MapInfo Data Interchange Format Reader/Writer

The MapInfo Data Interchange Format (MIF) Reader and Writer modules provide the Feature Manipulation Engine (FME) with the ability to read and write MapInfo import and export files. The MIF is a published ASCII format used by the MapInfo product for input and export. The *MapInfo Reference Manual* describes the MIF format and all constants it uses for color, style, symbol, and fill patterns.

MapInfo Interchange Format Files are often called MIF or MIF/MID files.

Overview

MapInfo is a two-dimensional (2D) system with no provision for transferring elevation data for each vertex in a MapInfo feature. However, point features can define an elevation attribute to store their elevation.

MIF files store both feature geometry and attribution. A logical MIF file consists of two physical files, having the following file name extensions:

File Name Extension	Contents
`.mif`	Vector geometric data
`.mid`	Attributes for the geometric data

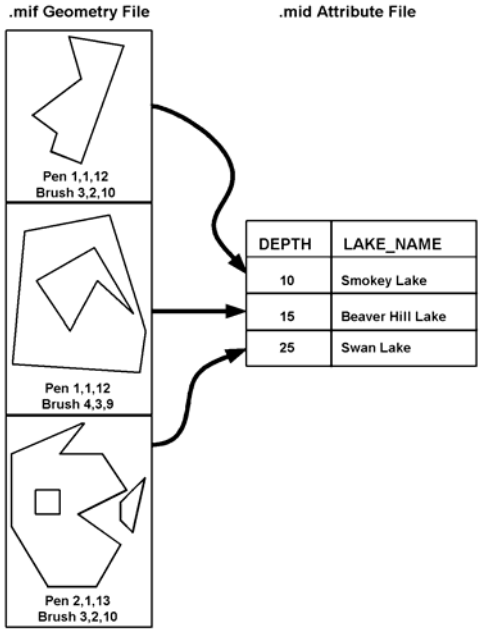
These extensions are added to the basename of the MIF file.

The MapInfo reader and writer support the storage of point, line, polyline, arc, ellipse, rectangle, rounded rectangle, region (polygon), and text geometric data in `.mif` files. The MIF format also stores features with no geometry. Features having no geometry are referred to as having a geometry of *none*.

Each geometric entity present in a `.mif` file has display properties such as pen and brush width, pattern, and color. In addition, each entity has a row of attributes stored in an associated `.mid` file. A single `.mif` file contains many different types of geometry however, the associated attribute in the `.mid` file must have the same number and type of fields for each entity in the `.mif` file. The order of the entries in the two files is synchronized. For example, the second geometric entity in the `.mif` file has the attributes held in the second row of the `.mid` file.

The number and type of attributes associated with each entity is specified by the user. There must be at least one attribute field in the `.mid` file.

The following example shows a MIF file containing three *region* entities in it. Note that 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.



The FME considers a MIF data set to be a collection of MIF files in a single directory. The attribute definitions for each MIF file must be defined in the mapping file before it can be read or written.

MIF Quick Facts

Format Type Identifier	MIF
Reader/Writer	Both
Dataset Type	Directory or File
Feature Type	File base name
Typical File Extensions	`.mif (.mid)`
Automated Translation Support	Yes
User-Defined Attributes	Yes
Coordinate System Support	Yes
Generic Color Support	Yes
Spatial Index	Never
Schema Required	Yes
Transaction Support	No
Geometry Type Attribute	`mif_type`

Geometry Support

Geometry	Supported	Geometry	Supported
aggregate	yes	polygon	yes
circles	yes	donut polygon	yes
circular arc	yes	line	yes
elliptical arc	yes	point	yes
ellipses	yes	text	yes
none	yes	3D	no

Reader Overview

The MIF reader first scans the directory it is given for the MIF files defined in the mapping file. For each MIF file that it finds, it checks to see if it 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 MIF file is opened. The MIF reader then extracts features from the file one at a time, and passes them on to the rest of the FME for further processing. When the file is exhausted, the MIF reader starts on the next file in the directory.

Optionally a single MIF file can be provided as the dataset. In this case, only that MIF file will be read.

Reader Keywords

The following table lists the keywords processed by the MIF reader. The suffixes shown are prefixed by the current <ReaderKeyword> in a mapping file. By default, the <ReaderKeyword> for the MIF reader is MIF.

Keyword Suffix	Value	Required/Optional
DATASET	Contains the directory name of the input MIF files, or a single MIF file.	Required
DEF	Defines a MIF file. The definition contains the file's base name without any of the extensions, and the definitions of the attributes. There may be many DEF lines, one for each file to be read.	Optional
IDs	Contains a list of MIF files to process. If this is not specified, then all defined MIF files in the directory are read.	Optional
BREAK_COLLECTION	Specifies how MIF collection features are handled. If YES, collection features are broken down into their component parts before being returned to FME. If NO, then the collections are preserved (usually this is only of use when doing a MIF to MIF translation). The default is YES.	Optional

DATASET

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

```
MIF_DATASET /usr/data/mapinfo/92i080
```

DEF

The definition specifies the base name of the file, and the names and the types of all attributes. The syntax of a MIF DEF line is:

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

The file names of the physical MIF files is constructed by using the directory specified by the DATASET keyword, the basename specified on the MIF DEF lines, and the .mif (geometry) and .mid (attributes) extensions.

MIF files require at least one attribute to 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 MIF file's attributes gets logged to the log file. There are no restrictions on the field names of MIF attributes.

Tip

MapInfo `decimal` fields are analogous to DataBase Format (DBF) `number` fields. MapInfo also provides `float`, `integer`, and `smallint` field types for storing numeric values.

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 dates as character strings with the format YYYYMMDD.
`decimal(<width>, <decimals>)`	Decimal 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.
`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.
`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`.
`smallint`	Small integer fields store 16 bit signed integers and therefore have a range of -32767 to +32767.

The following mapping file fragment defines two MIF files. Notice that neither definition specifies the geometric type of the entities it will contain since MIF files may contain any of the valid geometry types.

```
MIF_DEF landcover                                    \
     area              decimal(12,3)                 \
     landcoverType     char(11)                      \
     perimeter         float
MIF_DEF roads                                        \
     numberOfLanes     smallint                      \
     roadType          char(5)                       \
     underConstruction logical                       \
```

```
divided           logical                              \
travelDirection   char(6)
```

IDs

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

The syntax of the `IDs` keyword is:

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

The basenames must match those used in `DEF` lines.

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

```
MIF_IDs roads
```

BREAK_COLLECTION

This directive specifies how the MIF collections are processed. If no `BREAK_COLLECTION` is specified, then all MIF collections are broken down into their component parts before being returned to FME. If a MIF-to-MIF translation is being performed, then this may be set to `NO` to preserve the collections as single features.

This example shows how collections may be preserved:

```
MIF_BREAK_COLLECTION NO
```

Writer Overview

The MIF writer creates and writes feature data to MIF files in the directory specified by the `DATASET` keyword. As with the reader, the directory must exist before the translation occurs. Any old MIF files in the directory are overwritten with the new feature data. As features are routed to the MIF writer, the MIF writer determines the file into which the features are written and outputs them accordingly. Many MIF files can be written during a single FME session.

When the MIF writer receives a feature with an `fme_color` or `fme_fill_color` attribute, the writer will honor the color values. The only exception is when native MapInfo color settings are also present, in which case the native settings will take precedence.

Writer Keywords

The MIF writer processes the DATASET and DEF keywords as described in the *Reader Keywords* section. It does, however, make use of some additional keywords:

Keyword Suffix	Value	Required/Optional
DATASET	Contains the directory name of the output MapInfo files.	Required
DEF	Defines a MapInfo file. The definition contains the file's base name (without any of the extensions), and the definitions of the attributes. There may be many DEF lines, one for each file to be written.	Required
COORDSYS_ STATEMENT	Contains a MIF specific coordinate system string to be used for the output files.	Optional
BOUNDS	Specifies the bounds which should be used to limit the range of the output MIF files. Because MapInfo has limited precision available for the storage of coordinates, defining a tight bound on the data can preserve more accuracy. The syntax of this directive is: MIF_BOUNDS <xmin> <ymin> <xmax> <ymax>	Optional
FILENAME_ PREFIX	The value given this directive is prepended to every file output by the writer.	Optional
WRITE_REGION_ CENTROIDS	Directs the writer to output region centroids. Values: yes or no Default: no	Optional

COORDSYS_STATEMENT

The value for this keyword is the coordinate system statement that should be written to the header of the produced MIF files. Normally, FME examines the coordinate system information present on the features written to the MIF files, and generates a coordinate system statement based on this information. However, in certain circumstances it is necessary to override this and force a particular coordinate system to be output into the file. This is typically done to force the units of a *non-earth* projection to something other than the default, which are metres.

The syntax of this line is the same as the line defined for the CoordSys line in the MapInfo MIF/MID documentation. For example, to force a non-earth inches coordinate system, this line would be present in the mapping file:

```
MIF_COORDSYS_STATEMENT CoordSys NonEarth Units \"in\"
```

The FME appends bounds information to this statement when it is written to the MIF file. Notice that the quotes must be escaped, as they are required when the coordinate system statement is written to the MIF file.

BOUNDS

This directive allows explicit setting of the bounds of the output features. Because MIF has limited precision available for the storage of coordinates, defining a tight bound on the range of the data can preserve more accuracy. When this directive is specified, the coordinate system string written to the top of the MIF file will contain this bounds specification. The syntax of this directive is:

```
MIF_BOUNDS<xmin> <ymin> <xmax> <ymax>
```

FILENAME_PREFIX

The value for this keyword is prepended to every output file that is created by the writer.

For example, to have the word `temp` appear on the front of every file name, this line would be present in the mapping file:

```
MIF_FILENAME_PREFIX temp
```

WRITE_REGION_CENTROIDS

To direct the Writer to output region centroids, the syntax of this directive is:

```
WRITE_REGION_CENTROIDS yes
```

Feature Representation

MIF features consist of geometry and attributes. The attribute names are defined in the `DEF` line and there is a value for each attribute in each MIF feature. In addition, each MIF feature contains several special attributes to hold the type of the geometric entity and its display parameters. All MIF features contain the `mif_type` attribute, which identifies the geometric type. All MIF features may contain either or both of the `fme_color` and `fme_fill_color` attributes, which store the color and fill color of the feature respectively.

In addition to the generic FME feature attributes that FME Workbench adds to all features (see *About Feature Attributes* on page 1), this format adds the format-specific attributes described in this section.

Attribute Name	Contents
mif_type	The MIF geometric type of this entity. **Range:** mif_point| mif_polyline| mif_region| mif_text| mif_ellipse| mif_arc| mif_rectangle| mif_rounded_rectangle| mif_collection| mif_none **Default:** No default
fme_color	A normalized RGB triplet representing the color of the feature, with format r,g,b. **Range:** 0,0,0 to 1,1,1 **Default:** No default
fme_fill_color	A normalized RGB triplet representing the fill color of the feature, with format r,g,b. **Range:** 0,0,0 to 1,1,1 **Default:** No default

Points

mif_type: mif_point

MIF point features specify a single x and y coordinate in addition to any associated user-defined attributes. An aggregate of point features may also be read or written – this corresponds to the MIF MULTI_POINT primitive type.

A MIF point also specifies a symbol. The symbol is defined by a symbol number, a color, and a size. If no symbol is defined for a point entity, the previous symbol is used.

The table below lists the special FME attribute names used to control the MIF symbol settings.[1]

Attribute Name	Contents
mif_symbol_color	The color of the symbol. MapInfo colors are defined in relative concentrations of red, green, and blue. Each color ranges from 0 to 255, and the color value is calculated according to the formula: (red*65536) + (green*256) + blue **Range:** 0…2^24 - 1 **Default:** 0 (black)

Attribute Name	Contents
mif_symbol_shape	The number of the symbol. See the *MapInfo Reference Manual* for a list of the available symbols. **Range:** 31...67 **Default:** 35 (a star)
mif_symbol_size	The point size of the symbol. Note that this size is *not* scaled depending on the zoom level. **Range:** Any integer number > 0 **Default:** 10

Font Points

mif_type: mif_font_point

MIF font point are very similar to MIF points, but allow a symbol based on a TrueType font to be specified. In addition to the font, may specify rotation, color, shape number, size, and style.

The table below lists the special FME attribute names used to control the MIF font point settings:

Attribute Name	Contents
mif_symbol_color	The color of the symbol calculated according to the formula: (red*65536) + (green*256) + blue **Range:** 0…2^24 - 1 **Default:** 0 (black)
mif_symbol_shape	The number of the shape within the TrueType font to be used as the symbol. **Range:** Integer **Default:** No default
mif_symbol_size	The point size of the symbol. **Range:** Integer **Default:** 12
mif_symbol_font	The name of the TrueType font to be used for the symbol. **Range:** String **Default:** No default

1. MapInfo symbols cannot be rotated. However, some third-party add-ons to MapInfo rotate symbols based on a user-defined rotation attribute.

Attribute Name	Contents
mif_symbol_angle	The rotation angle for the symbol, measured in degrees counter-clockwise from horizontal. **Range:** -360.0..360.0 **Default:** 0
mif_symbol_style	The display style for the symbol. **Range:** 0 (Plain text) 1 (Bold text) 16 (Black border around symbol) 32 (Drop Shadow) 256 (White border around symbol) **Default:** 0

Custom Points

mif_type: mif_custom_point

MIF custom points are very similar to MIF points, but allow a bitmap image to be specified as the symbol to be drawn. In addition to the image, color, size, and style may be specified.

The table below lists the special FME attribute names used to control the MIF custom point settings:

Attribute Name	Contents
mif_symbol_color	The color of the symbol calculated according to the formula: (red*65536) + (green*256) + blue Whether or not the color is used depends on the setting of the style attribute. **Range:** 0…2^24 - 1 **Default:** 0 (black)
mif_symbol_file_name	The name of the bitmap file found in the MapInfo CustSymb directory. **Range:** String **Default:** No default
mif_symbol_size	The point size of the symbol. **Range:** Integer **Default:** 12

Attribute Name	Contents
mif_symbol_style	The display style for the symbol. **Range:** 0 (White pixels in the image are transparent, allowing whatever is beneath to show through. Non-white pixels are drawn in the same color as they are in the bitmap.) 1 (White pixels in the image are drawn as white. Non-white pixels are drawn in the same color as they are in the bitmap.) 2 (White pixels in the image are transparent. Non-white pixels are drawn in the color specified by mif_symbol_color.) 3 (White pixels in the image are drawn in white. Non-white pixels are drawn in the color specified by mif_symbol_color) **Default:** 0

Multipoints

mif_type: mif_point, mif_font_point, mif_custom_point

MIF multipoint feature specify a number of individual sets of points each defined by an x and y coordinate. All the points share the same attributes and geometry. This is supported as a homogeneous aggregate feature composed of points, font points or custom points.

The MIF multipoint uses the same attribute names control settings as the points, font points and custom point.

Polylines

mif_type: mif_polyline

MIF polyline features specify linear features defined by a sequence of x and y coordinates. Each polyline has a pen style associated with it specifying the color, width, and pen pattern of the line. A polyline may also specify that it is a smoothed line, in which case MapInfo uses a curve fitting algorithm when rendering the line[2]. If no pen style is defined, the previous style is used.

Tip

MapInfo MIF supports a special type for two point lines. The FME transparently converts such MIF *lines* into polylines, both as it reads MIF files and as it writes them.

2. MapInfo renders smoothed polylines substantially slower than unsmoothed polylines.

The table below lists the special FME attribute names used to control the MIF polyline settings.

Attribute Name	Contents
mif_pen_color	The color of the polyline. MapInfo colors are defined in relative concentrations of red, green, and blue. Each color ranges from 0 to 255, and the color value is calculated according to the formula: (red*65536) + (green*256) + blue **Range:** 0…2^24 - 1 **Default:** 0 (black)
mif_pen_pattern	The pattern used to draw the line. See the *MapInfo Reference Manual* for a list of the available patterns. **Range:** 1…77 **Default:** 2
mif_pen_width	The width of the line rendered for the polyline feature. This is measured as a thickness in pixels. A width of 1 is always drawn as a hairline. A width of 0 should be considered to be a line with no width, or a line with no style, or invisible, and should not normally be used. If an invisible line is necessary, it should be created by setting the pattern to 1 (None). If a hairline is desired, the pen should be created by setting the width to 1. The width can be specified as a point width, in which case this formula is used: `penwidth = (point width * 10) + 10` **Range:** 0...7 (pixel width) 11...2047 (point width) **Default:** 1
mif_smooth	Controls whether or not the polyline will be smoothed when rendered. **Range:** truelfalse **Default:** false

Regions

mif_type: mif_region

MIF region features specify area (polygonal) features. The areas that make up a single feature may or may not be disjoint, and may contain polygons that have holes. Each region has a pen style associated with it to control the color, width, and pen pattern used when its boundary is drawn. In addition, a region may set its brush pattern, foreground, and background color to control how its enclosed area will be filled. If no pen or brush style is defined for a region entity, the

previous style is used. The following table lists the special FME attribute names used to control the MIF region settings.

Attribute Name	Contents
mif_brush_pattern	The pattern used to fill the area the region contains. See the *MapInfo Reference Manual* for a list of the available brush patterns. **Range:** 1…71 **Default:** 2 (solid)
mif_brush_foreground	The foreground color used when the region is filled. MapInfo colors are defined in relative concentrations of red, green, and blue. Each color ranges from 0 to 255, and the color value is calculated according to the formula: (red*65536) + (green*256) + blue **Range:** 0…2^24 - 1 **Default:** 0 (black)
mif_brush_background	The background color used when the region is filled. (-1 specifies transparent color) **Range:** -1…2^24 - 1 **Default:** 16777215 (white)
mif_pen_color	The color of the boundary of the region. **Range:** 0…2^24 - 1 **Default:** 0 (black)
mif_pen_pattern	The pattern used to draw the region's boundary. See the *MapInfo Reference Manual* for a list of the available patterns. **Range:** 1…77 **Default:** 2
mif_pen_width	The width of the line rendered for the region's boundary. This is measured as a thickness in pixels. A width of 1 is always drawn as a hairline. A width of 0 should be considered to be a line with no width, or a line with no style, or invisible, and should not normally be used. If an invisible line is necessary, it should be created by setting the pattern to 1 (None). If a hairline is desired, the pen should be created by setting the width to 1. **Range:** 0...35 **Default:** 1
mif_center_xcoord	The centroid x coordinate. **Range:** Any real number **Default:** 0
mif_center_ycoord	The centroid y coordinate. **Range:** Any real number **Default:** 0

Text

mif_type: mif_text

MIF text features are used to specify annotation information. Each text feature can have its font, color, spacing, justification, and rotation angle set independently. The following table lists the special FME attribute names used to control the MIF text settings.

Attribute Name	Contents
mif_rotation	The rotation of the text, as measured in degrees counterclockwise from horizontal. **Range:** -360.0..360.0 **Default:** 0
mif_text_fontbgcolor	The background color used when the text is drawn. **Range:** 0…2^24 - 1 **Default:** 16777215 (white)
mif_text_fontfgcolor	The foreground color used when the text is drawn. MapInfo colors are defined in relative concentrations of red, green, and blue. Each color ranges from 0 to 255, and the color value is calculated according to the formula: (red*65536) + (green*256) + blue **Range:** 0…2^24 - 1 **Default:** 0 (black)
mif_text_fontname	The name of the font used to draw the text. The font named must be available on the destination computer system. **Range:** Any valid system font **Default:** Helvetica
mif_text_fontstyle	The style code of the text. This controls if the text is bold, underlined, or italic. See the *MapInfo Reference Manual* for a list of style codes and their meanings. **Range:** 0…7 **Default:** 0 (regular text)
mif_text_height	The height of the text in ground units. **Range:** Any real number >= 0 **Default:** 10
mif_text_justification	The justification of the text. **Range:** left | center | right **Default:** left
mif_text_spacing	The spacing between lines of multiline text. The measure is expressed as a multiple of the text height. **Range:** 1.0 | 1.5 | 2.0 **Default:** 1.0

Attribute Name	Contents
mif_text_string	The text to be displayed. **Range:** Any character string **Default:** No default
mif_text_width	The total width of the text string in ground units. The MIF text representation stores a bounding box for the text, and mif_text_width is the width of the bounding box. **Range:** Any real number >= 0 **Default:** 10
mif_text_linetype	The type of line attaching the text to the anchor point. **Range:** 0 (None: do not display a line with the label.) 1 (Simple: create a callout by using a simple line that connects the label to the anchor point.) 2 (Arrow: create a callout by using an arrow and line that connects the label to anchor point.) **Default:** 0 (None)
mif_text_line_end_x	The x position of the label line end point. The linetype needs to be 1 or 2 for the label line to be visible. **Range:** Any real number **Default:** No default
mif_text_line_end_y	The y position of the label line end point. The linetype needs to be 1 or 2 for the label line to be visible. **Range:** Any real number **Default:** No default

Tip

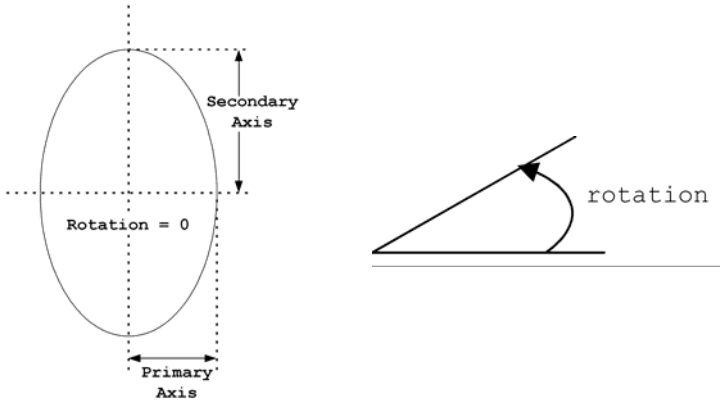
The font color and style settings will not be used unless a font name is specified.

Ellipse

mif_type: mif_ellipse

MIF ellipse features are point features, and have only a single coordinate. This point serves as the centre of the ellipse. Additional attributes specify the primary axis and secondary axis of the ellipse. MIF ellipses currently do not support rotation. For compatibility with other systems, the MIF reader always returns a

rotation of 0. If a rotation is specified to the writer, the ellipse is turned into a region, vectorized, and rotated by the amount specified.



Tip

The primary ellipse axis is **not** necessarily the longest axis, but rather the one on the x axis.

In addition to the attributes below, ellipses also make use of the brush and pen attributes as defined by `mif_region`.

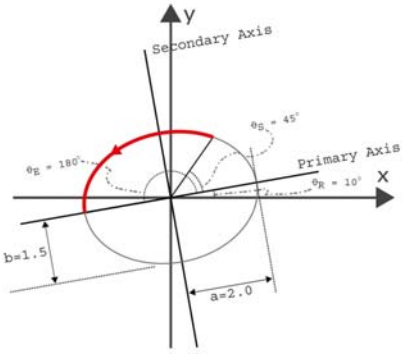
Attribute Name	Contents
`mif_primary_axis`	The length of the semi-major axis in ground units. **Range:** Any real number > 0 **Default:** No default
`mif_secondary_axis`	The length of the semi-minor axis in ground units. **Range:** Any real number > 0 **Default:** No default
`mif_rotation`	The rotation of the major axis. The rotation is measured in degrees counterclockwise up from horizontal. **Range:** -360.0..360.0 **Default:** 0

Arc

`mif_type:` `mif_arc`

MIF arc features are linear features used to specify elliptical arcs. As such, the feature definition for `mif_arc` is similar to the ellipse definition with two additional angles to control the portion of the ellipse boundary drawn. MIF arcs currently do not support rotation. For compatibility with other systems, the MIF

reader always returns a rotation of 0. In addition, if a rotation is specified to the writer, the arc is turned into a polyline, vectorized, and rotated by the amount specified.



Tip

The function `@Arc()` can be used to convert an arc to a linestring. This is useful for storing Arcs in systems not supporting them directly.

In addition to the attributes below, arcs also make use of the pen attributes as defined on `mif_polyline`.

Attribute Name	Contents
`mif_primary_axis`	The length of the semi-major axis in ground units. **Range:** Any real number > 0 **Default:** No default
`mif_secondary_axis`	The length of the semi-minor axis in ground units. **Range:** Any real number > 0 **Default:** No default
`mif_start_angle`	The start angle defines the counterclockwise distance from the primary axis to the starting point of the arc. It is measured in degrees. **Range:** 0.0..360.0 **Default:** 0
`mif_sweep_angle`	The sweep angle defines the sweep of the arc from the starting point along the ellipse boundary in the counterclockwise direction. It is measured in degrees. **Range:** 0.0..360.0 **Default:** No default

Attribute Name	Contents
mif_rotation	The rotation of the major axis. The rotation is measured in degrees counter clockwise up from horizontal. **Range:** -360.0..360.0 **Default:** 0

Rectangle

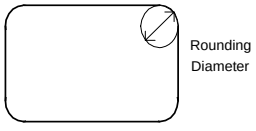
mif_type: mif_rectangle

MIF rectangle objects are represented in the FME as closed polygons. When a MIF rectangle is read, it is turned into a closed polygon feature. When a MIF rectangle is written, the minimum bounding rectangle of the feature is taken and used as the four corners of the rectangle. MIF rectangles take the same additional attributes as MIF regions to specify their brush and pen.

Rounded Rectangle

mif_type: mif_rounded_rectangle

MIF rounded rectangle objects are represented in the FME as closed polygons. When a MIF rounded rectangle is read, it is turned into a closed polygon feature and the corners are vectorized to preserve the intended shape of the rectangle. The rounding radius is also stored as an attribute. When a MIF rounded rectangle is written, the minimum bounding rectangle of the feature is taken and used as the four corners of the rectangle, and the rounding diameter is taken from an attribute of the feature. MIF rounded rectangles take the same additional attributes as MIF regions to specify their brush and pen.



Attribute Name	Contents
mif_rounding	Contains the diameter in ground unit, of the circle used to produce the rounded corners. **Range:** Any real number > 0 **Default:** No default

Collection

mif_type: mif_collection

MIF collections are defined as a combination of the other feature types. This is represented as nonhomogeneous aggregates composed of the other feature types.

The table below lists the special FME attribute name used to control the MIF collection settings:

Attribute Name	Contents
mif_collection_cmp{}	This is the list attribute prefix used to store the attributes for each collection part. The suffixes are the attribute names for the control settings of the other feature types. **Range:** none **Default:** none

Example 1

The following mapping file example translates linear road features from Shape file format into MIF. This example illustrates how specify the definition of the output MIF file using the DEF keyword. Notice how the mif_pen_width, mif_color, and mif_line_pattern attributes are used to control the MIF polyline settings based on the convenience macros defined in mapinfoMacros.fmi. The @Transform function converts the shape_type attribute to the appropriate mif_type. This allows a simple correlation that copies everything from the input to the output to be used.

```
#-------------------------------------------------------------------------
# Set up the Shape Reader
READER_TYPE SHAPE
SHAPE_DATASET "C:/Input"
SHAPE_DEF roads                                                          \
    SHAPE_GEOMETRY       shape_arc                                       \
    CLASS                number(5,0)                                     \
    STYLE                number(5,0)                                     \
    WEIGHT               number(5,0)                                     \
    LEVEL                number(5,0)                                     \
    COLOR                number(5,0)

#-------------------------------------------------------------------------
# Set up the MIF Writer
WRITER_TYPE MIF
INCLUDE "$(FME_HOME)/metafile/mapinfoMacros.fmi"
MIF_DATASET "C:/Output"
MIF_DEF roads                                                            \
    CLASS                decimal(5,0)                                    \
    STYLE                decimal(5,0)                                    \
    WEIGHT               decimal(5,0)                                    \
```

```
    LEVEL                decimal(5,0)                                    \
    COLOR                decimal(5,0)

#-------------------------------------------------------------------------
# Transform road features from SHAPE to MIF
FACTORY_DEF * TeeFactory                                                 \
    INPUT  FEATURE_TYPE *                                                \
    OUTPUT FEATURE_TYPE *                                                \
        @Transform(SHAPE, MIF)                                           \
        mif_pen_width $(MAPINFO_thick)                                   \
        mif_pen_pattern $(MAPINFO_thickRoad)                             \
        mif_pen_color $(MAPINFO_White)

#-------------------------------------------------------------------------
# Correlate everything from the input to the output
SHAPE *
MIF *
```

Tip

Using macros for the MIF colors, symbols, linestyles, and fill patterns makes mapping file maintenance and authoring much simpler.

Example 2

The following mapping file example merges many MapInfo Interchange (`MIF`) into a single MapInfo Native (`TAB`) file. This example illustrates how to configure the MIF reader to read all the files in the input directory, by not defining the `MIF_ID` or `MIF_DEF` keywords. The attributes to be saved in this example are "name" and "width": these are defined in the output `MAPINFO_DEF` line and are assumed to be present in all the input files. The `TeeFactory` is used to perform two useful services. First, it renames the feature type of all the input features to `merged` so that the match the `MAPINFO_DEF` specification. Second, it uses the `@Transform` function to convert the `mif_type` attribute to the appropriate `mapinfo_type`, allowing a simple correlation that copies everything from the input to the output to be used.

```
#-------------------------------------------------------------------------
# Set up the MIF Reader
READER_TYPE MIF
MIF_DATASET "C:/Input"

#-------------------------------------------------------------------------
# Set up the TAB Writer
WRITER_TYPE MAPINFO
MAPINFO_DATASET "C:/Output"
MAPINFO_DEF merged                                                       \
   name char(10)                                                         \
   width integer

#-------------------------------------------------------------------------
# Transform  features from MIF to TAB
FACTORY_DEF * TeeFactory                                                 \
   INPUT FEATURE_TYPE *                                                  \
```

```
   OUTPUT FEATURE_TYPE merged                                                \
       @Transform(MIF,MAPINFO)

#-------------------------------------------------------------------------
# Correlate everything from the input to the output
MIF *
MAPINFO *
```

Example 3

This mapping file "merges" all MIF files in a directory into a single MAPINFO TAB file. It is assumed that the structure of the MIF files is the same, or at least, that the attributes that are to be saved to the output TAB file are similarly named in all the input MIF files.

To use this mapping file, you'll have to modify the `MAPINFO_DEF` line at the bottom, and alter the `_DATASET` lines to point to your directories

Note that we have NO `MIF_IDs` or `MIF_DEF` lines in this mapping file; in their absence, the MIF reader reads everything it finds in the input directory.

```
READER_TYPE MIF

# Point this to where your input data is

MIF_DATASET v:\test


WRITER_TYPE MAPINFO

# Point this to where you want your output data

MAPINFO_DATASET v:\testout

# This factory renames the feature type of all the input features
# to "merged" (you could change this).  This name must match the MAPINFO_DEF
# at the end of the mapping file.  @Transform is used to convert
# the geometry specific attributes from MIF to MAPINFO on each feature
# as it goes by.

FACTORY_DEF * TeeFactory                                                   \
   INPUT FEATURE_TYPE *                                                    \
   OUTPUT FEATURE_TYPE merged @Transform(MIF,MAPINFO)


# This kind of correlation copies everything from the input
# to the output.  Note that the @Transform above
# caused the mif_type attribute to be converted to the
# appropriate mapinfo_type

MIF *

MAPINFO *
```

```
# Now you'd define your output file -- note that it should
# have the same name as what you changed the feature type's
# to in the above factory

MAPINFO_DEF merged                                                    \
   dummy char(10)
```