

# Intergraph FRAMME Standard Exchange Format (SEF) Writer

## Overview

The Standard Exchange Format (SEF) writer modules provide the Feature Manipulation Engine (FME) with the ability to write SEF data files. SEF files use a published ASCII format. The SEF data file structure is described in the *FRAMME Loader Reference Guide*. This section assumes an operational knowledge of SEF.

## SEF Quick Facts

Format Type Identifier	SEF
Reader/Writer	Writer
Licensing Level	Professional
Dependencies	Requires an extra-cost plug-in from Safe Software
Dataset Type	File
Feature Type	N/A
Typical File Extensions	.sef
Automated Translation Support	Yes
User-Defined Attributes	Yes
Coordinate System Support	No
Generic Color Support	No
Spatial Index	Not applicable
Schema Required	Yes
Transaction Support	No
Geometry Type	sef_type

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

## Writer Overview

SEF features can be associated with very complex feature geometries. A single SEF feature can have multiple geometries of different geometric types. Here we refer to each geometric type in a SEF feature as a “component”. In order to represent this in the FME, it is necessary to use aggregates of geometry. This is further explained in *Geometry Representation* on page 1187.

The SEF writer deals with three kinds of features: workset info features, reference features and regular features.

- Workset info features provide the SEF writer with the necessary information to create a workset when the output file is exported to FRAMME.
- Reference features are referenced multiple times in a workset.
- All other features are considered regular features.

It is also worth noting that the SEF writer does not handle donuts or cells.

The structure of a SEF Component are explained in *SEF Components* on page 1174.

---

**Note:** There are still some open issues regarding the SEF writer. These issues are being investigated. Currently, the range of the attributes are not strictly enforced. Many of them are still being investigated.

---

## Writer Directives

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

### DATASET

**Required/Optional:** *Required*

A SEF file consists of an ASCII output file, with a `.sef` extension. The value for this keyword is the SEF file to be written. A typical mapping file fragment specifying an output SEF dataset looks like:

```
SEF_DATASET /usr/data/Sef/output.sef
```

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

### DEF

**Required/Optional:** *Required*

Each SEF Feature type must be defined before it can be written. The definition specifies the base name of the file, and the names and the types of all attributes. Note that only user-defined attributes are required. The syntax of a SEF `DEF` line is:

```
<WriterKeyword>_DEF <baseName> \
    [<attrName> <attrType>]+ \
    [<componentName>:<attrName> <attrType>]+
```

The following table shows the attribute types supported.

Field Type	Description
char(<width>)	Character fields store fixed length strings. The <code>width</code> 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 date as character string with the format YYYYMMDD.
number(<width>, <decimals>)	Number fields store single and double precision floating point values. The <code>width</code> parameter is the total number of characters allocated to the field, including the decimal point. The <code>decimals</code> parameter controls the precision of the data and is the number of digits to the right of the decimal.
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.
smallint	Small integer fields store 16-bit signed integers and therefore have a range of -32767 to +32767.
logical	Logical fields store TRUE/FALSE data. Data read or written from and to such fields must always have a value of either <code>true</code> or <code>false</code> .

Here's an example feature with two nongraphic components:

`component{0}.streetName` is an integer while `component{1}.streetName` is a string; therefore, on our DEF line, we need to define `streetName` separately in the `<componentName>:<attrName> <attrType>` format. However, for all other attributes with no such type conflict, the `<attrName> <attrType>` format is used.

```

FACTORY_DEF * CreationFactory                                \
  CREATE_AT_END                                           \
  NUMBER_TO_CREATE 1                                     \
  OUTPUT FEATURE_TYPE feat                               \
    sef_component{0}.sef_type sef_nongraphic             \
    sef_component{0}.sef_lbl street1                     \
    sef_component{0}.streetName 6                       \
    sef_component{0}.sign stop                           \
    sef_component{1}.sef_type sef_nongraphic             \
    sef_component{1}.sef_lbl street2                     \
    sef_component{1}.streetName"Dow St" \
    sef_component{1}.sign "no right turn"

```

The output DEF line is:

```

SEF_DEF feat                                             \
street1:streetName integer                               \
street2:streetName char(50)                             \
sign char(50)

```

## SEF Components

A SEF component is the foundation for many of the SEF features. In order to make SEF reference features or other regular features, one must fully comprehend the structure of the SEF components.

Each SEF component has a name and a type in addition to its regular attributes and its geometry attributes. *Regular* attributes are those attributes which are not related to coordinates, and are stored as attributes on the FME feature. *Geometry* attributes are coordinate-specific attributes, and are stored as coordinates in the component's FME geometry. For more detail on Geometry attributes, please refer to *Geometry Representation* on page 1187. Since there can be more than one component in a feature, each component's attributes are prefixed by the keyword `sef_component{<number>}`. In this manner,

```
sef_component{0}.sef_lbl label0           \  
sef_component{0}.sef_type sef_arc        \  
sef_component{0}.sef_usrAttribute material \  
sef_component{0}.sef_prim_axis 10        \  
sef_component{1}.sef_lbl label1         \  
sef_component{1}.sef_type type1
```

where `<number>` is an integer that starts from 0.

If only one graphical component exists in any SEF Feature, the prefix `sef_component{}` can be omitted. For example, if we have a feature with a nongraphic component and a two-point line, then the `sef_component{}` list is optional. The SEF writer will write out all the user-defined attributes of the feature in a nongraphic component section preceding the graphical component section. However, if there is a preferred order in which the components are arranged, the user must do so via the `sef_component{}` mechanism.

The attributes below can be attached to all component types:

Keyword Suffix	Value	Required/Optional
<code>sef_type</code>	A text string specifying a supported SEF type. <b>Range:</b> sef_arc   sef_cell   sef_shape   sef_line   sef_curve   sef_symbol   sef_text   sef_fixed_text_node   sef_display_text_node   sef_complement_text_node   sef_repeat_text_node   sef_nongraphic <b>Default:</b> No default	Required

<b>Keyword Suffix</b>	<b>Value</b>	<b>Required/Optional</b>
sef_lbl	A text string with a max of 31 characters specifying the label of the component. This defaults to an empty string.	Optional
sef_out_of_scope	Used this flag only if the component continues beyond the area defined by the index shapes in a SEF file. <b>Range:</b> begin   end <b>Default:</b> No default	Optional
sef_level	The level on which the component is located. <b>Range:</b> An integer from 1 to 63 <b>Default:</b> No Default	Optional
sef_style	The style of the component. <b>Range:</b> An integer from 0 to 7 <b>Default:</b> No Default	Optional
sef_color	The color of the component represented by an integer. <b>Range:</b> An integer from 0 to 255 <b>Default:</b> No Default	Optional
sef_weight	The thickness of the component. <b>Range:</b> An integer from 0..31 <b>Default:</b> No Default	Optional
<user defined>	This is a user-defined text string.	Optional

Each component may have additional attributes, depending on the `sef_type` of the component.

### **sef\_type: sef\_arc**

This is the arc component. The SEF writer writes all arcs as a center point (origin) and regular attributes to describe the shape of the arc.

The SEF writer expects an `origin`, which specifies the coordinate of the origin for the arc on the FME geometry. Each `sef_arc` given to the writer must have exactly one point within the aggregate.

The specific attributes for `sef_arc` are shown below:

<b>Keyword Suffix</b>	<b>Value</b>	<b>Required/Optional</b>
sef_prim_axis	This is the primary axis of the component. <b>Range:</b> A real number in UORs <b>Default:</b> No default	Required
sef_sec_axis	The secondary axis of the component <b>Range:</b> A real number in UORs <b>Default:</b> No default	Required
sef_rot	This is the rotation angle of the component. <b>Range:</b> A real number in UORs <b>Default:</b> No default	Optional

Keyword Suffix	Value	Required/Optional
sef_start_ang	This is the start angle of the component. <b>Range:</b> A real number from 0 to 360 (in degrees) <b>Default:</b> No default	Optional
sef_sweep_ang	This is the sweep angle of the component. <b>Range:</b> A real number from -360 to 360 (in degrees) <b>Default:</b> No default	Optional

### sef\_type: sef\_cell

The SEF writer does not support this geometry shape. `sef_symbol` can be used as an alternative.

### sef\_type: sef\_shape

This is a shape component. A `sef_shape` has a list of point geometry attached to the FME geometry of the aggregate.

If the `sef_shape` component is intended to be an aggregate of more than one component (a complex shape), the `sef_element_type` must be set to 14. The only allowable subtypes for a complex shape are `sef_arc`, `sef_curve`, and `sef_line` with a `sef_element_type` of 3 or 4. Furthermore, if we are dealing with a complex shape with subtypes, the shared attributes should not be prefixed by the keyword `component{<number>}`. Please see *SEF Feature Examples* on page 1190 for a complete example.

It is also important to note that donuts are not supported by the SEF writer. The specific attributes for `sef_shape` are shown below:

Keyword Suffix	Value	Required/Optional
sef_element_type	This tells the SEF writer what type of shape element to make the feature into. An element type of 6 is a shape with 3 to 101 points. An element type of 14 is a) a shape with more than 101 points, or b) an aggregate of any number of components with a <code>sef_type</code> of <code>sef_arc</code> , <code>sef_crv</code> , or <code>sef_line</code> with a <code>sef_element_type</code> of 3 and 4.  <b>Tip:</b> By setting this attribute, the user can force the <code>sef_shape</code> into a particular element. If this keyword is not set, or if the user tries to have a <code>simple_shape</code> with more than 101 points, the SEF writer will automatically detect the appropriate <code>sef_element_type</code> .  <b>Range:</b> 6   14	Required, if the element is intended to be an aggregate of subtypes. Otherwise, not required.

### sef\_type: sef\_line

This is the line component. A `sef_line` has no other geometry attribute besides the list of points attached to the FME geometry of the aggregate.

If the `sef_line` component is intended to be an aggregate of more than one component (a complex string), the `sef_element_type` must be set to 12. The only allowable subtypes for a complex string are `sef_arc`, `sef_curve`, and `sef_line` with `sef_element_type` of 3 or 4. Furthermore, if we are dealing with a complex string with subtypes, the shared attributes should not be prefixed by the keyword `component{<number>}`. Please see *SEF Feature Examples* on page 1190 for a complete example.

The specific attributes for `sef_arc` are shown below:

Keyword Suffix	Value	Required/Optional
<code>sef_element_type</code>	<p>This tells the SEF writer which line type to store the feature as. An element type of 3 is a line with 2 points only. An element type of 4 is a line with 3 to 101 points. An element type of 12 is slightly more complicated. It can be a) a line with more than 101 points or b) an aggregate of any number of <code>sef_arc</code>, <code>sef_crv</code> or <code>sef_line</code> with <code>sef_element_type</code> of 3 or 4.</p> <p><b>Tip:</b> By setting this attribute, the user can force the <code>sef_shape</code> into a particular element. If this keyword is not set, or if the user tries to have a element type with more than the allowed number of points, the SEF writer will automatically detect the appropriate <code>sef_element_type</code>.</p> <p><b>Range:</b> 3   4   12</p>	Required, if the element is intended to be an aggregate of subtypes. Otherwise, not required.

### sef\_type: sef\_curve

This is the curve component of the SEF feature. The coordinates of the curve should be attached to the FME geometry of the aggregate. The SEF writer currently requires that a curve has at least 2 points. There are no additional attributes for this component type.

### sef\_type: sef\_symbol

This is the symbol component of the SEF feature. Each `sef_symbol` component passed to the writer must have exactly 1 point within the aggregate. Additional attributes for `sef_symbol` are as follows:

Keyword Suffix	Value	Required/Optional
<code>sef_character</code>	The character symbol attached to the component. <b>Range:</b> An integer from 1 to 255 <b>Default:</b> No default	Required
<code>sef_font</code>	The font for the symbol which is attached to the component. <b>Range:</b> An integer from 0 to 127 <b>Default:</b> No default	Optional
<code>sef_height</code>	The height of the symbol in absolute Units of Resolution. <b>Range:</b> An integer from 6 to 357913941 <b>Default:</b> No default	Optional
<code>sef_justification</code>	The justification of the symbol of the component. <b>Range:</b> An integer from 0 to 14 <b>Default:</b> No default	Optional
<code>sef_width</code>	The width of the symbol in absolute Units of Resolution. <b>Range:</b> An integer from 6 to 357913941 <b>Default:</b> No default	Optional
<code>sef_rot</code>	This is the rotation angle of the component. <b>Range:</b> A real number in UORs <b>Default:</b> No default	Optional
<code>sef_insert_pt_x</code>	This is the x coordinate of the insertion point. This will become the x coordinate of the translated origin. <b>Range:</b> Any real number <b>Default:</b> No default	Optional
<code>sef_insert_pt_y</code>	This is the y coordinate of the insertion point. This will become the y coordinate of the translated origin. <b>Range:</b> Any real number <b>Default:</b> No default	Optional

### sef\_type: sef\_text

This is the text component of the SEF feature. Each `sef_text` component passed to the writer must have exactly one point within the aggregate, otherwise it will not be process correctly.

The specific attributes for `sef_text` are shown below:

<b>Keyword Suffix</b>	<b>Value</b>	<b>Required/Optional</b>
<code>sef_ch</code>	Any ASCII character text string.	Required
<code>sef_font</code>	The font of <code>sef_ch</code> . <b>Range:</b> An integer from 0 to 127. <b>Default:</b> No default	Optional
<code>sef_height</code>	The height of <code>sef_ch</code> in absolute Units of Resolution. <b>Range:</b> An integer from 6 to 357913941 <b>Default:</b> No default	Optional
<code>sef_justification</code>	The justification of <code>sef_textsef_ch_string</code> <b>Range:</b> An integer from 0 to 14. <b>Default:</b> No default	Optional
<code>sef_width</code>	The width of the text string in absolute Units of Resolution. <b>Range:</b> An integer from 6 to 357913941 <b>Default:</b> No default	Optional
<code>sef_rot</code>	This is the rotation angle of the component. <b>Range:</b> A real number in UORs <b>Default:</b> No default	Optional
<code>sef_insert_pt_x</code>	This is the x coordinate of the insertion point. This will become the x coordinate of the translated origin. <b>Range:</b> Any real number <b>Default:</b> No default	Optional
<code>sef_insert_pt_y</code>	This is the y coordinate of the insertion point. This will become the y coordinate of the translated origin. <b>Range:</b> Any real number <b>Default:</b> No default	Optional

### **sef\_type: sef\_fixed\_text\_node**

This is the fixed text node component of the SEF feature. Each `sef_fixed_text_node` component passed to the writer must have exactly one point within the aggregate.

The specific attributes for `sef_fixed_text_node` are shown below:

<b>Keyword Suffix</b>	<b>Value</b>	<b>Required/Optional</b>
<code>sef_ch</code>	Any ASCII character text string.	Required
<code>sef_font</code>	The font of <code>sef_ch</code> . <b>Range:</b> An integer from 0 to 127. <b>Default:</b> No default	Optional

<b>Keyword Suffix</b>	<b>Value</b>	<b>Required/Optional</b>
sef_height	The height of sef_ch in absolute Units of Resolution. <b>Range:</b> An integer from 6 to 357913941 <b>Default:</b> No default	Optional
sef_node_justification	The justification of sef_ch. <b>Range:</b> An integer from 0 to 14. <b>Default:</b> No default	Optional
sef_width	The width of the text string in absolute Units of Resolution. <b>Range:</b> An integer from 6 to 357913941 <b>Default:</b> No default	Optional
sef_line_spacing	The spacing between each line of the sef_ch in absolute Units of Resolution. This is usually half of height. <b>Range:</b> An integer from 6 to 357913941 <b>Default:</b> No default	Optional
sef_rot	This is the rotation angle of the component. <b>Range:</b> A real number in UORs <b>Default:</b> No default	Optional
sef_insert_pt_x	This is the x coordinate of the insertion point. This will become the x coordinate of the translated origin. <b>Range:</b> Any real number <b>Default:</b> No default	Optional
sef_insert_pt_y	This is the y coordinate of the insertion point. This will become the y coordinate of the translated origin. <b>Range:</b> Any real number <b>Default:</b> No default	Optional

### **sef\_type: sef\_display\_text\_node**

This is the displayable text node component of the SEF feature. Each `sef_display_text_node` component passed to the writer must have exactly one point within the aggregate.

The specific attributes for `sef_display_text_node` are shown below:

<b>Keyword Suffix</b>	<b>Value</b>	<b>Required/Optional</b>
sef_font	The font of sef_ch. <b>Range:</b> An integer from 0 to 127. <b>Default:</b> No default	Optional
sef_height	The height of sef_ch in absolute Units of Resolution. <b>Range:</b> An integer from 6 to 357913941 <b>Default:</b> No default	Optional

<b>Keyword Suffix</b>	<b>Value</b>	<b>Required/Optional</b>
sef_node_justification	The justification of sef_ch <b>Range:</b> An integer from 0 to 14. <b>Default:</b> No default	Optional
sef_width	The width of the text string in absolute Units of Resolution. <b>Range:</b> An integer from 6 to 357913941 <b>Default:</b> No default	Optional
sef_line_spacing	The spacing between each line of the sef_ch in absolute Units of Resolution. This is usually half of height. <b>Range:</b> An integer from 6 to 357913941 <b>Default:</b> No default	Optional
sef_rot	This is the rotation angle of the component. <b>Range:</b> A real number in UORs <b>Default:</b> No default	Optional
sef_insert_pt_x	This is the x coordinate of the insertion point. This will become the x coordinate of the translated origin. <b>Range:</b> Any real number <b>Default:</b> No default	Optional
sef_insert_pt_y	This is the y coordinate of the insertion point. This will become the y coordinate of the translated origin. <b>Range:</b> Any real number <b>Default:</b> No default	Optional

### **sef\_type: sef\_complement\_text\_node**

This is the complement text node component of the SEF feature. Each `sef_complement_text_node` component passed to the writer must have exactly one coordinate within the aggregate.

The specific attributes for `sef_complement_text_node` are shown below:

<b>Keyword Suffix</b>	<b>Value</b>	<b>Required/Optional</b>
sef_font	The font of sef_ch. <b>Range:</b> An integer from 0 to 127. <b>Default:</b> No default	Optional
sef_height	The height of sef_ch in absolute Units of Resolution. <b>Range:</b> An integer from 6 to 357913941 <b>Default:</b> No default	Optional
sef_node_justification	The justification of sef_ch. <b>Range:</b> An integer from 0 to 14. <b>Default:</b> No default	Optional

<b>Keyword Suffix</b>	<b>Value</b>	<b>Required/Optional</b>
sef_width	The width of the text string in absolute Units of Resolution. <b>Range:</b> An integer from 6 to 357913941 <b>Default:</b> No default	Optional
sef_line_spacing	The spacing between each line of the sef_ch in absolute Units of Resolution. This is usually half of height. <b>Range:</b> An integer from 6 to 357913941 <b>Default:</b> No default	Optional
sef_rot	This is the rotation angle of the component. <b>Range:</b> A real number in UORs <b>Default:</b> No default	Optional
sef_insert_pt_x	This is the x coordinate of the insertion point. This will become the x coordinate of the translated origin. <b>Range:</b> Any real number <b>Default:</b> No default	Optional
sef_insert_pt_y	This is the y coordinate of the insertion point. This will become the y coordinate of the translated origin. <b>Range:</b> Any real number <b>Default:</b> No default	Optional

#### **sef\_type: sef\_repeat\_text\_node**

This is the repeat text node component of the SEF feature. Each `sef_repeat_text_node` component passed to the writer must have exactly one coordinate within the aggregate.

The specific attributes for `sef_repeat_text_node` are shown below:

<b>Keyword Suffix</b>	<b>Value</b>	<b>Required/Optional</b>
sef_font	The font of sef_ch. <b>Range:</b> An integer from 0 to 127. <b>Default:</b> No default	Optional
sef_height	The height of sef_ch in absolute Units of Resolution. <b>Range:</b> An integer from 6 to 357913941 <b>Default:</b> No default	Optional
sef_node_justification	The justification of sef_ch <b>Range:</b> An integer from 0 to 14. <b>Default:</b> No default	Optional
sef_width	The width of the text string in absolute Units of Resolution. <b>Range:</b> An integer from 6 to 357913941 <b>Default:</b> No default	Optional

<b>Keyword Suffix</b>	<b>Value</b>	<b>Required/Optional</b>
sef_line_spacing	The spacing between each line of the sef_ch in absolute Units of Resolution. This is usually half of height. <b>Range:</b> An integer from 6 to 357913941 <b>Default:</b> No default	Optional
sef_ref_com	A text string of up to 31 alphanumeric characters including the underbars `_' describing the reference component. This defaults to an empty string if not specified.	Optional
sef_rot	This is the rotation angle of the component. <b>Range:</b> A real number in UORs <b>Default:</b> No default	Optional
sef_insert_pt_x	This is the x coordinate of the insertion point. This will become the x coordinate of the translated origin. <b>Range:</b> Any real number <b>Default:</b> No default	Optional
sef_insert_pt_y	This is the y coordinate of the insertion point. This will become the y coordinate of the translated origin. <b>Range:</b> Any real number <b>Default:</b> No default	Optional

#### **sef\_type: sef\_nongraphic**

This contains only the user-defined attributes.

## **SEF Graphical Feature References**

A SEF Graphical Feature Reference is the graphical portion of the SEF Reference Feature and is not a stand-alone feature. When a graphical component is added as part of the SEF Reference feature, that component will automatically become a graphical feature reference. There is no need to explicitly indicate that a feature is a graphical feature reference.

If it is the case that there are multiple components within a graphical feature reference, these components have to further aggregated together. For example, if a graphical feature reference contains a text component and a complex string with subtype line and curve, the complex string (which is an aggregate already) should be aggregated together with the text component to form the graphical feature reference.

This feature reference has the same structure as a SEF Component but with the addition of the following attributes:

<b>Keyword Suffix</b>	<b>Value</b>	<b>Required/Optional</b>
sef_dgn	The name of the design file which is the source of the feature data.	Optional

<b>Keyword Suffix</b>	<b>Value</b>	<b>Required/Optional</b>
sef_map	This represents the map sheet ID of the map which is the source of the feature data.	Optional
sef_transaction {<number>}	The transaction type of the feature. Note that <number> is an integer which starts from 0 to indicate the order of the feature states. ADD and EDIT are for first transaction only, where as the rest are for any additional transactions. <b>Range:</b> ADD   EDIT OUTSTANDING   DESTATWRT   PLR   LITE <b>Default:</b> No default	Optional
sef_rbprimry	This represents the feature's graphic identifier.	Optional
sef_rbscndry	This represents the feature's UFID.	Optional
sef_segment	This represents the segment name of the feature.	Optional
sef_tsetid	This is the ID which is used to group main features and reference features.	Optional
sef_mrflag	This is used to identify which main feature is associated with a reference feature.	Optional
sef_dgntype	An integer which specifies whether the feature is a GEO or DETAIL feature. <b>Range:</b> 1 (GEO)   2 (DETAIL) <b>Default:</b> No default	Optional

## 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.

As previously stated, there are three types of feature in SEF: a SEF Workset Info Feature, a SEF Reference Feature and SEF Regular features.

The following is the keyword for all the SEF Features:

Keyword Suffix	Value	Required/Optional
sef_feature	This indicates which type of SEF feature the feature is. If this is not specified, then the feature is treated as a regular feature. <b>Range:</b> sef_workset_info   sef_ref_feat   sef_feat <b>Default:</b> sef_feat	Optional

### sef\_feature: sef\_workset\_info

This is the SEF Workset Info feature and it provides the writer with all the information necessary to create a workset in FRAMME when the output file is exported to FRAMME. This must be the first feature sent to the writer.

Additional attributes for the SEF Header Feature are shown below:

Keyword Suffix	Value	Required/Optional
sef_workset_ type	This tag indicates the type of the workset. <b>Range:</b> READ_WRITE   READ_ONLY   ALTERNATE_SCHEME   LITE <b>Default:</b> READ_ONLY	Optional
sef_segment {<number>}	This tag represents the segment name associated with the workset. There can be more than one occurrences of this tag if the workset has multiple segments. Note that <number> is an integer that starts at 0. If this is not specified, this is defaulted to an empty string.	Optional
sef_schema	This keyword is used only in conjunction with ALTERNATE_SCHEME. It is an alphanumeric text string with a limit of 16 characters that represents the name of the alternate scheme.	Optional

### sef\_feature: sef\_ref\_feature

A Reference Feature is a feature referenced at multiple places in a design file. This feature allows us to group together separate graphical elements, spanning over several maps, which represent the same item. For instance, a reference feature should be used in place of a regular SEF feature when a road spans over more than one map.

There are three parts to a SEF Reference Feature: reference attributes, any number of nongraphical components, and any number of graphical feature references.

The specific attributes for a SEF reference feature are shown below:

<b>Keyword Suffix</b>	<b>Value</b>	<b>Required/Optional</b>
sef_st {<number>}	This is a text string with a maximum of 31 characters specifying the state of the feature. Note that <number> is an integer which starts from 0 to indicate the order of the feature states. If none is specified, an empty string is used.	Optional
sef_out_of_scope	If the feature continues beyond the area defined by the index shapes in a SEF file, then this should be specified as <i>yes</i> . Use this only if the user is not certain about which component is out of scope. If he/she knows which component is out of scope, this tag should be used within the component. <b>Range:</b> <i>yes</i> <b>Default:</b> No default	Optional
sef_ref_key	This represents the key or ID number of the multi-referenced features. Use this key only when the graphic references are not all included in a single reference section.	Optional
sef_reload	This is the reload option for the reference feature. <b>Range:</b> NO_RELOAD   LITE <b>Default:</b> No default	Optional
sef_wid	This is the workset ID of the unloaded workset.	Optional
sef_schema_no	This is used with the <i>sef_wid</i> to uniquely identify a workset.	Optional

### **sef\_feature: sef\_feat**

These are the regular features for the SEF writer. Each of these features has a feature type as its name in addition to other feature specific attributes. Within each SEF feature lies many components and each component has a *sef\_component*{<number>} prefix, which denotes the order of the components. For more information on components, please refer to the *SEF Component* section.

The attributes for the SEF Features are shown below:

<b>Keyword Suffix</b>	<b>Value</b>	<b>Required/Optional</b>
sef_st {<number>}	This is a text string with a maximum of 31 characters specifying the state of the feature. Note that <number> is an integer which starts from 0 to indicate the order of the feature states. If none are specified, then an empty string is used as the default.	Optional

<b>Keyword Suffix</b>	<b>Value</b>	<b>Required/Optional</b>
sef_dgn	The name of the design file which is the source of the feature data.	Optional
sef_map	This represents the map sheet ID of the map which is the source of the feature data.	Optional
sef_reload	This is the reload option for the reference feature. Tags which are out of range are simply ignored. <b>Range:</b> NO_RELOAD   LITE <b>Default:</b> No default	Optional
sef_wid	This is the workset ID of the unloaded workset.	Optional
sef_schema_no	This is used with the sef_wid to uniquely identify a workset.	Optional
sef_transaction {<number>}	The transaction type of the feature. Note that <number> is an integer which starts from 0 to indicate the order of the feature states. ADD and EDIT are for first transaction only, where as the rest are for any additional transactions. <b>Range:</b> ADD   EDIT OUTSTANDING   DESTATWRT   PLR   LITE <b>Default:</b> No default	Optional
sef_rbprmry	This represents the feature's graphic identifier.	Optional
sef_rbscndry	This represents the feature's UFID.	Optional
sef_segment	This represents the segment name of the feature.	Optional
sef_tsetid	This is the ID which is used to group main features and reference features.	Optional
sef_mrflag	This is used to identify which main feature is associated with a reference feature.	Optional
sef_dgntype	An integer which specifies whether the feature is a GEO or DETAIL feature. <b>Range:</b> 1 (GEO)   2 (DETAIL) <b>Default:</b> No default	Optional

## Geometry Representation

As mentioned in the earlier sections, SEF features can have very complex geometry types. Within each feature lies many components. In order to handle the geometry inside of each component, the SEF writer requires all geometry attributes associated with

coordinates to be stored in the FME geometry of the SEF feature. The SEF writer expects all SEF features to be aggregates if they have more than one graphical component. As a rule of thumb, if there is more than one graphical component with a feature, the feature must be aggregated together.

For example, if we were to construct a two-point line with user-defined attributes, since the feature contains one geometry component, we can do either of the following:

The two components are constructed using the `sef_component{}` list structure. Through this, the user can control exactly which component sections the attributes fall into.

```

FACTORY_DEF * CreationFactory                                \
  2D_GEOMETRY 1 1 2 2                                     \
  CREATE_AT_END                                           \
  NUMBER_TO_CREATE 1                                       \
  OUTPUT FEATURE_TYPE feat                                 \
    sef_component{0}.sef_type sef_line                    \
    sef_component{0}.streetName"Dow St"                   \
    sef_component{0}.sef_lbl lineComponent                \
    sef_component{1}.sef_type sef_nongraphic              \
    sef_component{1}.streetNumber 6th                     \
    sef_component{1}.sef_lbl ngComponent                  \

```

The resulting feature to the output file is:

```

feat
{
  lbl="feat";
  st="";
  cmp { lbl="lineComponent"; typ=line;
    att
    {
      streetName="Dow St";
    } /* att section */
    geo
    {
      fst_pnt=1,1;
      snd_pnt=2,2;
    } /* geo section */
  } /* cmp section */
  cmp { lbl="ngComponent"; typ=n_g;
    att
    {
      streetNumber="6th";
    } /* att section */
  } /* cmp section */
} /* feat */

```

1. Only the graphical component (the two-point line) is specified. All other user-defined attributes will automatically fall into the component section preceding the graphical component section.

```

FACTORY_DEF * CreationFactory                                \
  2D_GEOMETRY 1 1 2 2                                     \
  CREATE_AT_END                                           \

```

```

NUMBER_TO_CREATE 1 \
OUTPUT FEATURE_TYPE feat \
    sef_type sef_line \
    sef_lbl lineComponent \
    streetName "Dow St" \
    streetNumber 6th

```

The resulting feature to the output file is:

```

feat
{
    lbl="feat";
    st="";
    cmp { lbl="ngComponent"; typ=n_g;
        att
        {
            streetNumber="6th";
            streetName="Dow St";
        } /* att section */
    } /* cmp section */
    cmp { lbl="ngComponent"; typ=line;
        geo
        {
            fst_pnt=1,1;
            snd_pnt=2,2;
        } /* geo section */
    } /* cmp section */
} /* feat */

```

If we were to construct a feature with more than one geometry component (for instance, a curve and a complex line), we would have to use an `AggregateFactory` to aggregate the different geometries together:

```

# A curve.
FACTORY_DEF * CreationFactory\
    2D_GEOMETRY 1 1 2 2 3 3 4 4\
    CREATE_AT_END \
    NUMBER_TO_CREATE 1\
    OUTPUT FEATURE_TYPE aggregate3\
        sef_type sef_curve\
        roadname 1st

# A two point line forced into a complex
# string.
FACTORY_DEF * CreationFactory\
    2D_GEOMETRY 1 1 2 2\
    CREATE_AT_END \
    NUMBER_TO_CREATE 1\
    OUTPUT FEATURE_TYPE aggregate3\
        sef_type sef_line\
        sef_element_type 12\
        signnumber 4

# Aggregate factory to aggregate the curve

```

```
# and the complex string together.
FACTORY_DEF * AggregateFactory \
  INPUT FEATURE_TYPE aggregate3 \
  LIST_NAME sef_component{ }\
  OUTPUT AGGREGATE FEATURE_TYPE aggregate3
```

The resulting feature to the output file would be:

```
feat
{
    lbl="aggregate3";
    st="";
    cmp { lbl=""; typ=crv;
        att
        {
            roadname="1st";
        } /* att section */
        geo
        {
            num_pnts=4;
            pnts=1,1 :
                                2,2 :
                                3,3 :
                                4,4;
        } /* geo section */
    } /* cmp section */
    cmp { lbl=""; typ=cplx_str;
        att
        {
            signnumber="4";
            elm_cnt=1;
        } /* att section */
        subtype=str;
        {
            geo
            {
                                num_pnts=2;
                                pnts=1,1 :
                                2,2;
                                /* geo section */
            }
        } /* subtype section */
    } /* cmp section */
} /* feat */
```

## SEF Feature Examples

### 1. SEF Workset Info Feature

```
FACTORY_DEF * CreationFactory\
  CREATE_AT_END \
  NUMBER_TO_CREATE 1\
  OUTPUT FEATURE_TYPE sef_header_feature\
    sef_featuresef_workset_info\
    sef_workset_typeREAD_ONLY\
    sef_segment{0}segment1
```

The output feature is:

```

workset_info
{
  name="sef_header";
  type=READ_ONLY;/* If changed to ALTERNATE_SCHEMA, add a schema tag
*/
  segment="segment1";
}

```

**1. A reference feature with a complex string, a text and a symbol component. Note that all the graphical components are turned into graphical feature references.**

```

# curve subtype for complex string
FACTORY_DEF * CreationFactory\
  2D_GEOMETRY 1 1 2 2 3 3 4 4\
  CREATE_AT_END \
  NUMBER_TO_CREATE 1\
  OUTPUT FEATURE_TYPE aggregate4\
    sef_type sef_curve\
    sef_out_of_scope begin

# 2 point line subtype for complex string
FACTORY_DEF * CreationFactory \
  2D_GEOMETRY 1 1 2 2 \
  CREATE_AT_END \
  NUMBER_TO_CREATE 1 \
  OUTPUT FEATURE_TYPE aggregate4 \
    sef_type sef_line \
    sef_out_of_scope begin

# Aggregate the curve and the 2 point line together into
# a complex line. The shared attributes of the above
# component ("streetName" for example) should be added
# here.
FACTORY_DEF * AggregateFactory\
  INPUT FEATURE_TYPE aggregate4\
  LIST_NAME sef_component{}\
  OUTPUT AGGREGATE FEATURE_TYPE aggregate4\
    sef_type sef_line\
    sef_lblpier \
    sef_element_type12\
    sef_out_of_scopebegin\
    sef_dgn dgnfile\
    sef_map mapid \
    sef_transaction{0}ADD\
    sef_transaction{1}OUTSTANDING\
    sef_rbprmygid\
    sef_rbscdry ufid\
    sef_segment segment\
    sef_tsetid id \
    sef_mrflag mrflag\
    sef_dgntype 1 \
    streetName "Royal Ave"

# Text.

```

```

FACTORY_DEF * CreationFactory\
2D_GEOMETRY 1 1\
  CREATE_AT_END \
  NUMBER_TO_CREATE 1\
  OUTPUT FEATURE_TYPE aggregate4\
    sef_lbl road\
    sef_type sef_text\
    sef_ch 11 \
    sef_font 12\
    sef_height 13\
    sef_justification 14\
    sef_width 15\
    sef_rot 16 \
    sef_insert_pt_x 17\
    sef_insert_pt_y 18\
    sef_level 19 \
    sef_color 20 \
    sef_style 21 \
    sef_weight 22\
    sef_out_of_scope begin

```

#symbol.

```

FACTORY_DEF * CreationFactory \
  2D_GEOMETRY 1 1\
  CREATE_AT_END \
  NUMBER_TO_CREATE 1 \
  OUTPUT FEATURE_TYPE aggregate4 \
    sef_type sef_symbol\
    sef_lbl symbol \
    sef_character 255 \
    sef_font 127\
    sef_height 10\
    sef_justification 14\
    sef_width 15\
    sef_rot 16 \
    sef_insert_pt_x 2\
    sef_insert_pt_y 2\
    sef_level 60\
    sef_color 16 \
    sef_style 17 \
    sef_weight 18 \
    sef_dgndgn \
    sef_out_of_scope end\
    sef_rbscndryid \
    sef_rbprmry id

```

# This makes the reference feature.

```

FACTORY_DEF * AggregateFactory \
  INPUT FEATURE_TYPE aggregate4 \
  LIST_NAME sef_component{} \
  OUTPUT AGGREGATE FEATURE_TYPE aggregate4\
    sef_feature sef_ref_feat \

```

```

sef_st{0}state1\
sef_st{1} state2 \
sef_ref_keyrefid \
sef_reload NO_RELOAD \
sef_wid      wid \
sef_schema_no num

```

**The resulting reference feature is:**

```

ref_feat
{
  lbl="aggregate4";
  st="state1";
  st="state2";
  ref_key="refid";
  reload="NO_RELOAD";/* DO NOT MODIFY */
  wid="wid";/* DO NOT MODIFY */
  schema_no="num";/* DO NOT MODIFY */
  gr_feat_ref
  {
    dgn="dgnfile";
    map="mapid";
    transaction="ADD";/* DO NOT MODIFY */
    transaction="OUTSTANDING";/* DO NOT MODIFY */
    rbprmry="gid";
    rbscndry="ufid";
    segment="segment";/* DO NOT MODIFY */
    tsetid="id";
    mrflag="mrflag";
    dgntype="1";
    cmp { lbl="pier"; typ=cplx_str;
      out_of_scope=begin;
      att
      {
        {
          streetName="Royal Ave";
          elm_cnt=2;
        } /* att section */
        subtype=crv;
        {
          geo
          {
            num_pnts=4;
            pnts=1,1 :
              2,2 :
              3,3 :
              4,4;
          } /* geo section */
        } /* subtype section */
        subtype=line;
        {
          geo
          {
            fst_pnt=1,1;
            snd_pnt=2,2;
          } /* geo section */
        } /* subtype section */
      } /* cmp section */
    } /* gr_feat_ref*/
  }
  gr_feat_ref
  {
    cmp { lbl="road"; typ=txt;
      out_of_scope=begin;

```

```

att
{
    weight=22;
    level=19;
    text_justification=14;
    text_width=15;
    text_font=12;
    text_ch="11";
    text_height=13;
    color=20;
    style=21;
} /* att section */
geo
{
    rot=16;
    org=1,1;
    trn_org=17,18;
} /* geo section */
} /* cmp section */
} /* gr_feat_ref*/
gr_feat_ref
{
    dgn="dgn";
    rbprmry="id";
    rbscndry="id";
    cmp { lbl="symbol"; typ=sym;
        out_of_scope=end;
        att
        {
            weight=18;
            level=60;
            symbol_character=255;
            symbol_justification=14;
            symbol_width=15;
            symbol_font=127;
            symbol_height=10;
            color=16;
            style=17;
        } /* att section */
        geo
        {
            rot=16;
            org=1,1;
            trn_org=2,2;
        } /* geo section */
    } /* cmp section */
} /* gr_feat_ref*/
} /* ref_feat */

```

**1. A reference feature with a nongraphic component, a graphical reference feature with a complex string and a line, and a graphical reference feature which is a line.**

```

# A nongraphic component.
FACTORY_DEF * CreationFactory \
CREATE_AT_END \
NUMBER_TO_CREATE 1\
OUTPUT FEATURE_TYPE ngComponent\
    sef_type sef_nongraphic\
    sef_lbl grmap\
    streetName102

```

```

# A curve subtype for the complex string.
FACTORY_DEF * CreationFactory \
  2D_GEOMETRY 1 1 2 2 3 3 4 4      \
  CREATE_AT_END                      \
  NUMBER_TO_CREATE 1                 \
  OUTPUT FEATURE_TYPE crvSubType     \
    sef_type      sef_curve

# A line subtype for the complex string.
FACTORY_DEF * CreationFactory\
  2D_GEOMETRY 1 1 2 2      \
  CREATE_AT_END            \
  NUMBER_TO_CREATE 1      \
  OUTPUT FEATURE_TYPE lineSubType \
    sef_type      sef_line

# Aggregate the curve and the line together to form a complex # string that
# will become a component of the first
# graphical reference feature.
FACTORY_DEF * AggregateFactory \
  INPUT FEATURE_TYPE crvSubType \
  INPUT FEATURE_TYPE lineSubType \
  LIST_NAME sef_component{} \
  OUTPUT AGGREGATE FEATURE_TYPE grCmplLineComponent \
    sef_type      sef_line \
    sef_lbl       street \
    sef_element_type 12 \
    streetName    90

# A line which will become part of the first graphical
# reference component.
FACTORY_DEF * CreationFactory      \
  2D_GEOMETRY 1 1 2 2      \
  CREATE_AT_END            \
  NUMBER_TO_CREATE 1      \
  OUTPUT FEATURE_TYPE grLineComponent \
    sef_type      sef_line \
    sef_lbl       bridge \
    streetName    91

# Aggregate the complex string and the above line together
# to form the first graphical reference feature.
FACTORY_DEF * AggregateFactory\
  INPUT FEATURE_TYPE grCmplLineComponent\
  INPUT FEATURE_TYPE grLineComponent \
  LIST_NAME sef_component{}\
  OUTPUT AGGREGATE FEATURE_TYPE grRef \
    sef_out_of_scopebegin\
    sef_dgn dgnfile \
    sef_map mapid \
    sef_transaction{0} ADD \
    sef_transaction{1} OUTSTANDING \
    sef_rbprmrygid\

```

```

    sef_rbscndryufid \
    sef_segment segment\
    sef_tsetid id \
    sef_mrflag mrflag \
    sef_dgntype 1 \
    sef_out_of_scope begin

# A graphical reference with a line only.
FACTORY_DEF * CreationFactory \
    2D_GEOMETRY 1 1 2 2 \
    CREATE_AT_END \
    NUMBER_TO_CREATE 1 \
    OUTPUT FEATURE_TYPE grRef \
        sef_type sef_line\
        sef_out_of_scope end \
        sef_lbl crossStreet \
        streetName 92

# Aggregate the nongraphic component, and the two
# graphical references together to form the reference feature.
FACTORY_DEF * AggregateFactory \
    INPUT FEATURE_TYPE ngComponent \
    INPUT FEATURE_TYPE grRef \
    LIST_NAME sef_component{} \
    OUTPUT AGGREGATE FEATURE_TYPE refFeat \
        sef_feature sef_ref_feat \
        sef_st{0} state1 \
        sef_st{1} state2 \
        sef_out_of_scopeyes \
        sef_ref_keyrefid \
        sef_reloadNO_RELOAD \
        sef_wid wid \
        sef_schema_nonum

```

The output feature is:

```

ref_feat
{
    lbl="refFeat";
    st="state1";
    st="state2";
    out_of_scope;
    ref_key="refid";
    reload="NO_RELOAD";/* DO NOT MODIFY */
    wid="wid";/* DO NOT MODIFY */
    schema_no="num";/* DO NOT MODIFY */
    cmp { lbl="grmap"; typ=n_g;
        att
        {
            streetName="102";
        } /* att section */
    } /* cmp section */
    gr_feat_ref
    {
        dgn="dgnfile";
        map="mapid";
    }
}

```

```

transaction="ADD";/* DO NOT MODIFY */
transaction="OUTSTANDING";/* DO NOT MODIFY */
rbprmry="gid";
rbscndry="ufid";
segment="segment";/* DO NOT MODIFY */
tsetid="id";
mrflag="mrflag";
dgntype="1";
cmp { lbl="street"; typ=cplx_str;
  att
  {
    streetName="90";
    elm_cnt=2;
  } /* att section */
  subtype=crv;
  {
    geo
    {
      num_pnts=4;
      pnts=1,1 :
      2,2 :
      3,3 :
      4,4;

    } /* geo section */
  } /* subtype section */
  subtype=line;
  {
    geo

```

```

        {
            fst_pnt=1,1;
            snd_pnt=2,2;
        } /* geo section */
    } /* subtype section */
} /* cmp section */
cmp { lbl="bridge"; typ=line;
att
{
    streetName="91";
} /* att section */
geo
{
    fst_pnt=1,1;
    snd_pnt=2,2;
} /* geo section */
} /* cmp section */
} /* gr_feat_ref*/
gr_feat_ref
{
    cmp { lbl="crossStreet"; typ=line;
out_of_scope=end;
att
{
    streetName="92";
} /* att section */
geo
{
    fst_pnt=1,1;
    snd_pnt=2,2;
} /* geo section */
} /* cmp section */
} /* gr_feat_ref*/
} /* ref_feat */

```

### 1. A regular feature with a complex string, a complex shape and a non-graphic component.

```

# A curve for the complex string.
FACTORY_DEF * CreationFactory \
    2D_GEOMETRY 1 1 2 2 \
    CREATE_AT_END \
    NUMBER_TO_CREATE 1 \
    OUTPUT FEATURE_TYPE aggregatel \
    sef_type sef_curve \

# A line for the complex line. Since the element type
# is not specified, the SEF writer automatically check
# and determine that this is a two point line.
FACTORY_DEF * CreationFactory \
    2D_GEOMETRY 3 3 4 5\
    CREATE_AT_END \
    NUMBER_TO_CREATE 1 \
    OUTPUT FEATURE_TYPE aggregatel\
    sef_type sef_line

# Aggregate the curve and the two point line together
# into a complex string.
FACTORY_DEF * AggregateFactory\
    INPUT FEATURE_TYPE aggregatel \

```

```

LIST_NAME sef_component{}\
OUTPUT AGGREGATE FEATURE_TYPE aggregate1\
    sef_type sef_line \
    sef_lblroad \
    sef_element_type 12 \
    streetName Broadway

# Simple shape forced into a complex shape.
FACTORY_DEF * CreationFactory \
    2D_GEOMETRY 1 1 10 1 5 10 1 1 \
    CREATE_AT_END \
    NUMBER_TO_CREATE 1 \
    OUTPUT FEATURE_TYPE aggregate1 \
        sef_type sef_shape \
        sef_element_type 14 \
        sef_lbl polygon \
        streetName Lake

# A nongraphic component.
FACTORY_DEF * CreationFactory\
    CREATE_AT_END \
    NUMBER_TO_CREATE 1 \
    OUTPUT FEATURE_TYPE aggregate1 \
        sef_type sef_nongraphic\
        sef_lbl landmark \
        streetName "kingsway"

# This makes the feature.
FACTORY_DEF * AggregateFactory\
    INPUT FEATURE_TYPE aggregate1\
    LIST_NAME sef_component{}\
    OUTPUT AGGREGATE FEATURE_TYPE aggregate1\
        sef_feature sef_feat \
        sef_st{0} state1 \
        sef_st{1} state2 \
        sef_dgn dgnfile \
        sef_map mapid \
        sef_reload NO_RELOAD \
        sef_wid wid \
        sef_schema_no num \
        sef_transaction{0} EDIT \
        sef_transaction{1} PLR \
        sef_rbprmry gid \
        sef_rbscdry ufid\
        sef_segment segment \
        sef_tsetid id \
        sef_mrflag mrflag \
        sef_dgntype 1

```

**The output feature is:**

```

feat
{
    lbl="aggregate1";

```

```

st="state1";
st="state2";
dgn="dgnfile";
map="mapid";
reload="NO_RELOAD";/* DO NOT MODIFY */
wid="wid";/* DO NOT MODIFY */
schema_no="num";/* DO NOT MODIFY */
transaction="EDIT";/* DO NOT MODIFY */
transaction="PLR";/* DO NOT MODIFY */
rbprmry="gid";/* DO NOT MODIFY */
rbscndry="ufid";/* DO NOT MODIFY */
segment="segment";/* DO NOT MODIFY */
tsetid="id";/* DO NOT MODIFY */
mrflag="mrflag";/* DO NOT MODIFY */
dgnstype="1";/* DO NOT MODIFY */
cmp { lbl="road"; typ=cplx_str;
  att
  {
    streetName="Broadway";
    elm_cnt=2;
  } /* att section */
  subtype=crv;
  {
    geo
    {
      num_pnts=2;
      pnts=1,1 :
        2,2;
    } /* geo section */
  } /* subtype section */
  subtype=line;
  {
    geo
    {
      fst_pnt=3,3;
      snd_pnt=4,5;
    } /* geo section */
  } /* subtype section */
} /* cmp section */
cmp { lbl="polygon"; typ=cplx_shp;
  att
  {
    streetName="Lake";
    elm_cnt=1;
  } /* att section */
  subtype=str;
  {
    geo
    {
      num_pnts=4;
      pnts=1,1 :
        10,1 :
        5,10 :
        1,1;
    } /* geo section */
  } /* subtype section */
} /* cmp section */
cmp { lbl="landmark"; typ=n_g;
  att
  {
    streetName="kingsway";
  } /* att section */
} /* cmp section */

```

```
} /* feat */
```

