

Scalable Vector Graphics (SVG) Writer

The Scalable Vector Graphics (SVG) Writer enables the Feature Manipulation Engine (FME) to write documents that conform to the World Wide Web Consortium's (W3C) SVG 1.1 specification. This chapter assumes familiarity with the specification.

Overview

FME's SVG output is optimal for scripting and spatial information display. Specific features include:

- coordinate preservation;
- layered spatial geometry with adjustable paint order;
- template processing for incorporation of predefined scripts, style sheets, images and other SVG entities;
- single SVG element output for each FME feature (including features with donut and aggregate geometry)
- international character support.
- gzip compression support.

SVG Quick Facts

Format Type Identifier	SVG
Reader/Writer	Writer
Licensing Level	Professional
Dependencies	None
Dataset Type	File
Feature Type	group ID attribute name
Typical File Extensions	.svg, .svgz
Automated Translation Support	Yes
User-Defined Attributes	Yes
Coordinate System Support	No
Generic Color Support	Yes
Spatial Index	Not applicable
Schema Required	Optional
Transaction Support	No
Geometry Type	svg_type

Geometry Support			
Geometry	Supported?	Geometry	Supported?
aggregate	yes	point	yes
circles	yes	polygon	yes

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

Writer Overview

The SVG writer converts a set of FME features into geometric SVG elements. These elements are output in a document with three sections: the template section, the layer section and the geometric section.

- The template section is composed of a single non-extended SVG document.
- The layer section is embedded in the template section, and is composed of zero to more SVG group elements
- The geometric section is embedded in the layer section, and is composed of zero to more geometric SVG elements.

In the layer section, one SVG group element is produced for each unique feature type in a writer's feature set. The `id` attribute on this group element is set equal to the feature type name. The layer is inserted immediately before the template document's closing root element tag.

The document's geometric section contains a single SVG element for each FME feature that is sent to the writer. These elements are grouped according to their feature type and embedded under the group element with a matching ID in the layer section.

The SVG writer does not check FME attribute name characters that are invalid XML attribute name characters. FME attribute names are transcoded from the operating system's local code page to UTF-8 and written directly to the SVG document; the writer does not escape any attribute name characters into character entities. It is the responsibility of the user to ensure the FME attribute names are valid XML attribute names.

Writer Directives

The directives listed below are processed by the SVG writer. The suffixes shown are prefixed by the current `<WriterKeyword>` in a mapping file. By default, the `<WriterKeyword>` for the SVG writer is `SVG`.

DATASET

Required/Optional: *Required*

The value for this keyword is the pathname for the output SVG file. If a file with this pathname already exists, then it will be overwritten. A typical mapping file fragment specifying an output SVG dataset looks like:

```
SVG_DATASET /tmp/outputFile.svg
```

If the output filename's extension is `svgz` then the output document will be compressed using gzip compression.

Note that all SVG documents are written using UTF-8 encoding.

Workbench Parameter: [Destination SVG File](#)

TEMPLATE

Required/Optional: *Optional*

This optional parameter directs the writer to the location of the SVG document to use as the outline of the output dataset.

The syntax for this keyword's value is:

```
<WriterKeyword>_TEMPLATE <value>
```

(where `<value>` is location of the template path)

If this keyword is not provided in the mapping file, then the file named `defaultTemplate.svg` under the `svg` directory in the FME home directory is used.

The template document has several uses including: the insertion of predefined geometric elements, the inclusion of Cascading Style Sheets, and the embedding of scripting information. There are a few issues that must be considered to ensure proper template processing. The template must conform to the non-extended SVG language defined by the SVG 1.1 specification. The encoding of the template must be one of the following: ASCII, UTF-8, UTF-16, UCS4, ISO-8859-1 or Windows-1252. Note that the encoding of the output SVG document is always UTF-8. Any document type declaration provided in the template will be overridden in the output document.

Two placeholder macros have been defined for use in the SVG template in order to retrieve information specified in other keywords:

- `$(FME_SVG_ATTR_NS_PREFIX)` will be replaced with the value of the `ATTR_NAMESPACE_PREFIX` keyword, and
- `$(FME_SVG_ATTR_NS_URI)` will be replaced with the value of the `ATTR_NAMESPACE_URI` keyword.

These macros will only work inside CDATA sections (`<![CDATA[...]]>`) of the SVG template. If they are found outside a CDATA section, they will remain unchanged.

Workbench Parameter: [Template File](#)

COORDINATE PRECISION

Required/Optional: *Optional*

This optional parameter specifies the number of decimal digits to use when writing an SVG element coordinate's value. The default is 6. Specifying a larger value increases coordinate precision and may increase rendering precision.

Workbench Parameter: [Precision](#)

NORMALIZE

Required/Optional: *Optional*

This optional parameter will normalize the lower coordinate bounds of the writer's feature set to (0,0). Normalization can reduce rendering inaccuracies by SVG viewers with small coordinate precision capability. A normalized document's file size is typically smaller than a non-normalized version.

Workbench Parameter: [Normalize](#)

DEF

Required/Optional: *Optional*

The syntax for DEF is:

```
<WriterKeyword>_DEF <FeatureType>
  SVG_PAINT_ORDER [0-9]+
  SVG_LAYER_STYLE string
  <UserAttributeName0> char([0-9]+)
  ...
  <UserAttributeNameN> char([0-9]+)
```

The `SVG_PAINT_ORDER` parameter on a DEF line is used to determine the order of feature output. Features in layers that have a higher value for this parameter will be output last. Following SVG's "painter" algorithm, features that are in layers with higher values will be painted last when the SVG document is rendered.

The `SVG_LAYER_STYLE` parameter on a DEF line is used to specify the value to set the layer's `STYLE` attribute in the output layer group.

The user attribute keywords specify which FME attributes to extract from an incoming FME feature. The extracted FME attributes are embedded in the geometric element's attribute list.

If there are no user attribute DEF line parameters specified, then no FME user attributes will be inserted in any SVG element's attribute list, and no SVG DTD extension is produced.

ABSOLUTE_COORDINATES

Required/Optional: *Optional*

Allows absolute instead of relative coordinates to be used for lines and polygons that are written out as `<path>` elements. The valid values for this keyword are `Yes` and `No`; its default value is `No`.

Workbench Parameter: [Use absolute Coordinate](#)

WHITE_STROKES_TO_BLACK

Required/Optional: *Optional*

Determines whether the SVG writer should automatically switch white `fme_color` specifications into black. This directive does not affect the `svg_color` (that is, the

`svg_color` attribute takes precedence over the `fme_color`). The valid values for this keyword are `Yes` and `No`; its default value is `Yes`.

Workbench Parameter: [Automatically turn white strokes into black](#)

DOCTYPE_EXTERNAL

Required/Optional: *Optional*

Determines if the SVG file depends on an external SVG DTD. The valid values for this keyword are `Yes` (default value) and `No`. When set to `Yes` the document type declaration's public and system identifier for SVG 1.1 are used by default, but these default identifiers can also be overwritten with the `DOCTYPE_PUBLIC_ID` and `DOCTYPE_SYSTEM_ID` keywords.

Workbench Parameter: [Reference external SVG DTD](#)

DOCTYPE_PUBLIC_ID

Required/Optional: *Optional*

This keyword only applies when the `DOCTYPE_EXTERNAL` keyword is set to `Yes`. It specifies the public identifier for the document type declaration. This keyword must be used in conjunction with the `DOCTYPE_SYSTEM_ID` keyword, that is, a system identifier must also be simultaneously specified; otherwise, this keyword has no effect.

Workbench Parameter: [DOCTYPE public identifier](#)

DOCTYPE_SYSTEM_ID

Required/Optional: *Optional*

This keyword only applies when the `DOCTYPE_EXTERNAL` keyword is set to `Yes`. It specifies the system identifier for the document type declaration. This keyword can be used alone or in conjunction with the `DOCTYPE_SYSTEM_ID` keyword.

Workbench Parameter: [DOCTYPE system identifier](#)

ATTR_NAMESPACE_PREFIX

Required/Optional: *Optional*

This directive specifies the prefix which will be used to identify the namespace of all user attributes in the SVG document.

Each user attribute written to the SVG file will be written as an XML attribute with the format `namespace_prefix:user_attr = "value"`.

The default namespace prefix is `"fme"`.

Workbench Parameter: [Attributes Namespace Prefix](#)

ATTR_NAMESPACE_URI

Required/Optional: *Optional*

This directive specifies the URI with which the namespace prefix (specified by `ATTR_NAMESPACE_PREFIX`) will be associated. This will be the namespace URI for all

user attributes in the SVG document. (Note that the writer does not check if this is a valid URI that complies with XML standards.)

The namespace will be defined as follows, where `namespace_prefix` is the value defined by `ATTR_NAMESPACE_PREFIX` and `namespace_uri` is defined by `xmlns:namespace_prefix="namespace_uri"`

The default value is `"http://www.safe.com/fme"`

[Workbench Parameter: Attributes Namespace URI](#)

VIEWBOX_MINX, VIEWBOX_MINY, VIEWBOX_WIDTH, VIEWBOX_HEIGHT

Required/Optional: *Optional*

These directives allow the user to set viewbox size when writing to SVG format. By default, the viewbox is not available. Only when the user has set all four parameters (listed below), the viewbox will be displayed.

Syntax for the values is:

```
<WriterKeyword>_VIEWBOX_MINX<value>
<WriterKeyword>_VIEWBOX_MINY<value>
<WriterKeyword>_VIEWBOX_WIDTH<value>
<WriterKeyword>_VIEWBOX_HEIGHT<value>
```

All values must be specified in decimal, integer or scientific notation.

[Workbench Parameter: Viewbox - Min x](#)

[Workbench Parameter: Viewbox - Min y](#)

[Workbench Parameter: Viewbox - Width](#)

[Workbench Parameter: Viewbox - Height](#)

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.

Any feature that is sent to the SVG writer has several attributes that the writer uses to determine that feature's geometric representation in SVG. The start of this processing occurs when the writer examines the feature's `svg_type` attribute. Once the writer determines this attribute's value, it can process the other attribute information required to complete the geometric conversion process. If, for example, the feature has an `svg_type` of `svg_arc`, then the writer will retrieve the `svg_primary_axis`, `svg_secondary_axis`, `svg_start_angle` and `svg_rotation` attributes to determine what geometric attribute values will exist in the feature's path element representation.

All format-specific attributes begin with `svg_` and are predefined in this section's tables. In addition to format attributes, the SVG writer can process user-defined attributes. The writer extracts these user attributes from the incoming features and inserts them the output element's attribute list. To determine which attributes to extract, the writer examines its mapping file's `DEF` lines. The feature's feature type must match the `DEF` lines type for this extraction to occur. The user-defined attributes are defined in the SVG element's attribute list under the qualified name prefix `fme`. An ex-

tension to the SVG DTD is produced to accommodate these user-defined attributes. This extension is defined in the document's internal document type declaration.

When producing an element's attribute list, the writer will examine the contents of the attribute values to determine if there are any < or " characters. If these values are present, they are output using < and " respectively. Attribute values are embedded in an element's attribute list using the quote (") delimiter.

The following table lists the format attributes that are common to all features sent to the SVG writer. Other than `svg_color` and `svg_fill_color`, all attributes in this table have a direct mapping to the attribute names that can be set on individual SVG elements. Selected SVG elements attribute names have been prepended with a `svg_` string to produce the FME attribute names.

Attribute Name	Contents
<code>svg_color</code>	<p>The color used to stroke an element. The string is formatted with three comma-separated values representing the ordered intensities red, green, and blue. The individual intensity values are character decimal character strings that can range in value from 0 to 1 with 1 being the highest. See note on color below.</p> <p>Range: string 0.0..1.0, 0.0..1.0, 0.0..1.0</p> <p>Default: None</p>
<code>svg_fill_color</code>	<p>The color used to fill an element. The string is formatted with three comma-separated values representing the ordered intensities red, green, and blue. The individual intensity values represent decimals that can range in value from 0 to 1 with 1 being the highest. This value is not applicable to <code>svg_line</code> or <code>svg_arc</code> features. See note on color below.</p> <p>Range: string 0.0..1.0, 0.0..1.0, 0.0..1.0</p> <p>Default: None</p>
<code>svg_id</code>	<p>An element's unique identifier. Directly maps to an element's <code>id</code> attribute. Refer to the XML 1.0 specification for applicable values. It is strongly recommended that users not create IDs that begin with "FME_".</p> <p>Range: string</p> <p>Default: None</p>
<code>svg_class</code>	<p>Assigns a class name or set of class names to an element. Directly maps to an element's <code>class</code> attribute. Refer to the SVG 1.1 specification for applicable values.</p> <p>Range: string</p> <p>Default: None</p>
<code>svg_style</code>	<p>Specifies style information for an element. Directly maps to an element's <code>style</code> attribute. Refer to the SVG 1.1 specification for applicable values.</p> <p>Range: string</p> <p>Default: None</p>

Attribute Name	Contents
svg_onfocusin	<p>Identifies the script method to call when an element receives focus. Directly maps to an element's <code>onfocusin</code> attribute.</p> <p>Range: string (must match an available script method ID) Default: None</p>
svg_onfocusout	<p>Identifies the script method to call when an element loses focus. Directly maps to an element's <code>onfocusout</code> attribute.</p> <p>Range: string (must match an available script method ID) Default: None</p>
svg_onclick	<p>Identifies the script method to call when a pointing device button is clicked over an element. Directly maps to an element's <code>onclick</code> attribute.</p> <p>Range: string (must match an available script method ID) Default: None</p>
svg_onmousedown	<p>Identifies the script method to call when a pointing device button is pressed over an element. Directly maps to an element's <code>onmousedown</code> attribute.</p> <p>Range: string (must match an available script method ID) Default: None</p>
svg_onmouseup	<p>Identifies the script method to call when a pointing device button is release over an element. Directly maps to an element's <code>onmouseup</code> attribute.</p> <p>Range: string (must match an available script method ID) Default: None</p>
svg_onmouseover	<p>Identifies the script method to call when a pointing device button is moved on to an element. Directly maps to an element's <code>onmouseover</code> attribute.</p> <p>Range: string (must match an available script method ID) Default: None</p>
svg_onmousemove	<p>Identifies the script method to call when a pointing device button is moved while it is over an element. Directly maps to an element's <code>onmousemove</code> attribute.</p> <p>Range: string (must match an available script method ID) Default: None</p>
svg_onmouseout	<p>Identifies the script method to call when a pointing device button is moved away from an element. Directly maps to an element's <code>onmouseout</code> attribute.</p> <p>Range: string (must match an available script method ID) Default: None</p>

Note: The attributes `fme_color` and `svg_color` can both be used to set the value on an element's `stroke` attribute. `fme_color` and `svg_color` are translated to SVG's RGB function syntax. For the case where more than one color attribute is specified on a feature, order of precedence is `svg_color`, and then `fme_color`.

The same processing occurs for the attribute `fme_fill_color`, `svg_fill_color`, except both values can be used to set the element's 'fill' attribute.

In addition, if the values for the `svg_color` or `svg_fill_color` do not match the FME color specification, i.e., "r,g,b" where r,g,b are in [0..1], then the writer will plainly transfer the value specified into the SVG `stroke` and `fill` attributes respectively. This is useful if the user needs to by pass the FME "r,g,b" syntax, for example, to use the SVG's predefined color names, "red", "black", etc..., or if the user wants to use gradient fill. A user-defined SVG template (see the writer's `TEMPLATE` keyword) could define several gradients to be referenced by the FME feature's `svg_fill_color` attribute.

Consider the following `TEMPLATE` for the SVG writer. It defines the `MyGradient` `linearGradient` that can be referenced by FME features by setting their `svg_fill_color` attribute to the value, `url(#MyGradient)`:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg xmlns="http://www.w3.org/2000/svg">
  <defs>
    <linearGradient id="MyGradient">
      <stop offset="5%" stop-color="#F60" />
      <stop offset="95%" stop-color="#FF6" />
    </linearGradient>
  </defs>
</svg>
```

Points

svg_type: `svg_point`

Point features can have one or more coordinates. All point features are usually output as SVG path elements. Single, non-aggregate, point features may also be output by an SVG use element that references another element whose graphical content is to be drawn at the position of the single point:

Attribute Name	Contents
<code>svg_use</code>	<p>This format attribute only applies to single point features. If it is present and non-empty in a single point feature then the point feature is written out with a SVG use rather than a SVG path element. The value of the <code>svg_use</code> attribute must be a valid href location because this value is directly copied into the use element's <code>xlink:href</code> attribute. The single coordinate of the point is also transferred onto the use element's <code>x</code> and <code>y</code> attributes. The element referenced by the the SVG use element may be predefined in an FME SVG writer template file, see the writer's <code>TEMPLATE</code> keyword.</p> <p>Range: string Default: Empty String</p>

Lines

svg_type: `svg_line`

Polyline features must have at least two coordinates. Line features are output as SVG path elements

Polygons

svg_type: `svg_polygon`

Polygon features must have at least two coordinates. Polygon feature are output as SVG path elements, and are automatically closed if the first and last coordinate of a polygon segment do not match.

Text

svg_type: `svg_text`

Text features must have exactly one coordinate. Text features are output as SVG text elements, and have the following additional attributes:

Attribute Name	Contents
<code>svg_text_string</code>	The text string may contain blanks and there is no limit on its length. This attribute must be present for all <code>svg_text</code> features. Range: <code>string</code> Default: Empty String
<code>svg_text_size</code>	The size of the text in ground units. Range: Any real number > 0 Default: 0
<code>svg_rotation</code>	The rotation of the text, as measured in degrees counter-clockwise from the horizontal. Range: -360.0...360.0 Default: 0

Ellipse

svg_type: `svg_ellipse`

Ellipse features must have exactly one coordinate. Ellipse features are output as SVG ellipse elements and have the following additional attributes:

Attribute Name	Contents
<code>svg_primary_axis</code>	The length of the semi-major axis in ground units. Range: Any real number > 0 Default: 0
<code>svg_secondary_axis</code>	The length of the semi-minor axis in ground units. Range: Any real number > 0 Default: 0

Attribute Name	Contents
svg_rotation	The rotation of the ellipse, as measured in degrees counterclockwise from the horizontal. Range: -360.0...360.0 Default: 0

Arc

svg_type: svg_arc

Arc features must have exactly one coordinate. Arc features are output as SVG path elements, and have the following additional attributes:

Attribute Name	Contents
svg_primary_axis	The length of the semi-major axis in ground units. Range: Any real number > 0 Default: 0
svg_secondary_axis	The length of the semi-minor axis in ground units. Range: Any real number > 0 Default: 0
svg_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
svg_rotation	The rotation of the major axis. The rotation is measured in degrees counterclockwise up from the horizon. Range: -360.0...360.0 Default: 0

Rectangle

svg_type: svg_rectangle

The extents of this feature are calculated using its bounding box. Rectangle features are output as SVG rect elements.

Rounded Rectangle

svg_type: svg_rectangle

The extents of this feature are calculated using its bounding box. Rounded Rectangle features are output as SVG rect elements.

Attribute Name	Contents
svg_rounding	Contains the diameter, in ground units, of the circle used to produce the rounded corners. Range: Any real number > 0 Default: 0

