. This is a comma-separated list of floating values which correspond to the vertex measures. The first value is for the first vertex, second for the second, and so on. Not-a-Number values are represented by the string "NaN".</p> <p>However, this will not be present on features if the <code>FME_GEOMETRY_HANDLING</code> directive is set to <code>YES</code>.</p>

A Shapefile defines a set of features that share the same geometry type and the same list of user-defined attributes. In other words, all features belonging to the same shapefile have the same value for the `SHAPE_GEOMETRY` attribute and the same list of user-defined attributes. The values of the user-defined attributes can vary from feature to feature within the same Shapefile. The geometry type and the names of the user-defined attributes for an individual shapefile are specified in the `DEF` line for that Shapefile. The feature type of a Shape feature is the same as the `baseName` specified in the `DEF` line.

When reading Shape features, the `SHAPE_GEOMETRY` attribute will correspond to the geometry type specified in the `DEF` line for that for the shapefile. When writing Shape features, the `SHAPE_GEOMETRY` attribute is not required and will be ignored if it is present because the geometry type is taken from the `DEF` line for the shapefile. If the feature being written out cannot be converted into the geometry type specified on the `DEF` line, this feature will not be written out and a warning message will be printed in the logfile. (An example of this would be trying to write an area feature into a point geometry file.)

There is one exception where the geometry type indicated on the `DEF` line may not be the type of file that is actually created. If the `DEF` line indicates that a point file is to be created, but the first actual feature written to that file is instead a multipoint, a multipoint file will be created instead. (The same will be true for `pointz/multipointz` as well as `pointm/multipointm` files.)

As of ESRI ArcGIS Desktop 9.3 Shape files of type `shape_null` are no longer valid. Any `DEF` lines with `SHAPE_GEOMETRY` set to `shape_null` will be output instead as `shape_point`.

When reading a polyline feature with multiple parts, the FME representation consists of an aggregate of lines. Similarly, when reading a polygon feature with multiple parts, the FME representation consists of an aggregate of polygons. Conversely, when writing aggregates of lines or polygons, the FME will output multi-part polyline and polygon Shape features.