

# Google Earth KML Reader/Writer (deprecated format)

## FORMAT NOTES:

- **This format is now deprecated – see the KML21 Reader/Writer.**

The KML Reader and Writer modules provide the Feature Manipulation Engine (FME) with the capability to read and write KML files.

## Overview

KML is an xml import/export format used by Google™ Earth. The KML specification is available at [http://www.keyhole.com/kml/kml\\_doc.html](http://www.keyhole.com/kml/kml_doc.html) and <http://www.keyhole.com/kml/docs/webhelp/index.htm>.

Familiarity with the KML documentation is important, as the KML reader and writer were designed to provide direct access to individual XML elements; some advanced functionality requires knowledge of the specification.

## KML Quick Facts

Format Type Identifier	KML
Reader/Writer	<b>Reader:</b> The KML Reader only supports reading feature geometries <b>Writer</b>
Licensing Level	Base
Dependencies	None
Dataset Type	File
Feature Type	Geometry
Typical File Extensions	.kml .kmz
Automated Translation Support	Yes
User-Defined Attributes	Yes. Attributes will be output in a format suitable for display, but KML Schema elements are not currently supported.
Coordinate System Support	LL-84 required
Generic Color Support	Yes
Spatial Index	n/a
Schema Required	No
Transaction Support	No
Geometry Type	xml_type

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

## Reader Overview

The KML reader supports reading only feature geometry from KML documents. Files with a .kmz extension, also known as compressed kml documents, are not supported. Folder hierarchies will be ignored.

## Reader Directives

The KML Writer processes one directive in the mapping file, as shown in the following table. These are all prefixed by the current <ReaderKeyword>\_. By default, the Reader keyword is KML.

## **DATASET**

**Required/Optional:** *Required*

The value for this directive is the file name of the KML file to be read. The normal extension for the files is `.kml`.

An example of the `DATASET` directive in use is:

```
KML_DATASET /user/data/kml/roads.kml
```

[Workbench Parameter: <WorkbenchParameter>](#)

## **Writer Overview**

The KML writer is capable of creating uncompressed (`.kml`) and compressed (`.kmz`) KML documents.

## **Writer Directives**

The KML Writer processes several directives in the mapping file, as shown in the following table. These are all prefixed by the current `<WriterKeyword>_`. By default, the Writer keyword is `KML`.

### **DATASET**

**Required/Optional:** *Required*

Specifies the location of the output KML document. If the dataset extension is `.kmz`, a compressed KML document, capable of embedding raster images and icons, will be created. Otherwise a uncompressed KML document will be created.

[Workbench Parameter: <WorkbenchParameter>](#)

### **DOCUMENT\_NAME**

**Required/Optional:** *Optional*

Specifies the name of the document, as displayed in Google Earth.

[Workbench Parameter: <WorkbenchParameter>](#)

### **DOCUMENT\_DESC**

**Required/Optional:** *Optional*

Specifies the description of the document, as displayed in Google Earth.

[Workbench Parameter: <WorkbenchParameter>](#)

### **DOCUMENT\_LOOKAT\_HEADING**

**Required/Optional:** *Optional*

Specifies `<LookAt><heading/></Lookat>` for the document.

[Workbench Parameter: <WorkbenchParameter>](#)

## **DOCUMENT\_LOOKAT\_TILT**

**Required/Optional:** *Optional*

Specifies `<LookAt><tilt/></Lookat>` for the document.

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

## **DOCUMENT\_LOOKAT\_RANGE**

**Required/Optional:** *Optional*

Specifies `<LookAt><range/></Lookat>` for the document.

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

## **DOCUMENT\_LOOKAT\_LAT**

**Required/Optional:** *Required if heading, tilt, or range is specified. Optional otherwise.*

Specifies `<LookAt><latitude/></Lookat>` for the document.

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

## **DOCUMENT\_LOOKAT\_LONG**

**Required/Optional:** *Required if heading, tilt, or range is specified. Optional otherwise.*

Specifies `<LookAt><longitude/></Lookat>` for the document.

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

## **DOCUMENT\_VISIBILITY**

**Required/Optional:** *Optional*

Provides the value of the `<visibility/>` element for the document.

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

## **ATTR\_IN\_DESCRIPTION**

**Required/Optional:** *Optional*

Determines whether User-Defined attributes will be included in each feature's description field. Depending on the value of the `HTML_DESCRIPTIONS` directive, the attribute names and values will either be represented by a HTML table, or on individual lines with the name separated from the table by a colon.

**Values:** *YES | NO*

**Default value:** *YES*

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

## HTML\_DESCRIPTIONS

**Required/Optional:** *Optional*

Determines whether the description field will be automatically enclosed in a CDATA block.

**Values:** YES | NO

**Default value:** YES

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

## DETECT\_RASTERS

**Required/Optional:** *Optional*

Specifies whether features are checked for raster geometry. Setting the directive to NO will result in a slight performance increase.

**Values:** YES | NO

**Default value:** YES

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

## COPY\_IMAGES\_AND\_ICONS

**Required/Optional:** *Optional*

Specifies whether referenced raster and icon files will be copied to the same directory as the output dataset. This option does not apply when writing KMZ files.

**Values:** YES | NO

**Default value:** NO

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

## INFORMATION\_POINT\_ICON

**Required/Optional:** *Optional*

Specifies the icon to use as a information point. No icon is specified by default. See section on information points for further details.

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

## DEBUG\_FEATURES

**Required/Optional:** *Optional*

If enabled, each feature will be logged after it has been transformed, but before it has been written to the destination dataset.

**Values:** YES | NO

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

## Feature Representation

KML is a hierarchical XML format that consists of multiple KML elements that can nest within each other. A KML file contains one Document element that can contain multiple Folder elements. Document and Folder elements can each contain multiple Placemark, GroundOverlay, ScreenOverlay, and NetworkLink elements. Folder elements can also contain other Folder elements.

### KML Document Hierarchy

To aid in the construction of a Folder hierarchy, the KML writer uses two format attributes: `kml_id`, and `kml_parent`.

The `kml_id` attribute must be unique within a KML file, and is only used within FME: it is not written to the resulting data file, nor used by Google Earth. All features written to the KML writer must have a `kml_id` attribute; if the `kml_id` attribute does not exist, one will be automatically generated by the writer.

Each FME feature can have a `kml_parent` attribute that specifies the `kml_id` of the Folder element that contains the feature. If no `kml_parent` attribute is specified, it is assumed to belong to the Document element, unless the feature has a user-defined schema.

### Supported Feature Types and Attributes

The KML Reader and Writer support two classes of FME Feature Types: User-Defined feature types and KML Element feature types.

KML Element feature types have a fixed schema, and correspond directly to KML elements, whereas User-Defined feature types have user-defined schema, and are internally converted to KML Element feature types.

The KML Reader only supports KML Element feature types; the KML writer supports both classes of feature type.

If a User-Defined feature type is output to the KML writer, its feature type will be changed to either Placemark or GroundOverlay depending on whether the feature has vector or raster geometry.

For each User-Defined feature type, a KML Folder element will be created to hold all features with the same original feature type. This is equivalent to creating a layer for each feature type.

In addition, the KML writer supports feature type fanout; if enabled, a two-level folder hierarchy will be created. The top-level folder will be named after the original feature type, and contain second-level folders named according to the values of the fanout attribute.

During the writing process, the KML Writer applies several transformations to convert User-Defined feature types to KML Element feature types.

Similarly, the KML Reader and Writer support four classes of feature attributes:

- FME attributes: attributes prefixed by `fme_`, such as `fme_color`, that are used as part of the translation process.

- KML Element attributes: attributes that directly correspond to xml elements within the KML file.
- KML format attributes: attributes used internally by the KML Reader and Writer
- User-Defined attributes: attributes defined by the user

As part of the writing process, the KML Writer will use generate KML Element attributes using FME attributes and KML format attributes. The KML Writer will also add a table of User-Defined attribute names and values to each feature's description element, allowing them to be viewed within Google Earth.

Notes:

- Both KML Element attributes and KML Format attributes use the `kml_` prefix
- KML Element attribute names are chosen to closely match the names of the corresponding xml element. For example, `kml_Snippet` corresponds to `<Snippet/>`.
- KML Element attributes use a period (.) to denote an XML path. This is necessary to disambiguate mapping between certain KML elements, such as `<color/>`, that can have different parent elements, and the corresponding KML Element attribute. For example, `kml_PolyStyle.color` corresponds to `<PolyStyle><color/></PolyStyle>`.

## Geometry

KML displays vector overlays using the Placemark element, which corresponds to the Placemark feature type. Similarly, raster overlays are displayed using the GroundOverlay element which corresponds to the GroundOverlay feature type.

## Coordinate System Support

The KML specification requires Placemarks and GroundOverlay elements to specify coordinates using the LL84 coordinate system. The KML writer will check each feature's coordinate system during the writing process, and, if necessary, reproject the feature's geometry to the LL84 coordinate system.

## Coordinate Dimensions

KML requires coordinates to be specified in three dimensions. If two dimensional features are sent to the KML writer, they will be forced to three dimensions.

## Raster Support

By default, the KML writer will check each feature to determine if it is a raster feature that has been created by either the GeoTIFF or JPEG readers. If the feature is a raster, and was properly tagged with the `fme_dataset` attribute, the writer will perform the following transformations on the feature

1. Change the feature type to GroundOverlay.
2. Extract the reprojected bounding box of the feature, and created the corresponding `kml_LatLonBox.north`, `kml_LatLonBox.south`, `kml_LatLonBox.east`, and `kml_LatLonBox.west` attributes.
3. If it doesn't already exist, create the `kml_Icon.href` attribute to hold the location of the raster's file.

4. If writing to a compressed KML file, copy the raster's file into the destination dataset.
5. If the `COPY_IMAGES_AND_ICONS` option is enabled, copy the raster's file to the destination dataset's directory.

## KML Element Feature Types

### Document Feature Type

Corresponds the KML Document element. If multiple features with a Document feature type are sent to the writer, their attributes will be combined to create a single Document feature that contains document-level preferences.

### Element Attributes

```
kml_description <description/>
kml_name        <name/>
kml_visibility  <visibility/>
kml_Snippet     <Snippet/>
kml_LookAt.heading    <LookAt><heading/></LookAt>
kml_LookAt.tilt      <LookAt><tilt/></LookAt>
kml_LookAt.range     <LookAt><heading/></LookAt>
kml_LookAt.latitude  <LookAt><range/></LookAt>
kml_LookAt.longitude <LookAt><longitude/></LookAt>
```

---

**Note:** If heading, tilt, or range is specified for LookAt, both the latitude and longitude must be specified.

---

### Format Attributes

<code>kml_enable_html_description</code>	Value can either be "yes" or "no".
<code>kml_enable_description_attributes</code>	Value can either be "yes" or "no".
<code>kml_icon</code>	Allows specification of the icon for all features in the document.

## Folder

### Element Attributes

Attribute	KML Element
<code>kml_open</code>	<code>&lt;open/&gt;</code>
<code>kml_description</code>	<code>&lt;description/&gt;</code>
<code>kml_name</code>	<code>&lt;name/&gt;</code>
<code>kml_visibility</code>	<code>&lt;visibility/&gt;</code>
<code>kml_Snippet</code>	<code>&lt;Snippet/&gt;</code>
<code>kml_LookAt.heading</code>	<code>&lt;LookAt&gt;&lt;heading/&gt;&lt;/LookAt&gt;</code>
<code>kml_LookAt.tilt</code>	<code>&lt;LookAt&gt;&lt;tilt/&gt;&lt;/LookAt&gt;</code>
<code>kml_LookAt.range</code>	<code>&lt;LookAt&gt;&lt;heading/&gt;&lt;/LookAt&gt;</code>
<code>kml_LookAt.latitude</code>	<code>&lt;LookAt&gt;&lt;range/&gt;&lt;/LookAt&gt;</code>
<code>kml_LookAt.longitude</code>	<code>&lt;LookAt&gt;&lt;longitude/&gt;&lt;/LookAt&gt;</code>

---

**Note:** If heading, tilt, or range is specified for LookAt, both the latitude and longitude must be specified.

---

## Format Attributes

None.

## Placemark

### Element Attributes

Attribute	KML Element
kml_description	<description/>
kml_name	<name/>
kml_visibility	<visibility/>
kml_Snippet	<Snippet/>
kml_styleUrl	<styleUrl/>
kml_address	<address/>
kml_LookAt.heading	<LookAt><heading/></LookAt>
kml_LookAt.tilt	<LookAt><tilt/></LookAt>
kml_LookAt.range	<LookAt><heading/></LookAt>
kml_LookAt.latitude	<LookAt><range/></LookAt>
kml_LookAt.longitude	<LookAt><longitude/></LookAt>
kml_altitudeMode	<altitudeMode/>
kml_extrude	<extrude/>
kml_tessellate	<tessellate/>

---

#### Note:

- The Placemark can also use KML Element attributes from the Style Element. Doing so will create a Style element within the Placemark element.
  - If heading, tilt, or range is specified for LookAt, both the latitude and longitude must be specified.
- 

## Format Attributes

kml\_common\_style The ID of a Style element to be shared by arbitrary groups of features. See the Styling section for more information.

## GroundOverlay

### Element Attributes

Attribute	KML Element
kml_description	<description/>
kml_name	<name/>
kml_visibility	<visibility/>
kml_Snippet	<Snippet/>
kml_drawOrder	<drawOrder/>
kml_Icon.href	<Icon><href/></Icon>
kml_Icon.x	<Icon><x/></Icon>
kml_Icon.y	<Icon><y/></Icon>

kml_Icon.w	<Icon><w/></Icon>
kml_Icon.h	<Icon><h/></Icon>
kml_LatLonBox.rotation	<LatLonBox><rotation/></LatLonBox>
kml_LatLonBox.north	<LatLonBox><north/></LatLonBox>
kml_LatLonBox.south	<LatLonBox><south/></LatLonBox>
kml_LatLonBox.east	<LatLonBox><east/></LatLonBox>
kml_LatLonBox.west	<LatLonBox><west/></LatLonBox>
kml_LookAt.heading	<LookAt><heading/></LookAt>
kml_LookAt.tilt	<LookAt><tilt/></LookAt>
kml_LookAt.range	<LookAt><heading/></LookAt>
kml_LookAt.latitude	<LookAt><range/></LookAt>
kml_LookAt.longitude	<LookAt><longitude/></LookAt>

---

**Note:** If heading, tilt, or range is specified for LookAt, both the latitude and longitude must be specified.

---

### Format Attributes

None.

## ScreenOverlay

### Element Attributes

Attribute	KML Element
kml_description	<description/>
kml_name	<name/>
kml_visibility	<visibility/>
kml_Snippet	<Snippet/>
kml_drawOrder	<drawOrder/>
kml_size.x	x attribute of <size/>
kml_size.y	y attribute of <size/>
kml_size.xunits	xunits attribute of <size/>
kml_size.yunits	yunits attribute of <size/>
kml_overlayXY.x	x attribute of <overlayXY/>
kml_overlayXY.y	y attribute of <overlayXY/>
kml_overlayXY.xunits	xunits attribute of <overlayXY/>
kml_overlayXY.yunits	yunits attribute of <overlayXY/>
kml_screenXY.x	x attribute of <screenXY/>
kml_screenXY.y	y attribute of <screenXY/>
kml_screenXY.xunits	xunits attribute of <screenXY/>
kml_screenXY.yunits	yunits attribute of <screenXY/>
kml_rotation	<rotation/>

### Format Attributes

None.

## NetworkLink

### Element Attributes



<code>fme_color</code>	An FME color specification: a comma-separated list of floating point values ranging from 0 to 1, representing R,G,B values.
<code>fme_fill_color</code>	An FME color specification.
<code>kml_icon_color</code>	An FME color specification.
<code>kml_label_color</code>	An FME color specification.
<code>fme_opacity</code>	A floating point value ranging from 0 to 1 representing the percentage of opaqueness. For example, 0.5 corresponds to 50%. Setting <code>fme_opacity</code> is the same as setting both <code>fme_pen_opacity</code> and <code>fme_fill_opacity</code> .
<code>fme_pen_opacity</code>	A floating point value ranging from 0 to 1.
<code>fme_fill_opacity</code>	A floating point value ranging from 0 to 1.
<code>kml_icon_opacity</code>	A floating point value ranging from 0 to 1.
<code>kml_label_opacity</code>	A floating point value ranging from 0 to 1.

## StyleMap

StyleMap elements are used to provide mouseover effects for Placemark elements with point geometry. See the Styling section for more information.

### Element Attributes

```
kml_styleUrl.normal <StyleMap><Pair><key>normal</key><styleUrl/></Pair>
The 'normal' styleUrl element in a StyleMap
kml_styleUrl.highlight <StyleMap><Pair><key>highlight</key><styleUrl/></
Pair> The 'highlight' styleUrl element in a StyleMap
```

### Format Attributes

None

## Styling

KML has a rich styling model that allows styling to be applied to the lines, polygons, icons, and labels of individual Placemark elements. Each Placemark element can be styled directly, by specifying any of the KML Element attributes from the Style feature type, or indirectly by setting the `kml_styleUrl` attribute to the `kml_Style.id` of a Style feature.

Note: if the value of the `kml_Style.id` of the Style feature is set to `'mystyleid'`, the corresponding value of `kml_styleUrl` on the Placemark feature must be `'#mystyleid'`, where the `'#'` character indicates to Google Earth that the style element is local to the current document.

To some extent, it is possible to mix direct styling, and indirect styling, however, results may vary.

## Common Styling

Indirect specification of styling can dramatically reduce the size of the output KML dataset, and decrease the amount of time taken by Google Earth to load the file, how-

ever creating individual Style elements can be time-consuming. To simplify the use of indirect styling, the KML writer supports "Common Styling".

The general idea of common styling is that you set the styling information on all the members of an arbitrary group of features, and then indicate to the KML writer that it should generate a Style element that will be used by all members of the group of features.

To use common styling, each member of the group must have a `kml_common_style` format attribute that contains a value that is unique to the group. The KML writer will then use the styling information from the first member of that group that it encounters to create a Style element, and styling attributes will be removed from the each of the group members.

Common styling is used by the `KMLStyler` transformer.

## Colorization

KML allows color and transparency effects to the Line (pen), Polygon (fill), Icon, and Label aspects of Placemark elements

### Color and Opacity Attributes

KML combines color and opacity values using a single `<color/>` element within the `<LineStyle/>`, `<PolyStyle/>`, `<LabelStyle/>`, and `<IconStyle/>` elements. The corresponding FME attributes for setting these elements are `kml_LineStyle.color`, `kml_PolyStyle.color`, `kml_LabelStyle.color`, and `kml_IconStyle.color`.

In addition, the KML writer allows color and opacity values to be independently specified using standard FME color specifications. The following table lists the corresponding attributes

<b>KML Element Attribute</b>	<b>Color Attribute</b>	<b>Opacity Attribute</b>
<code>kml_LineStyle.color</code>	<code>fme_color</code>	<code>fme_pen_opacity</code>
<code>kml_PolyStyle.color</code>	<code>fme_fill_color</code>	<code>fme_fill_opacity</code>
<code>kml_LabelStyle.color</code>	<code>kml_label_color</code>	<code>kml_label_opacity</code>
<code>kml_IconStyle.color</code>	<code>kml_icon_color</code>	<code>kml_icon_opacity</code>

Notes:

1. If the KML Element attribute exists, the corresponding color and opacity attributes will be ignored
2. The `fme_opacity` attribute can be used to simultaneously specify both pen and fill opacity.
3. Color attributes have values of the form R,G,B where R G and B are floating point numbers ranging in value from 0 to 1.
4. Opacity attributes have floating point values that range from 0 to 1.
5. If color is not specified, but opacity is, the opacity value will be ignored.

## Colormode

For each of the above styling aspects, KML also supports 'colormode', which can either be 'normal' or 'random'. By default, the colormode is normal; if random is specified, the color specified by the attributes above will be used as the maximum color value. For example, if the fmc color 1,1,1 (pure white) is used, the resulting random colors will include all possible RGB values. Similarly, if the fmc color 1,0,0 (pure red) is used, the resulting random colors will all be shades of red. It is important to note that the color randomization is applied by Google Earth as part of its rendering process; FME is not involved in any way.

## Icon Colorization

As noted above, icons can also be colorized. When icons are colorized, each pixel of the icon is multiplied by the color specified. For best results, colorization should be applied to icons that are either grayscale or pure white; icons with existing colors do not tend to colorize well.

## Icons

Icons can be added to Placemark features via two mechanisms: manually via the `kml_Icon.href` element attribute, or via the `kml_icon` format attribute. If `kml_Icon.href` is already present on the feature, the value of `kml_icon` will be ignored.

If a value is specified for `kml_icon`, the icon file will be located using the following heuristic to locate the referenced icon

1. Does the value specify either the full or relative path of an icon file
2. Does the value specify the name of a file in the `$FME_HOME/icons` directory
3. Does the value specify the rootname of a file in the `$FME_HOME/icons` directory (i.e., the file without the `.png` extension)

## Icon Copying

If the destination dataset is a compressed KML file with a `.kmz` extension, icons specified via `kml_icon` will always be copied into the compressed file.

If the `COPY_IMAGES_AND_ICONS` directive is set to `yes`, and the destination dataset is an uncompressed KML file, the icons specified by `kml_icon` will be copied into the same directory as the destination dataset.

Notes:

1. When using icons, it is recommended to always use compressed KML files.
2. Icons specified via the `kml_Icon.href` attribute will not be copied, nor included in compressed KML files.

## Information Point Creation

If the `kml_icon` format attribute is present on features with non-point geometry, a new point feature will be created with the same attributes as the original feature. This allows the original feature to be easily selected from within the Google Earth map view.

## Bundled Icons

FME comes bundled with a set of icons, located in `$FME_HOME/icons`, that can be used to enhance KML datasets. When using the bundled icons, only the root filename needs to be used. For example, to use `K5.png` as an icon, the `kml_icon` attribute can be set to `"K5"`.

Within the bundled icons, there are two classes of icons: pre-colored and colorizable. Pre-colored icons can be used as-is, whereas colorizable icons are white shapes that are intended to be used in conjunction with Google Earth's icon colorization. Pre-colored icons are, by convention, named with a letter followed by a number, whereas colorizable icons are named with two-digit numbers.

## Style Maps

StyleMap elements are used by Google Earth to create mouse-over effects for Placemarks with point geometry. Generally this is used to change icons and hide or display labels.

A simple style map recipe

To create a style map, you need to do the following

1. Create a feature with a Style feature type to define the 'normal' style.
  - a. Use the KMLStyler transformer to set the styling accordingly
  - b. Set the `kml_Style.id` attributes to `'normal_style'` accordingly
2. Create a feature with a Style feature type to define the 'highlight' style.
  - a. Use the KMLStyler transformer to set the styling accordingly
  - b. Set the `kml_Style.id` attributes to `'highlight_style'` accordingly
3. Create a feature with a StyleMap feature type
  - a. set its `kml_StyleMap.id` attribute to `'stylemap'`
  - b. set its `kml_styleUrl.normal` attribute to `'normal_style'`
  - c. set its `kml_styleUrl.highlight` attribute to `'highlight_style'`
4. Set the `kml_styleUrl` on every feature to use the stylemap to `'stylemap'`

### Style map Tips

Some tips for creating attractive style maps:

- Hide the label when the placemark isn't highlighted by setting `kml_LabelStyle.scale` to 0 on the normal Style feature.
- Use the same icon for both the normal and highlight styles; reduce the icon scale of the normal icon, and use icon colorization to apply a highlight color.

## Description Field Creation

The KML writer provides several enhancements that make the creation of KML `<description>` elements much easier.

### CDATA Blocks

The KML writer can wrap the contents of the description field with a CDATA block:

```
<![CDATA[ description goes here ]]>
```

Wrapping the description contents allows HTML to be easily embedded within the description.

This behavior is controlled by the `HTML_DESCRIPTIONS` directive, which is set to "yes" by default.

### Attribute Tables

A table containing the names and values of each feature's attributes can be automatically appended to the description of each feature. This enables easy display of each feature's schema.

This behavior is controlled by the `ATTR_IN_DESCRIPTION` directive which is set to "yes" by default.

Note: If HTML descriptions are enabled, an HTML table of the attributes will be created. Otherwise the attribute names & values will be colon separated.

### Snippet Creation

If the `kml_Snippet` attribute does not exist on the feature, an empty `kml_Snippet` attribute will be automatically created. This will result in an empty `<Snippet/>` KML element that will force Google Earth to not include a description in the Places menu, which reduces the amount of clutter in the menu.