

# GeoJSON (Geographic JavaScript Object Notation) Reader/Writer

## FORMAT NOTES:

This format is not supported by FME Base Edition.

GeoJSON is a standard for encoding spatial data in JSON structured text. The GeoJSON specification is available at <http://geojson.org>.

## Overview

GeoJSON encodes both geometry and feature information into objects. It also provides support for geometry and feature collections.

GeoJSON represents geometry with a single JSON object. The type of geometry is identified by the value of the *type* key, which must be present in a GeoJSON object. Possible values of the *type* key are *Point*, *LineString*, *Polygon*, *MultiPoint*, *MultiLineString*, and *MultiPolygon*. The geometry coordinates are stored in the *coordinates* key of the geometry object.

```
{
  "type": "LineString",
  "coordinates": [ [100.0, 0.0], [101.0, 1.0] ]
}
```

An aggregate geometry is represented by a GeoJSON object which has a *type* key with value *GeometryCollection*. A *GeometryCollection* object must contain a *geometries* key whose value is an array containing GeoJSON geometry objects.

```
{
  "type": "GeometryCollection",
  "geometries": [
    {
      "type": "LineString",
      "coordinates": [ [100.0, 0.0], [101.0, 1.0] ]
    },
    {
      "type": "MultiPoint",
      "coordinates": [ [100.0, 0.0], [101.0, 1.0] ]
    }
  ]
}
```

A feature is represented by a GeoJSON object which has a *type* key with value *Feature*. A *Feature* object may contain a *geometry* key whose value is a GeoJSON geometry object or a GeoJSON *GeometryCollection* object. A *Feature* object may also contain a *properties* key whose value is an object containing attribute names and values.

```

{
  "type": "Feature",
  "geometry": {
    "type": "MultiPoint",
    "coordinates": [
      [102.0, 2.0],
      [103.0, 3.0]
    ]
  }
}

```

A collection of features is represented by a GeoJSON object which has a *type* key with value *FeatureCollection*. A *FeatureCollection* must contain a *features* key whose value is any array containing GeoJSON *Feature* objects.

```

{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "MultiPoint",
        "coordinates": [
          [102.0, 2.0],
          [103.0, 3.0]
        ]
      }
    },
    {
      "type": "Feature",
      "geometry": {
        "type": "GeometryCollection",
        "geometries": [
          {
            "type": "LineString",
            "coordinates": [ [100.0, 0.0], [101.0, 1.0] ]
          },
          {
            "type": "MultiPoint",
            "coordinates": [ [100.0, 0.0], [101.0, 1.0] ]
          }
        ]
      }
    }
  ]
}

```

Coordinate systems are supported in GeoJSON through the use of a *crs* key and either EPSG codes, OGC URNs, or URLs. If a GeoJSON object has the *crs* key, it is assumed to represent the coordinate reference system of the included features or geometries.

The value of the *crs* key must be an object containing both a *type* and a *properties* key. If the *crs* key is absent, the projection is assumed to be LL84. If an EPSG code or URN is specified, the feature(s) will be tagged with that coordinate system. If a URL is specified, it will be stored in the `json_crs_url` attribute, with its corresponding type in the `json_crs_url_type` attribute.

### EPSG Code

```
{
  "type": "Feature",
  "crs": {
    "type": "EPSG",
    "properties": { "code": 4267 }
  },
  "geometry": {
    "type": "Point",
    "coordinates": [100.0, 0.0]
  }
}
```

### OGC URN

```
{
  "type": "Feature",
  "crs": {
    "type": "OGC",
    "properties": { "urn": "urn:ogc:def:crs:OGC:1.3:CRS84" }
  },
  "geometry": {
    "type": "Point",
    "coordinates": [100.0, 0.0]
  }
}
```

### URL

```
{
  "type": "Feature",
  "crs": {
    "type": "URL",
    "properties": {
      "url": "http://spatialreference.org/ref/epsg/2001/proj4/",
      "type": "proj4"
    }
  },
  "geometry": {
    "type": "Point",
    "coordinates": [100.0, 0.0]
  }
}
```

## GeoJSON Quick Facts

|                               |               |
|-------------------------------|---------------|
| Format Type Identifier        | GeoJSON       |
| Reader/Writer                 | Reader/Writer |
| Licensing Level               | Professional  |
| Dependencies                  | None          |
| Dataset Type                  | File/URL      |
| Feature Type                  |               |
| Typical File Extensions       | .json         |
| Automated Translation Support | Yes           |
| User-Defined Attributes       | No            |
| Coordinate System Support     | Yes           |
| Generic Color Support         | No            |
| Spatial Index                 | Never         |
| Schema Required               | No            |
| Transaction Support           | No            |
| Geometry Type                 | json_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 | no         |
| none             | yes        |          |            |

## Reader Overview

The GeoJSON reader is capable of reading several different GeoJSON structures. If the base JSON element is a GeoJSON geometry object, then the reader will return a single FME feature with the given geometry. If the base JSON element is a GeoJSON *GeometryCollection* object, then the reader returns a single FME feature with an aggregate geometry. In both cases, the FME feature type will be *GeoJSON*.

If the base JSON element is a GeoJSON *Feature* object, then the GeoJSON reader will return a single FME feature. The feature geometry will be taken from the *geometry* key of the *Feature* object, and the feature attributes will be taken from the *properties* key of the *Feature* object. If the base JSON element is a GeoJSON *FeatureCollection* object, then the GeoJSON reader will return an FME feature for each element of the *features* array of the *FeatureCollection* object. In both cases, the FME feature type for each feature will be *GeoJSON*.

If the base JSON element is an array, then any GeoJSON objects in the array are converted into FME features as described above.

If the base JSON element is an object, but not a GeoJSON object, then any value which is a GeoJSON object is converted into FME features as described above, with the exception that the FME feature type is the key name of the corresponding GeoJSON object.

## Coordinate Systems

The GeoJSON reader currently supports coordinate systems in EPSG, OGC URN, or URL format as described in the overview.

## Reader Directives

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

### DATASET

**Required/Optional:** Required

The location of the GeoJSON file to be read. This can be the path to a local or network file, or a URL.

#### Examples:

```
GEOJSON_DATASET c:\json_sample.json
GEOJSON_DATASET \\path\to\network\file.json
GEOJSON_DATASET http://geojson.org/sample
```

### Workbench Parameter: XML Feed

### DELETE\_DOWNLOAD\_FILE

**Required/Optional:** Optional

If the value of this directive is 'Yes', then when the reader has finished reading downloaded GeoJSON text, it will delete the file that the text was downloaded to. The default value is 'Yes'. The value of this directive is only meaningful if the dataset is a URL.

#### Example:

```
GEOJSON_DELETE_DOWNLOAD_FILE No
```

### Workbench Parameter: Delete downloaded file

### PROXY\_URL

**Required/Optional:** Optional

Specifies a proxy server that the reader will be use when accessing a URL dataset. The port number of the proxy server can be set in the URL, or by using the PROXY\_PORT directive.

#### Example:

```
GEOJSON_PROXY_URL www.someproxy.net  
GEOJSON_PROXY_URL www.someproxy.net:8080
```

### Workbench Parameter: Http Proxy URL

#### PROXY\_PORT

**Required/Optional:** Optional

Specifies the port number of the proxy server indicated by the `PROXY_URL` directive. This directive should only be used if the port number was not indicated in the `PROXY_URL` directive. This directive is ignored if the `PROXY_URL` directive has no value.

**Example:**

```
GEOJSON_PROXY_PORT 8080
```

### Workbench Parameter: Http Proxy Port

#### PROXY\_USERNAME

**Required/Optional:** Optional

Specifies the username to use when accessing a password protected proxy server. This directive is ignored if any of the `PROXY_URL`, `PROXY_PASSWORD` or `PROXY_AUTH_METHOD` directives have no value.

**Example:**

```
GEOJSON_PROXY_USERNAME someusername
```

### Workbench Parameter: Http Proxy Username

#### PROXY\_PASSWORD

**Required/Optional:** Optional

Specifies the password to use when accessing a password protected proxy server. This directive is ignored if any of the `PROXY_URL`, `PROXY_USERNAME` or `PROXY_AUTH_METHOD` directives have no value.

**Example:**

```
GEOJSON_PROXY_PASSWORD password1234
```

### Workbench Parameter: Http Proxy Password

#### PROXY\_AUTH\_METHOD

**Required/Optional:** Optional

Specifies the authentication method to use when accessing a password protected proxy server. This directive is ignored if any of the `PROXY_URL`, `PROXY_USERNAME` or `PROXY_PASSWORD` directives have no value. Acceptable values for this directive are 'Basic' or 'Digest'.

**Example:**

```
GEOJSON_PROXY_AUTH_METHOD Basic
```

### Workbench Parameter: Http Proxy Authentication Method

## Writer Overview

The GeoJSON writer writes out a single object, in which each key is an FME feature type, and the value of each key is a GeoJSON *FeatureCollection* object which contains the features of the given type.

## Coordinate Systems

The GeoJSON writer currently supports coordinate systems in EPSG, OGC URN, or URL format as described in the overview. To write a URL coordinate system, ensure that the `json_crs_url` and `json_crs_url_type` attributes are complete and that no coordinate system is set on the feature. If a coordinate system is specified which violates the GeoJSON specifications, the coordinate system will be reprojected to LL84. If no coordinate system is available, no coordinate system will be output. However, it is important to note that the specifications state that GeoJSON objects not specifically tagged with a coordinate reference system are assumed to be the LL84. If multiple coordinate systems exist among features, the first feature will be used to determine the coordinate system for the feature collection.

## Geometry

FME feature geometry is written out in a GeoJSON geometry object as the value of the *geometry* key in a *Feature* type GeoJSON object. Because GeoJSON only supports linear geometry, arcs will be stroked to lines, and ellipses will be stroked to polygons. Also, paths are simplified to a single line, and an FME feature with text geometry only has its location written; the text value is ignored.

The value of the *geometry* key for an FME feature with aggregate geometry will be a *GeometryCollection* object, whose *geometries* key will have an array of GeoJSON geometry objects as its value.

## Writer Directives

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

### DATASET

**Required/Optional:** Required

The file to which the GeoJSON writer should write to. If the file does not exist it will be created.

**Example:**

```
GEOJSON_DATASET c:\geojson_file.json
```

### Workbench Parameter: Destination GeoJSON File

## WRITE\_NULL\_ATTRIBUTE\_VALUES

**Required/Optional:** Optional

This directive specifies whether or not the object containing an FME feature's attributes should contain a key for attributes for which the feature has no value. Possible values for this directive are Yes and No. If the value is No, then the attributes object will only contain keys for which the FME feature has an attribute value. If the value of the directive is Yes, then the output JSON objects will contain keys for every attribute in the feature type schema, and keys for which an FME feature has no attribute value will have a *null* JSON value. The default value of this directive is No.

**Example:**

```
GEOJSON_WRITE_NULL_ATTRIBUTE_VALUES Yes
```

**Workbench Parameter:** Write 'null' for attributes with no value

## WRITER\_CHARSET

**Required/Optional:** Optional

The character set encoding in which the GeoJSON text will be written. Possible values for this directive are UTF-8, UTF-16, UTF-16BE, UTF16-LE, UTF-32, UTF-32BE and UTF-32LE. If no character set is specified, the GeoJSON text will be written in the UTF-8 character set.

**Example:**

```
GEOJSON_WRITER_CHARSET UTF-16
```

**Workbench Parameter:** Output Character Set

## WRITE\_BOM

**Required/Optional:** Optional

The value of this directive specifies whether or not the GeoJSON writer should preface the JSON text with a byte order marker to indicate the endianness of the Unicode text. Possible values for this directive are Yes and No. The default value is No.

**Example:**

```
GEOJSON_WRITE_BOM Yes
```

**Workbench Parameter:** Byte Order Marker

## 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.

## Geometry

The geometry of GeoJSON features may be identified by the `json_type` attribute. The valid values for this attribute are:

| <b>json_type</b>             | <b>Description</b>                |
|------------------------------|-----------------------------------|
| <code>json_no_geom</code>    | FME Feature with no geometry.     |
| <code>json_point</code>      | Point feature.                    |
| <code>json_line</code>       | Linear feature.                   |
| <code>json_polygon</code>    | Simple polygon or donut feature.  |
| <code>json_collection</code> | Feature with multiple geometries. |

### No Geometry

**json\_type:** `json_no_geom`

Features with their `json_type` attribute set to `json_no_geom` do not contain any geometry data.

### Points

**json\_type:** `json_point`

Features with their `json_type` set to `json_point` are single coordinate features or an aggregate of single points.

### Lines

**json\_type:** `json_line`

Features with their `json_type` set to `json_line` are polyline features or an aggregate of polylines.

### Areas

**json\_type:** `json_polygon`

Features with their `json_type` set to `json_polygon` are polygon features which may or may not have interior boundaries, or an aggregate of such polygons.

### Aggregates

**json\_type:** `json_collection`

Features with their `json_type` set to `json_collection` are a heterogeneous collection of multiple geometries.

