

## Appendix J: MapInfo® Data Interchange Format

This appendix describes the data interchange format for MapInfo Professional®. In this appendix, you will find information on:

- MIF File Header

- MIF Data Section

- Pen, Brush, Symbol, and Font Codes in MIF

- MID File

This versatile format allows generic data to be attached to a variety of graphical items. Since it is ASCII, it is editable, relatively easy to generate, and works on all platforms supported by MapInfo. Perhaps the best way to understand the MapInfo Interchange Format (MIF) is to study the sample file at the end of this appendix in conjunction with the explanation of the file format. You can also create samples of your own by exporting files to MIF and then examining those files in a text editor.

MapInfo data is in two files — the graphics reside in a .MIF file and textual data is contained in a .MID file. The textual data is delimited data, with one row per record and either Carriage Return, Carriage Return plus Line Feed, or Line Feed between lines. The MIF file has two areas — the file header area and the data section. Information on how to create MapInfo tables is in the header; the graphical object definitions are in the data section.

### MIF File Header

This is a description of MIF file header with optional information in square brackets.

```
VERSION n

Charset "characterSetName"

[ DELIMITER "<c>" ]

[ UNIQUE n,n.. ]

[ INDEX n,n.. ]

[ COORDSYS... ]

[ TRANSFORM... ]

COLUMNS n

  <name> <type>

  <name> <type>

  .

  .

DATA
```

#### Version

- TAB files are always saved as 300 unless noted below.
- WOR files are always saved as 400 unless noted below.
- Once a table is 'upgraded' to a later version, MapInfo *Professional* does not 'downgrade' it to an earlier version if the feature which forced the 'upgrade' is removed.
- Workspaces are written fresh each time, and so can be 'downgraded' if there are no features that require the 'upgrade'.
- MIF/MID files created by exporting TAB files from MapInfo *Professional* match the version of the TAB file.
- Pen(0,0,0) and Brush(0,0,0) which were valid in 4.1 are not valid in 4.5 and later.
- Line widths in points are encoded in the Pen clause by multiplying the line width value by 10 and adding 10 to the result (.2 -> 12, 1.0 -> 20, etc.). This forces a 450 in the MIF file header.
- Interleaved line styles are encoded in the Pen clause by increasing the line style value by 128. This does not force a 450 workspace, appears to be 4.x compatible, and gets a 400.

File versions:

Version Number	TAB	MIF-MID	WOR	Action
400				
	X			Table is a linked ODBC table
	X	X	X	Table uses interleaved line styles or objects in the cosmetic layer interleaved line styles
			X	Saved queries (controlled by 'Save Queries in Workspaces' check box in Options>Preferences>Startup dialog) are implemented as straight Select statements
410				
	X	X		Table uses MS Access DB's
450				
	X	X		Region and polyline objects, that have more than 32K nodes (actually !edit_version 450 and !version 300)
		X	X	Table uses point sized line widths
		X	X	Table is a query table
			X	Workspaces that specify dot density color (shade ... density... color)
			X	Workspaces that specify Line width in points (Pen (12,x,x))
452				
		X	X	Regional Mercator projection - number 26
500				
		X	X	Table specifies a 'grid' file (*.MIG, raster style = 6 1)
			X	Workspaces that contain surface thematic layers (inflect)
			X	Workspaces that contain cartographic legends (Create Cartographic Legend)
550				
		X	X	Polyconic projection - number 27
		X	X	Irish (WOFO) ellipsoid - number 49

## Appendix J: MapInfo Data Interchange Format

Version Number	TAB	MIF-MID	WOR	Action
		X	X	Table is a linked\live Oracle 8i table
			X	Workspaces with Oracle 8i connection information
			X	Workspaces that include surface thematics with Hillshading
600				
			X	Save a workspace with a 3D Map
	X	X		Azimuthal Equidistant projection, Oblique aspect - number 28
	X	X		Everest (Pakistan) ellipsoid - number 50
	X	X		ATS 77 ellipsoid - number 51
				<p><b>New Datums introduced in Version 6.0:</b>            Numbered 115-150 (115 and 150 included) and 1004-1011 (1004 and 1011 included).            MapInfo Professional writes the datum number if all the parameters match the parameters of the internal datum table. Also note, that MapInfo writes the FIRST datum number with these parameters, that is found in the internal table. That means that if, for example, a table was created using datums EUREF89 (115), GDA94 (116) or NZGD2000 (117), the MIF file would have datum GRS 80 (33) written into it and the MIF file version would not be incremented            Since MapInfo does not write the datum number into the TAB file, the version (number) will be incremented only if the ellipsoid is new.</p>
			X	Workspace includes Advanced Printer settings
			X	Workspace includes a Hotlink
			X	Workspace includes a 3D Mapper window
			X	Workspaces that include new clip region settings
650				
	X	X		New object types: Multipoint and Collection (actually !edit_version 650 and !version 300)
	X	X		Cassini-Soldner projection (number 30)

Version Number	TAB	MIF-MID	WOR	Action
	X	X		Lambert Azimuthal Equal-Area projection - Oblique aspect(number 29)
			X	Cassini-Soldner projection for the map window
			X	Lambert Azimuthal Equal-Area projection (Oblique aspect) for the map window
			X	New object types (Multipoint and Collection) created in a cosmetic layer or in a layout
			X	Save a workspace with a Prism Map

### Charset

The Charset clause specifies which character set was used to create text in the table. For example: Specify "WindowsLatin1" to indicate that the file was created using the Windows US & Western Europe character set; specify "MacRoman" to specify the Macintosh US & Western Europe character set; or specify "Neutral" to avoid converting the text into another character set. If you are not using one of these character sets, you can determine the correct syntax for your character set by exporting a table and examining the .MIF file in a text editor.

### Delimiter

Specify the delimiting character in quotation marks, for example:

**DELIMITER ";"**

The default delimiter is Tab; if you are using the default, you do not need the DELIMITER line.

### Unique

Specify a number. This number refers to a database column; 3 is the third column, 7 is the seventh column, and so forth. What happens to columns in the UNIQUE list is subtle. For example, imagine that you have a database with highways in it. Each highway has only one name, but it might be represented by several segments. You would put the NAME column in the UNIQUE list, while the column containing data for the individual segments would not be in that list. This has the effect of creating two related tables; one with names, and one with the other attributes of the objects. This is how MapInfo's various street maps (StreetInfo) are prepared.

### Index

To indicate that columns in the table are indexed, include a number (or a comma-separated list of numbers) in the Index clause. Each number refers to a database column; 3 is the third column, 7 is the seventh column, and so forth. Columns in the INDEX list will have indexes prepared for them.

### CoordSys Clause

Specify the COORDSYS clause to note that the data is not stored in longitude/latitude form. When no COORDSYS clause is specified, data is assumed to be stored in longitude/latitude forms.

All coordinates are stored with respect to the northeast quadrant. The coordinates for points in the United States have a negative X while coordinates for points in Europe (east of Greenwich) have a positive X. Coordinates for points in the Northern hemisphere have a positive Y while coordinates for points in the Southern hemisphere have a negative Y.

**Syntax1**

```
CoordSys Earth
[ Projection type,
    datum,
    unitname
  [ , origin_longitude ]
  [ , origin_latitude ]
  [ , standard_parallel_1 [ , standard_parallel_2 ] ]
  [ , azimuth ]
  [ , scale_factor ]
  [ , false_easting ]
  [ , false_northing ]
  [ , range ] ]
[ Affine Units unitname, A, B C, D, E, F ]
[ Bounds ( minx, miny ) ( maxx, maxy ) ]
```

**Syntax2**

```
CoordSys Nonearth
[ Affine Units unitname, A, B C, D, E, F ]
Units unitname
Bounds ( minx, miny ) ( maxx, maxy)
```

**Syntax3**

```
CoordSys Layout Units paperunitname
```

**Syntax4**

```
CoordSys Table tablename
```

**Syntax5**

```
CoordSys Window window_id
```

*type is a positive integer value representing which coordinate system to use*

*datum is a positive integer value identifying which datum to reference*

*unitname is a string representing a distance unit of measure (e.g. -m" for meters); for a list of unit names, see Set Distance Units*

*origin\_longitude is a float longitude value, in degrees*

## Appendix J: MapInfo Data Interchange Format

---

*origin\_latitude* is a float latitude value, in degrees

*standard\_parallel\_1* and *standard\_parallel\_2* are float latitude values, in degrees

*azimuth* is a float angle measurement, in degrees

*scale\_factor* is a float scale factor

*range* is a float value from 1 to 180, dictating how much of the Earth will be seen

*minx* is a float specifying the minimum x value

*miny* is a float specifying the minimum y value

*maxx* is a float specifying the maximum x value

*maxy* is a float specifying the maximum y value

*paperunitname* is a string representing a paper unit of measure (e.g. -in" for inches);

for a list of unit names, see Set Paper Units

*tablename* is the name of an open table

*window\_id* is an Integer window identifier corresponding to a Map or Layout window

*A* performs scaling or stretching along the X axis.

*B* performs rotation or skewing along the X axis.

*C* performs shifting along the X axis.

*D* performs scaling or stretching along the Y axis.

*E* performs rotation or skewing along the Y axis.

*F* performs shifting along the Y axis.

### Transform Clause

When you have MIF files with coordinates stored with respect to the northwest quadrant (quadrant 2), you can transform them to the northeast quadrant (quadrant 1) with a transform clause.

Quadrant 2: Northwest Quadrant	Quadrant 1: Northeast Quadrant
Quadrant 3: Southwest Quadrant	Quadrant 4: Southeast Quadrant

The transform clause has the following syntax:

**TRANSFORM *Xmultiplier*, *Ymultiplier*, *Xdisplacement*,  
*Ydisplacement***



To transform quadrant 2 data into quadrant 1 data, use the following transform clause:

```
TRANSFORM -1,0,0,0
```

The zeroes instruct MapInfo to ignore that parameter.

When you have an application which creates MIF files in quadrant 2, you can:

- Add the TRANSFORM clause to the MIF files
- Change the application so that it creates coordinates in quadrant 1
- Change the application so that it adds a TRANSFORM clause to the MIF files

### Columns

Specify the number of columns. Then, for each column, create a row containing the column name, the column type, and, for character and decimal columns, a number to indicate the width of the field. Valid column types are:

- char (width)
- integer (which is 4 bytes)
- smallint (which is 2 bytes, so it can only store numbers between -32767 and +32767)
- decimal (width,decimals)
- float
- date
- logical

This is an example of the columns section of the header:

```
COLUMNS 3  
STATE char (15)  
POPULATION integer  
AREA decimal (8,4)
```

For the database specified in this header, the MID file has three columns:

- a 15 character field that represents the STATE column,
- an integer field that represents the POPULATION column,
- an AREA column that consists of a decimal field with up to 8 total characters (digits, decimal points, and optional sign) and 4 digits after the decimal.

### MIF Data Section

The data section of the MIF file follows the header and must be introduced with DATA on a single line:

#### DATA

The data section of the MIF file can have any number of graphical primitives, one for each graphic object. MapInfo matches up entries in the MIF and MID files, associating the first object in the MIF file with the first row in the MID file, the second object in the MIF file with the second row in the MID file, and so on.

When there is no graphic object corresponding to a particular row in the MID file, a “blank” object (NONE) must be written as a place holder in the corresponding place in the MIF file.

#### NONE

The graphical objects that can be specified are:

point

line

polyline

region

arc

text

rectangle

rounded rectangle

ellipse

multipoint

collection

A **point** object takes two parameters; an X coordinate and a Y coordinate. As an option, specify the symbol that represents the point. Symbols are designated by numbers. If you omit the SYMBOL clause, the current symbol is used.

**POINT**    **x y**

      [ **SYMBOL** (*shape, color, size*)]

MapInfo 4.0 also supports two variations on the SYMBOL clause; see Symbol discussion later in this appendix.

A **line** object requires four parameters; an X and a Y coordinate for each end point. As an option, specify a pen type. When no pen type is specified, the current pen type is used.

```
LINE    x1 y1 x2 y2
        [ PEN (width, pattern, color) ]
```

A **polyline** object consists of one or more sections. If the polyline has more than one section, include the MULTIPLE keyword, followed by the number of sections. For each section, specify a numpts argument (which indicates the number of nodes in that section), followed by an x/y coordinate pair for each node. Use the optional PEN clause (described later in this appendix) to specify the line style. If you include the optional SMOOTH keyword, the polyline is smoothed.

```
PLINE  [ MULTIPLE numsections ]
        numpts1
        x1 y1
        x2 y2
        :
        [ numpts2
        x1 y1
        x2 y2      ]
        :
        [ PEN (width, pattern, color) ]
        [ SMOOTH ]
```

A **region** object consists of one or more polygons. Specify the number of polygons through the numpolygons argument (immediately after the REGION keyword). For each polygon, specify a numpts argument (which indicates the number of nodes in that polygon), followed by an x/y coordinate pair for each node. Use the optional PEN and BRUSH clauses (described later in this appendix) to specify the object's style. Use the optional CENTER clause to define the object's centroid explicitly. The centroid must be within the object.

```
REGION numpolygons
        numpts1
        x1 y1
```

```
    x2 y2
    :
[ numpts2
  x1 y1
  x2 y2 ]
:
[ PEN (width, pattern, color)]
[ BRUSH (pattern, forecolor, backcolor)]
[ CENTER x y ]
```

An **arc** requires the diagonally opposite corners of its bounding rectangle and the beginning (a) and ending (b) angles of the arc in degrees, moving counter-clockwise with zero at three o'clock. As an option, specify the pen type. (An arc specifies a section of an ellipse, the corners of which are determined by the bounding rectangle.)

```
ARC x1 y1 x2 y2
  a b
  [ PEN (width, pattern, color)]
```

A **text** object consists of a text string, up to 255 characters long. To make the text string wrap onto multiple lines, insert the characters `\n` within the textstring argument (e.g. "First line \nSecond line \nThird line"). The x1, y1, x2, and y2 arguments specify the location of the text on the map. Spacing can be 1.0 (single spacing), 1.5, or 2.0 (double spacing). Use the Font clause (described later in this appendix) to control the typeface, etc.

```
TEXT "textstring"
  x1 y1 x2 y2
  [ FONT...]
[ Spacing {1.0 | 1.5 | 2.0}]
[ Justify {Left | Center | Right}]
[ Angle text_angle]
[ Label Line {simple | arrow} x y ]
```

A **rectangle** requires the coordinates of the diagonally opposite corners. As an option, specify pen and brush types.

```
RECT  x1 y1 x2 y2
      [ PEN (width, pattern, color)]
      [ BRUSH (pattern, forecolor, backcolor)]
```

A **rounded rectangle** requires the coordinates of the diagonally opposite corners and the degree of rounding (a). As an option, specify pen and brush types. Degree of rounding is expressed in coordinate units.

```
ROUNDRECT  x1 y1 x2 y2
           a
          [ PEN (width, pattern, color)]
          [ BRUSH (pattern, forecolor, backcolor)]
```

An **ellipse** object requires the coordinates of the diagonally opposite corners of its bounding rectangle. As an option, specify pen and brush types.

```
ELLIPSE x1 y1 x2 y2
        [ PEN (width, pattern, color)]
        [ BRUSH (pattern, forecolor, backcolor)]
```

A **multipoint** object takes multiple parametrics, consisting of xy coordinate pairs. The number of points is indicated by the num\_points parameter as an option, specify the symbol that represents the multipoint. Symbols are designed by numbers. If you omit the SYMBOL clause, the current symbol is used.

```
MULTIPOINT num_points
           x1 y1  x2 y2  x3 y3 ...
```

**EXAMPLE:**

```
Multipoint 7
-3.113504 10.532464
-2.113504 11.532464
-1.113504 12.532464
-0.113504 14.532464
-4.113504 11.532464
-0.113504 8.532464
0.886496 13.532464
Symbol (35,0,12)
```

## Appendix J: MapInfo Data Interchange Format

---

```
Collection format
Collection num_parts
Region
.....
Pline
.....
Multipoint
.....
```

A **collection** object takes multiple parameters, consisting of the parameters of the object types included in the collection. Individual formats for the Region, Pline, and Multipoint parts of the collection are the same as those for the corresponding object type. The `num_parts` parameter is required if the number of parts in the collection is less than three. If this number is omitted, it is assumed that the collection contains all three parts. In exports, MapInfo always writes this number into the MIF file.

```
COLLECTION num_parts
Region
Pline
Multipoint
EXAMPLE:
```

```
Collection 3
Region 3
  5
  4.850832 10.077456
  5.850832 11.077456
  6.850832 13.077456
 12.850832 19.077456
  4.850832 10.077456
  4
-5.149168 0.077456
```

```

-4.149168 1.077456
-3.149168 3.077456
-5.149168 0.077456
  4
14.850832 20.077456
15.850832 21.077456
16.850832 23.077456
14.850832 20.077456
  Pen (1,2,0)
  Brush (2,16777215,16777215)
  Center 8.850832 14.577456
Pline 3
-7.149168 0.077456
-3.149168 -2.922544
-2.149168 2.077456
  Pen (1,2,0)
Multipoint 2
-6.149168 -0.922544
-5.149168 0.077456
  Symbol (35,0,12)

```

## Pen Styles

The Pen clause specifies the width, pattern, and color of a linear object, such as a line, polyline arc, or the border of a region. The Pen clause has the following syntax:

```
PEN (width, pattern, color)
```

Width is a number from 1 to 7. 1–7 is the width in screen pixels. 11–2047 are values that will be converted to points:

penwidth = (number of points \* 10) + 10

0 is only valid when the pen pattern is 1 for invisible lines.

Color is an integer, representing a 24-bit RGB color value.

## Appendix J: MapInfo Data Interchange Format

Pattern is an integer from 1 to 118; pattern number 1 is invisible. The pattern number corresponds to a pen number in the pen file. The pen file can be modified using a pen editor.

Valid pen numbers are from 1 to maximum number of pens in the pen file, which should not exceed 127. If a pen style is interleaved, 128 will be added to the pen number. Interleaved styles are in the range 129–255. Because the pen file can be modified, and interleaved can be specified, the pen pattern can be a number between 1–255.

The following table lists the available line styles by default:

01		31	+	61	←	91	■
02	—	32	-	62	—	92	■
03	.....	33	-	63	—	93	■
04	.....	34	-	64	—	94	▲
05	.....	35	-	65	—	95	—
06	.....	36	-	66	—	96	▲
07	.....	37	-	67	—	97	▲
08	.....	38	/	68	—	98	▲
09	.....	39	x	69	+	99	▲
10	.....	40	-	70	—	100	▲
11	.....	41	-	71	—	101	▲
12	.....	42	-	72	—	102	◆
13	.....	43	-	73	—	103	◆
14	.....	44	-	74	—	104	◆
15	.....	45	-	75	—	105	◆
16	.....	46	.....	76	—	106	◆
17	.....	47	.....	77	—	107	◆
18	.....	48	.....	78	●	108	◆
19	.....	49	.....	79	●	109	◆
20	.....	50	.....	80	●	110	~
21	.....	51	.....	81	●	111	~
22	.....	52	.....	82	●	112	~
23	.....	53	.....	83	●	113	~
24	.....	54	→	84	■	114	■
25	.....	55	←	85	■	115	■
26	+	56	↔	86	■	116	■
27	+	57	→	87	■	117	■
28	+	58	←	88	■	118	■
29	+	59	→	89	■		
30	+	60	←	90	■		



## Brush Styles




































































Brush specifies the pattern, foreground color, and background color of a filled object, such as a circle or region. The Brush clause has the following syntax:

**Brush (pattern, forecolor [, bgcolor ])**

The forecolor and bgcolor arguments are both integers, representing 24-bit RGB color values.

Pattern is a number from 1 to 71. Note: Pattern number 1 is “no fill,” and pattern number 2 is a solid fill. Pattern numbers 9–11 are reserved. The following table illustrates the available styles:

**Tip:** To specify a transparent fill style, use pattern number three or larger, and omit the bgcolor argument. For example: **Brush(5, 255)**

01		19		34		49		64	
02		20		35		50		65	
03		21		36		51		66	
04		22		37		52		67	
05		23		38		53		68	
06		24		39		54		69	
07		25		40		55		70	
08		26		41		56		71	
12		27		42		57			
13		28		43		58			
14		29		44		59			
15		30		45		60			
16		31		46		61			
17		32		47		62			
18		33		48		63			

## Symbol Styles

The Symbol clause specifies the appearance of a Point object. There are three different forms of the Symbol clause, described below.

### Symbol Clause-MapInfo 3.0 Syntax

The Symbol clause specifies the appearance of a Point object. There are three different forms of the Symbol clause. To specify a symbol style using “Old MapInfo Symbols” (the symbols that were used in earlier versions of MapInfo), use the following syntax:





































**SYMBOL (shape, color, size)**

The shape argument is an integer value, 31 or larger; 31 represents a blank symbol (i.e. the object will not be visible). The standard set of symbols includes symbols 32 through 67, inclusive, but the user can customize the symbol set by using the Symbol application.

The color argument is an integer representing a 24-bit RGB color value.

The size argument is an integer from 1 to 48, representing a point size.

The following table lists the default symbols provided with MapInfo:

31		41		51		61	
32		42		52		62	
33		43		53		63	
34		44		54		64	
35		45		55		65	
36		46		56		66	
37		47		57		67	
38		48		58			
39		49		59			
40		50		60			

### Symbol Clause – TrueType Font Syntax

To specify a symbol style based on a character from a TrueType font, use the following syntax:

```
SYMBOL (shape, color, size, fontname, fontstyle, rotation)
```

The fontname argument is a text string that identifies the name of a font (e.g. "Wingdings").

The fontstyle argument is an integer that controls settings such as Bold. The following table lists the values you can use as fontstyle.

<b>fontstyle value</b>	<b>Effect on Symbol style</b>
0	Plain text
1	Bold text
16	Black border around symbol
32	Drop shadow
256	White border around symbol

To specify two or more style attributes, add the values from the left column. For example, to specify Bold and Drop Shadow, use 33.

The rotation argument is a floating-point number, representing a rotation angle, in degrees.

### Symbol clause – Custom Bitmap File Syntax

To specify a symbol style based on a character from a TrueType font, use the following syntax:

```
SYMBOL (filename, color, size, customstyle)
```

The filename argument is a text string that identifies a bitmap file (e.g. "Arrow.BMP") in the CustSymb directory.

## Appendix J: MapInfo Data Interchange Format

The `customstyle` argument is an integer that controls whether color and background attributes are used. The following table lists the values you can use as `customstyle`:

<b>customstyle value</b>	<b>Effect on Symbol style</b>
0	Both the Show Background setting and the Apply Color setting are off; symbol appears in default state. White pixels in the bitmap appear transparent, allowing whatever is behind the symbol to show through.
1	The Show Background settings is on; white pixels in the image are opaque.
2	The Apply Color setting is on; non-white colors in the image are replaced with the Symbol's color value.
3	Both Show Background and Apply Color settings are on.

## Font Styles

The Font clause specifies the appearance (typeface, color, etc.) of text objects. The Font clause has the following syntax:

```
FONT ("fontname", style, size, forecolor [, bgcolor] )
```

Fontname in double quotation marks is the typeface to be displayed. Style is the text attribute of the typeface as shown in the following table. Size must be 0 in a MIF file, because each text object on a Map is attached to the map itself (thus the text size changes as you zoom in or out). Forecolor is an integer representing a 24-bit RGB color. The background color is optional; if you include it, MapInfo fills the area behind the text with the color you specify.

<b>style value</b>	<b>Effect on Font Appearance</b>
0	Plain
1	Bold
2	Italic
4	Underline
16	Outline (only supported on the Macintosh)
32	Shadow
256	Halo
512	All Caps
1024	Expanded

To specify two or more style attributes, add the values from the left column. For example, to specify Bold and All Caps, use 513.

### Colors

Colors are often defined in relative concentrations of red, green, and blue. Each color is a number from 0 to 255, inclusive; the RGB value of a color is calculated by the following formula:

$$(\text{red} * 65536) + (\text{green} * 256) + \text{blue}$$

These are some often used colors and their values:

Red: 16711680

Green: 65280

Blue: 255

Cyan: 65535

Magenta: 16711935

Yellow: 16776960

Black: 0

### MID File

The MID file contains data, one record of data per row, delimited by the character specified in the delimiter statement. The default delimiter is Tab. Each row in the MID file is associated with a corresponding object in the MIF file; first row with first object, second row with second object.

If delimiter character is included as part of the data in a field, enclose the field in quotation marks.

The MID file is an optional file. When there is no MID file, all fields are blank.

