Independent
Internet-Draft
Intended status: Informational
Expires: November 22, 2014

H. Butler
Hobu Inc.
M. Daly
Cadcorp
A. Doyle
MIT
S. Gillies
Mapbox Inc.
T. Schaub
OpenGeo
S. Drees

May 21, 2014

The GeoJSON Format
draft-butler-geojson-03

Abstract

GeoJSON is a geospatial data interchange format based on JavaScript Object Notation (JSON). It defines several types of JSON objects and the manner in which they are combined to represent data about geographic features, their properties, and their spatial extents. This document recommends a single coordinate reference system based on WGS 84. Other coordinate reference systems are not recommended.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

GeoJSON is a format for encoding data about geographic features using JavaScript Object Notation (JSON) [RFC7159]. The format is concerned with features in the broadest sense; any thing with qualities that are bounded in geographical space may be a feature whether it is a physical structure or not. The concepts in GeoJSON are not new; they are derived from pre-existing open geographic information system standards (for COM, SQL, and XML) and have been streamlined to better suit web application development using JSON.

GeoJSON comprises the seven concrete geometry types defined in the OpenGIS Simple Features Implementation Specification for SQL [SFSQL]: 0-dimensional Point and MultiPoint; 1-dimensional curve LineString and MultiLineString; 2-dimensional surface Polygon and MultiPolygon; and the heterogeneous GeometryCollection. GeoJSON representations of instances of these geometry types are analogous to the well-known binary (WKB) and text (WKT) representations described in that same specification.

GeoJSON also comprises the types Feature and FeatureCollection. Feature objects in GeoJSON contain a geometry object with one of the above geometry types and additional properties. A FeatureCollection object contains an array of feature objects. This structure is analogous to that of the Web Feature Service (WFS) response to GetFeatures requests specified in [WFSv1] or to a KML Folder of Placemarks [KMLv2.2]. Some implementations of the WFS specification also provide GeoJSON formatted responses to GetFeature requests, but there is no particular service model or feature type ontology implied in the GeoJSON format specification.

Since its initial publication in 2008 [GeoJSON], the GeoJSON format specification has steadily grown in popularity. It is widely used in JavaScript web mapping libraries, JSON-based document databases, and web APIs.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and

"OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.2. Conventions Used in This Document

The ordering of the members of any JSON object defined in this document MUST be considered irrelevant, as specified by [RFC7159].

Some examples use the combination of a JavaScript single line comment (//) followed by an ellipsis (...) as placeholder notation for content deemed irrelevant by the authors. These placeholders must of course be deleted or otherwise replaced, before attempting to validate the corresponding JSON code example.

Whitespace is used in the examples inside this document to help illustrate the data structures, but is not required. Unquoted whitespace is not significant in JSON.

1.3. Definitions

o JavaScript Object Notation (JSON), and the terms object, name, value, array, number, true, false, and null are to be interpreted as defined in [RFC7159].

o Inside this document the term "geometry type" refers to the seven case-sensitive strings: "Point", "MultiPoint", "LineString", "MultiLineString", "Polygon", "MultiPolygon", and "GeometryCollection".

o As another shorthand notation, the term "GeoJSON types" refers to the nine case-sensitive strings "Feature", "FeatureCollection" and the geometry types listed above.

1.4. Example

A GeoJSON feature collection:

```
   {
       "type": "FeatureCollection",
       "features": [{
           "type": "Feature",
           "geometry": {
               "type": "Point",
               "coordinates": [102.0, 0.5]
           },
           "properties": {
               "prop0": "value0"
           }
       }, {
           "type": "Feature",
           "geometry": {
               "type": "LineString",
               "coordinates": [
                   [102.0, 0.0],
                   [103.0, 1.0],
                   [104.0, 0.0],
                   [105.0, 1.0]
               ]
           },
           "properties": {
               "prop0": "value0",
               "prop1": 0.0
           }
       }, {
           "type": "Feature",
           "geometry": {
               "type": "Polygon",
               "coordinates": [
                   [
                       [100.0, 0.0],
                       [101.0, 0.0],
                       [101.0, 1.0],
                       [100.0, 1.0],
                       [100.0, 0.0]
                   ]
               ]
           },
           "properties": {
               "prop0": "value0",
               "prop1": {
                   "this": "that"
               }
           }
       }]
   }
```

2. GeoJSON Object

GeoJSON always consists of a single object. This object (referred to as the GeoJSON object below) represents a geometry, feature, or collection of features.

- o The GeoJSON object MUST have a member with the name "type". The value of the member MUST be one of the GeoJSON types.

- o The GeoJSON object MAY have any number of other members.

- o A GeoJSON object MAY have a "bbox" member, the value of which MUST be a bounding box array (see 4. Bounding Boxes).

2.1. Geometry Object

A geometry object is a GeoJSON object where the "type" value is one of the geometry types. A GeoJSON geometry object of any type other than "GeometryCollection" MUST have a member with the name "coordinates". The value of the coordinates member is always an array. The structure of the elements in this array is determined by the type of geometry. GeoJSON processors MAY interpret geometry objects with empty coordinates arrays as null objects.

2.1.1. Position

A position is the fundamental geometry construct. The "coordinates" member of a geometry object is composed of either:

- o one position (in the case of a Point geometry),

- o an array of positions (LineString or MultiPoint geometries),

- o an array of arrays of positions (Polygons, MultiLineStrings),

- o or a multidimensional array of positions (MultiPolygon).

A position is represented by an array of numbers. There MUST be two or more elements. The first two elements will be longitude and latitude, or easting and northing, precisely in that order and using decimal numbers. Altitude or elevation MAY be included as an optional third element.

Additional position elements MAY be included but MUST follow the three specified above and MAY be ignored by software. Interpretation and meaning of additional elements is beyond the scope of this specification.

Examples of positions and geometries are provided in "Appendix A. Geometry Examples".

2.1.2. Point

For type "Point", the "coordinates" member MUST be a single position.

2.1.3. MultiPoint

For type "MultiPoint", the "coordinates" member MUST be an array of positions.

2.1.4. LineString

For type "LineString", the "coordinates" member MUST be an array of two or more positions.

2.1.5. MultiLineString

For type "MultiLineString", the "coordinates" member MUST be an array of LineString coordinate arrays.

2.1.6. Polygon

To specify a constraint specific to polygons, it is useful to introduce the concept of a linear ring:

- o A linear ring is a closed LineString with 4 or more positions.

- o The first and last positions are equivalent (they represent equivalent points).

- o A linear ring is the boundary of a surface or the boundary of a hole in a surface.

Though a linear ring is not explicitly represented as a GeoJSON geometry type, it leads to a canonical formulation of the Polygon geometry type definition as follows:

- o For type "Polygon", the "coordinates" member MUST be an array of linear ring coordinate arrays.

- o For Polygons with more than one of these rings, the first MUST be the exterior ring and any others MUST be interior rings. The exterior ring bounds the surface and the interiors rings (if present) bound holes within the surface.

2.1.7. MultiPolygon

For type "MultiPolygon", the "coordinates" member MUST be an array of Polygon coordinate arrays.

2.1.8. Geometry Collection

A GeoJSON object with type "GeometryCollection" is a geometry object which represents a collection of geometry objects. A geometry collection MUST have a member with the name "geometries". The value corresponding to "geometries" is an array. Each element in this array is a GeoJSON geometry object.

2.2. Feature Object

A GeoJSON object with the type "Feature" is a feature object.

- o A feature object MUST have a member with the name "geometry". The value of the geometry member SHALL be either a geometry object as defined above or, in the the case that the feature is unlocated, a JSON null value.

- o A feature object MUST have a member with the name "properties". The value of the properties member is an object (any JSON object or a JSON null value).

- o If a feature has a commonly used identifier, that identifier SHOULD be included as a member of the feature object with the name "id" and the value of this member is either a JSON string or number.

2.3. Feature Collection Object

A GeoJSON object with the type "FeatureCollection" is a feature collection object. An object of type "FeatureCollection" MUST have a member with the name "features". The value corresponding to "features" is an array. Each element in the array is a feature object as defined above.

3. Coordinate Reference System Object

The coordinate reference system (CRS) of a GeoJSON object is determined by its OPTIONAL "crs" member (referred to as the CRS object below). If an object has no crs member, then its parent or grandparent object's crs member SHALL be acquired. If no crs member can be so acquired, the default CRS SHALL apply to the GeoJSON object.

The default coordinate reference system for all GeoJSON objects SHALL be a geographic coordinate reference system, using the [WGS84] datum, and with longitude and latitude units of decimal degrees. This coordinate reference system is equivalent to the OGC's urn:ogc:def:crs:OGC::CRS84 [OGCURN]. An OPTIONAL third position element SHALL be the height in meters above the WGS 84 reference ellipsoid. For widest interoperability, GeoJSON data SHOULD use this default coordinate reference system and omit CRS objects.

If an application demands a different coordinate reference system, the following specifications shall be followed:

- o CRS SHOULD be defined as highly in GeoJSON object hierarchy as possible, ideally in the FeatureCollection, and SHOULD NOT be repeated or overridden in child objects.

- o The value of a member named "crs" is a JSON object (referred to as the CRS object below) or JSON null. If the value of CRS is null, no CRS can be assumed.

- o A non-null CRS object has two mandatory members: "type" and "properties".

- o The value of the type member indicates the type of CRS object.

- o The value of the properties member must be an object.

- o The presence of a CRS object SHALL NOT change the ordering of coordinates specified in section 2.1.1.

3.1. Named CRS

A CRS object may indicate a coordinate reference system by name. In this case, the value of its "type" member must be the string "name". The value of its "properties" member must be an object containing a "name" member. The value of that "name" member must be a string identifying a coordinate reference system. OGC CRS URNs such as "urn:ogc:def:crs:OGC::CRS84" are RECOMMENDED over legacy identifiers such as "EPSG:4326".

```
"crs": {
    "type": "name",
    "properties": {
        "name": "urn:ogc:def:crs:OGC::CRS84"
    }
}
```

3.2. Linked CRS

A CRS object may link to CRS parameters on the Web. In this case, the value of its "type" member must be the string "link", and the value of its "properties" member must be a Link object.

A link object has one required member: "href", and one optional member: "type". The value of the required "href" member must be a dereferenceable URI. The value of the optional "type" member must be a string that hints at the format used to represent CRS parameters at the provided URI. Suggested values are: "proj4", "ogcwkt", "esriwkt", but others can be used::

```
"crs": {
    "type": "link",
    "properties": {
        "href": "http://example.com/crs/42",
        "type": "proj4"
    }
}
```

Relative links may be used to direct processors to CRS parameters in an auxiliary file:

```
"crs": {
    "type": "link",
    "properties": {
        "href": "data.crs",
        "type": "ogcwkt"
    }
}
```

4. Bounding Box

A GeoJSON object MAY have a member named "bbox" to include information on the coordinate range for its geometries, features, or feature collections. The value of the bbox member MUST be an array of length 2*n where n is the number of dimensions represented in the contained geometries, with the lowest values for all axes followed by the highest values. The axes order of a bbox follows the axes order of geometries.

Example of a bbox member on a feature:

```
{
    "type": "Feature",
    "bbox": [-180.0, -90.0, 180.0, 90.0],
    "geometry": {
        "type": "Polygon",
        "coordinates": [
            [
                [-180.0, 10.0],
                [20.0, 90.0],
                [180.0, -5.0],
                [-30.0, -90.0]
            ]
        ]
    }
    //...
}
```

Example of a bbox member on a feature collection:

```
{
    "type": "FeatureCollection",
    "bbox": [100.0, 0.0, 105.0, 1.0],
    "features": [
    //...
    ]
}
```

5. Security Considerations

GeoJSON shares security issues common to all JSON content types. See [RFC7159] Section 12 for additional information. GeoJSON does not provide executable content.

As with other geographic data formats, e.g., [KMLv2.2], providing details about the locations of sensitive persons, animals, habitats, and facilities can expose them to unauthorized tracking or injury. GeoJSON does not provide privacy or integrity services; if sensitive data requires privacy or integrity protection the service must be provided externally.

6. Interoperability Considerations

There is a difference of opinion among geographic data formats over whether latitude or longitude come first in a pair of numbers. Longitude comes first in GeoJSON coordinates as it does in [KMLv2.2].

7. IANA Considerations

The MIME media type for GeoJSON text is application/vnd.geo+json.

Type name: application

Subtype name: vnd.geo+json

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: binary

Security considerations: See section 5 above

Interoperability considerations: See section 6 above

Published specification: draft-butler-geojson

Applications that use this media type: various

Additional information:

Magic number(s) : n/a

File extension(s) : .json, .geojson

Macintosh file type code : TEXT

Object Identifiers: n/a

Person to contact for further information:

Sean Gillies

sean.gillies@gmail.com

Intended usage: COMMON

Restrictions on usage: none

8. References

8.1. Normative References

[GeoJSON] Butler, H., Daly, M., Doyle, A., Gillies, S., Schaub, T., and C. Schmidt, "The GeoJSON Format Specification", June 2008.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, March 2014.

8.2. Informative References

[KMLv2.2] Wilson, T., "OGC KML", OGC 07-147r2, April 2008.

[OGCURN] Whiteside, A., "Definition identifier URNs in OGC namespace", OGC 07-092r3, January 2009.

[SFSQL] OpenGIS Consortium, Inc., "OpenGIS Simple Features Specification For SQL Revision 1.1", OGC 99-049, May 1999.

[WFSv1] Vretanos, P., "Web Feature Service Implementation Specification", OGC 02-058, May 2002.

[WGS84] National Imagery and Mapping Agency, "Department of Defense World Geodetic System 1984, Third Edition", 1984.

Appendix A. Geometry Examples

Each of the examples below represents a valid and complete GeoJSON object.

A.1. Points

Point coordinates are in x, y order (easting, northing for projected coordinates, longitude, latitude for geographic coordinates):

```
{
    "type": "Point",
    "coordinates": [100.0, 0.0]
}
```

A.2. LineStrings

Coordinates of LineString are an array of positions (see "2.1.1. Position"):

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

A.3. Polygons

Coordinates of a Polygon are an array of LinearRing (cf. "2.1.6 Polygon") coordinate arrays. The first element in the array represents the exterior ring. Any subsequent elements represent interior rings (or holes).

No holes:

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

With holes:

```
     {
         "type": "Polygon",
         "coordinates": [
             [
                 [100.0, 0.0],
                 [101.0, 0.0],
                 [101.0, 1.0],
                 [100.0, 1.0],
                 [100.0, 0.0]
             ],
             [
                 [100.2, 0.2],
                 [100.8, 0.2],
                 [100.8, 0.8],
                 [100.2, 0.8],
                 [100.2, 0.2]
             ]
         ]
     }
```

A.4. MultiPoints

Coordinates of a MultiPoint are an array of positions::

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

A.5. MultiLineStrings

Coordinates of a MultiLineString are an array of LineString coordinate arrays:

```
     {
         "type": "MultiLineString",
         "coordinates": [
             [
                 [100.0, 0.0],
                 [101.0, 1.0]
             ],
             [
                 [102.0, 2.0],
                 [103.0, 3.0]
             ]
         ]
     }
```

A.6. MultiPolygons

Coordinates of a MultiPolygon are an array of Polygon coordinate arrays:

```
     {
         "type": "MultiPolygon",
         "coordinates": [
             [
                 [
                     [102.0, 2.0],
                     [103.0, 2.0],
                     [103.0, 3.0],
                     [102.0, 3.0],
                     [102.0, 2.0]
                 ]
             ],
             [
                 [
                     [100.0, 0.0],
                     [101.0, 0.0],
                     [101.0, 1.0],
                     [100.0, 1.0],
                     [100.0, 0.0]
                 ],
                 [
                     [100.2, 0.2],
                     [100.8, 0.2],
                     [100.8, 0.8],
                     [100.2, 0.8],
                     [100.2, 0.2]
                 ]
             ]
         ]
     }
```

A.7. GeometryCollections

Each element in the geometries array of a GeometryCollection is one of the geometry objects described above:

```
{
    "type": "GeometryCollection",
    "geometries": [{
        "type": "Point",
        "coordinates": [100.0, 0.0]
    }, {
        "type": "LineString",
        "coordinates": [
            [101.0, 0.0],
            [102.0, 1.0]
        ]
    }]
}
```

Appendix B. Contributors

The GeoJSON format is the product of discussion on the GeoJSON mailing list: http://lists.geojson.org/listinfo.cgi/geojson-geojson.org.

Comments are solicited and should be addressed to the GeoJSON mailing list at geojson@lists.geojson.org or to the GeoJSON issue tracker at https://github.com/geojson/draft-geojson/issues.

Authors' Addresses

H. Butler
Hobu Inc.

M. Daly
Cadcorp

A. Doyle
MIT

S. Gillies
Mapbox Inc.

Email: sean.gillies@gmail.com
URI: http://sgillies.net

T. Schaub
OpenGeo

S. Drees
Rheinaustr. 62
Bonn 53225
DE

Email: stefan@drees.name
URI: http://sdre.es/