

# ComGraphix Data Exchange Format (CGDEF) Reader/Writer

The ComGraphix Data Exchange Format (CGDEF) Reader and Writer modules provide the Feature Manipulation Engine (FME) with the ability to read and write MapGraphix import and export files. The CGDEF is a published ASCII format that can be used by the MapGraphix™ product for input and output.

ComGraphix Data Exchange Format Files are often called CGDEF files.

## Overview

MapGraphix is a two-dimensional (2D) system with no provision for storing user-defined attributes with the geometric data. The CGDEF reader and writer support symbols (point), lines (vector), polylines, arcs (arc, ovalarc, and polyarc), ellipses (oval, circle), polygons, and text geometric data.

Some geometric entities may have display properties such as pen and brush width (lineweight), pattern, and color.

CGDEF files are ASCII format, and use a system of keywords and values to define map parameters, overlay (layer) structure, element definitions and graphic attributes. All CGDEF map and vector geometric data is contained in a single file with the `.cgdef` extension.

FME does **not** support the import and export of TIFF files with the CGDEF. TIFF files operate on the image layer, which is something that FME does not support.

## CGDEF Quick Facts

Format Type Identifier	CGDEF
Reader/Writer	Both
Licensing Level	Base
Dependencies	None
Dataset Type	File
Feature Type	Overlay base name
Typical File Extensions	.cgdef
Automated Translation Support	Yes
User-Defined Attributes	No
Coordinate System Support	No
Generic Color Support	Yes
Spatial Index	Never
Schema Required	Yes
Transaction Support	No
Geometry Type	cgdef_type

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

## Reader Overview

The CGDEF reader first opens CGDEF file defined in the mapping file. The CGDEF reader then extracts map parameters, followed by overlay definitions, and all the features from the file. Options are provided for returning symbols as single points, or exploded into their component pieces.

## Reader Directives

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

### DATASET

**Required/Optional:** *Required*

The value for this directive is the file name of the CGDEF file to be read.

**Example:**

```
CGDEF_DATASET /usr/data/cgdef/myfile.cgdef
```

[Workbench Parameter: <WorkbenchParameter>](#)

**EXPLODE\_SYMBOLS**

**Required/Optional:** *Optional*

**Default Value:** *Yes*

The value for this directive will determine the reader's action when it comes across a symbol in the file. If the value is YES, then the reader, rather than outputting a symbol feature, will look at the symbol definition for that symbol and output it as a series of individual features that make up the symbol. If the value is NO, then a symbol in the source file is sent as a `cgdef_symbol` type rather than attempting to send its actual features individually. The default value is YES, since most formats cannot properly interpret a symbol type as anything more than a point feature – thus, by exploding the symbol, it can be seen.

**Example:**

```
EXPLODE_SYMBOLS yes
```

[Workbench Parameter: <WorkbenchParameter>](#)

## Writer Overview

The CGDEF writer creates and writes feature data to the CGDEF file specified by the DATASET keyword. The directory where the file is created must exist before the translation occurs, and if there is an old CGDEF file with the same filename in the directory, it will be overwritten with the new feature data. Before actually writing out features, the writer first scans the prototype/template file defined in the mapping file (see *Writer Keywords*) to extract all the header information required by CGDEF.

## Writer Directives

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

**DATASET**

**Required/Optional:** *Required*

The value for this directive is the file name into which data is to be written.

**Example:**

```
CGDEF_DATASET /usr/data/cgdef/myfile.cgdef
```

[Workbench Parameter: <WorkbenchParameter>](#)

## PROTOTYPE\_FILE

**Required/Optional:** *Optional*

The value following this directive is the file name of the CGDEF file that is used as a template file. This file should include all setup and header information along with RGB color definitions (required), symbol definitions (required) and overlay definitions (optional). The writer will collect this information and use it as it processes features to output.

### Example:

```
CGDEF_PROTOTYPE_FILE /usr/data/cgdef/template.cgdef
```

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

## 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.

CGDEF features consist of geometry but no user-defined attributes, although there are special attributes to hold the type of the geometric entity and its display parameters.

The FME considers the CGDEF overlay name to be the FME feature type of a CGDEF feature. When writing, the CGDEF writer will create a new overlay for each unique feature type that is passed to the writer. All CGDEF features contain a `cgdef_type` attribute, which identifies the geometric type. Each geometric/element type can also have an id, up to 31 characters long, associated with it. Every element type except symbols will have associated colors attached to it. Both `cgdef_symbol_name` and `cgdef_symbol_sequence_number` are fields that are only filled if the element is part of a symbol instance. Depending on the geometric type, the feature contains additional attributes specific to the geometric type. These are described in subsequent sections.

Attribute Name	Contents
<code>cgdef_type</code>	The CGDEF geometric type of this entity. <b>Range:</b> <code>cgdef_symbol </code> <code>cgdef_polyline </code> <code>cgdef_polygon </code> <code>cgdef_text </code> <code>cgdef_ellipse </code> <code>cgdef_arc</code> <b>Default:</b> No default
<code>cgdef_element_id</code>	The CGDEF ID for this entity, this is an optional attribute <b>Range:</b> String <b>Default:</b> No default
<code>cgdef_color.red</code>	The element's red color intensity, as determined by looking up the element's color index in the color table. <b>Range:</b> 0..65535 <b>Default:</b> 27000 (when writing only)

<b>Attribute Name</b>	<b>Contents</b>
cgdef_color.green	The element's green color intensity, as determined by looking up the element's color index in the color table. <b>Range:</b> 0..65535 <b>Default:</b> 30000 (when writing only)
cgdef_color.blue	The element's blue color intensity, as determined by looking up the element's color index in the color table. <b>Range:</b> 0..65535 <b>Default:</b> 38000 (when writing only)
fme_color	This is a string that represents the color intensities of the element. It is formatted as red, green, blue intensities which range between 0..1 This 0..1 value is arrived at by taking the color intensity and dividing it by the total intensity range, in this case, 65535 <b>Range:</b> String. (0..1, 0..1, 0..1) <b>Default:</b> 27000/65535, 30000/65535, 38000/65535(when writing only)
cgdef_symbol_name	If the element is part of a symbol and the symbol has been exploded into its individual elements, then this field contains the symbol name <b>Range:</b> String <b>Default:</b> None
cgdef_symbol_sequence_number	If the element is part of a symbol and the symbol has been exploded into its individual elements, then this field contains the a unique number which identifies itself and the other elements in the symbol <b>Range:</b> String <b>Default:</b> None

## Symbols

**cgdef\_type:** cgdef\_symbol

CGDEF symbol features specify a single x and y coordinate. This coordinate defines the center of the symbol The symbol is defined by a symbol number, and a scale attribute. If no scale is defined, then the symbol will be placed at the current default symbol scale setting.

The table below lists the special FME attribute names used to control the CGDEF symbol settings.

<b>Attribute Name</b>	<b>Contents</b>
cgdef_symbol_number	This number references a resource in the map. If the symbol number does not have a resource in the map, the default symbol rectangle is placed at the specified location. <b>Range:</b> Any integer number > 0 <b>Default:</b> No default
cgdef_symbol_scale	The scale at which the symbol is to be placed. <b>Range:</b> 1..20 <b>Default:</b> 1

# Symbol and Group Definitions

## Symbol Definitions

Symbols are defined as a set of feature/element types. The collection of feature types becomes the symbol. As an example, a symbol can be defined as two circles and an arc, which together form a happy face. Thus, a `cgdef_symbol` type actually references its definition through the `cgdef_symbol_number` and places that symbol with appropriate scaling at the coordinates specified in the `cgdef_symbol` type.

## Group Definitions

Along the same lines is a group definition. A group is another feature which is made up of a set of other element types. However, a group does not have a group number or group name to identify it (although it may still have an ID which any feature may possess).

FME **does not** recognize groups; instead, it outputs the elements of the group as independent features.

## Text

**cgdef\_type:** `cgdef_text`

CGDEF text is used for text annotation in CGDEF. The coordinates specify the lower left coordinates of the text when it is placed. In addition, the size and angle that the text is output can be specified.

The table below lists the special FME attribute names used to control the CGDEF text:

Attribute Name	Contents
<code>cgdef_text_size</code>	The size of the text specified in ground units of the map. <b>Range:</b> float > 0 <b>Default:</b> 0
<code>cgdef_text_angle</code>	The text angle is given in degrees and measured from the horizontal. <b>Range:</b> -360..360 <b>Default:</b> 0
<code>cgdef_text_font</code>	The type of font. <b>Range:</b> String <b>Default:</b> No default
<code>cgdef_text_style</code>	The display style for the text. <b>Range:</b> String <b>Default:</b> No default
<code>cgdef_text_string</code>	The text to be displayed <b>Range:</b> String <b>Default:</b> No default

## Polylines

**cgdef\_type:** `cgdef_polyline`

CGDEF polyline features specify linear features defined by a sequence of x and y coordinates. Polylines encapsulate the concept of a line since a line is just a sequence of two points. Furthermore, the polyline type will be used with the `cgdef_arc` type to handle poly arcs used in MapGraphix. Poly arcs will be represented by a sequence of polylines and arcs.

Each polyline has a pen style associated with it specifying the color, line weight, and line type used when the line is drawn. If no pen style is defined for a polyline entity, the previous style is used.

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

Attribute Name	Contents
<code>cgdef_pen_lineweight</code>	Defines the lineweight used to draw the polyline. This is measured in screen pixels. <b>Range:</b> 1..127 <b>Default:</b> 1
<code>cgdef_pen_linetype</code>	The linetype used to draw the line. <b>Range:</b> 1..19 <b>Default:</b> 1

## Polygons

**cgdef\_type:** `cgdef_polygon`

CGDEF polygon 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 polygon has a pen style associated with it to control the color, line weight, line type, and brush pattern used when it's drawn. If no pen style is defined for a polygon entity, the previous style is used.

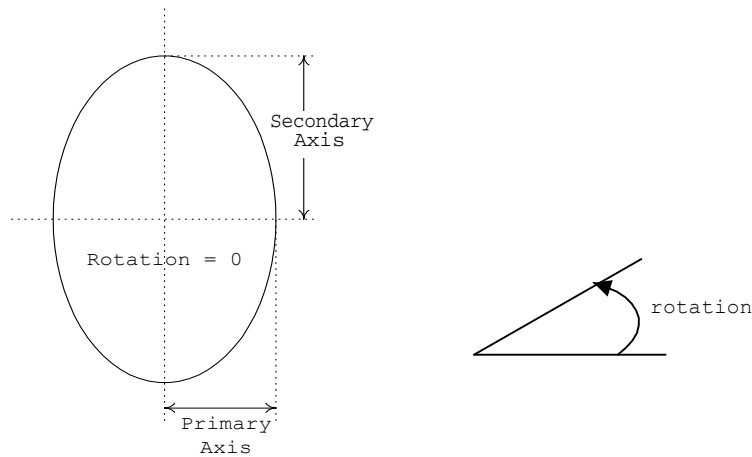
The following table lists the special FME attribute names used to control the CGDEF polygon settings.

Attribute Name	Contents
<code>cgdef_pen_lineweight</code>	Defines the lineweight used to draw the polyline. This is measured in screen pixels <b>Range:</b> 1...127 <b>Default:</b> 1
<code>cgdef_pen_linetype</code>	The linetype used to draw the line. <b>Range:</b> 1...19 <b>Default:</b> 1
<code>cgdef_brush_pattern</code>	The pattern used to draw the line. <b>Range:</b> 1...41 <b>Default:</b> 1

## Ellipse

**cgdef\_type:** `cgdef_ellipse`

The `cgdef_ellipse` corresponds to ovals in MapGraphix. Ellipse features are point features, and have only a single coordinate. This point serves as the center of the ellipse. Additional attributes specify the primary axis (X) and secondary axis (Y) of the ellipse. CGDEF ellipses also support rotation.




---

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

---

CGDEF ellipses can also arrive at circles, since circles are just ellipses with equal primary axis and the secondary axis.

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

Attribute Name	Contents
<code>cgdef_primary_axis</code>	The length of the semi-major axis in ground units. (x-axis) <b>Range:</b> Any real number > 0 <b>Default:</b> No default
<code>cgdef_secondary_axis</code>	The length of the semi-minor axis in ground units. (y-axis) <b>Range:</b> Any real number > 0 <b>Default:</b> No default
<code>cgdef_rotation</code>	The rotation of the major axis. The rotation is measured in degrees counterclockwise up from horizontal. <b>Range:</b> -360.0..360.0 <b>Default:</b> 0

## Arc

**cgdef\_type:** `cgdef_arc`

The arc definition here handles arcs, oval arcs and parts of poly arcs used in MapGraphix. Poly arcs use the `cgdef_arc` type as well as `cgdef_polyline` types to form the original poly arc defined in the CGDEF file.

## ComGraphix Data Exchange Format (CGDEF) Reader/Writer

CGDEF arc features are linear features used to specify elliptical arcs. As such, the feature definition for `cgdef_arc` is similar to the ellipse definition, with two additional angles to control the portion of the ellipse boundary drawn. CGDEF arcs also support rotation.

---

**Tip:** The function `@Arc()` can be used to convert an arc to a linestring. This is useful for storing Arcs in systems that don't support them directly.

---

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

---

Attribute Name	Contents
<code>cgdef_primary_axis</code>	The length of the semi-major axis in ground units. (x-axis) <b>Range:</b> Any real number > 0 <b>Default:</b> No default
<code>cgdef_secondary_axis</code>	The length of the semi-minor axis in ground units. (y-axis) <b>Range:</b> Any real number > 0 <b>Default:</b> No default
<code>cgdef_start_angle</code>	Refer to the <code>@Arc</code> (function) in the <i>FME Functions and Factories manual</i> for a detailed definition of <code>start_angle</code> . <b>Range:</b> 0.0..360.0 <b>Default:</b> 0
<code>cgdef_sweep_angle</code>	Refer to the <code>@Arc</code> (function) in the <i>FME Functions and Factories manual</i> for a detailed definition of <code>sweep_angle</code> . <b>Range:</b> 0.0..360.0 <b>Default:</b> No default
<code>cgdef_rotation</code>	The rotation of the major axis. The rotation is measured in degrees counterclockwise up from horizontal. <b>Range:</b> -360.0..360.0 <b>Default:</b> 0

---

