

JSGeofencing

A geofencing extension for GeoJson

# Versions

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Version | Comment |
| 10/04/2021 | Guillaume Pelletier | 0.1 | First version |
| 19/04/2021 | Guillaume Pelletier | 0.2 | Update of format |
| 29/04/2021 | Guillaume Pelletier | 0.3 | Update of format |

# Abstract

This specification defines a data model and JSON representation of geofence data that can be used for alerts in a motion and GPS navigation environment. It aims to be an alternative to custom data format and to be unambiguous, extendable and simple to process.

## Copyright notice

Copyright (c) 2021 DotVision and the persons identified as the document authors. All rights reserved.

Table of content

[Versions 1](#_Toc71824659)

[Abstract 2](#_Toc71824660)

[Copyright notice 2](#_Toc71824661)

[Introduction 6](#_Toc71824662)

[Notational convention 6](#_Toc71824663)

[Relation to GeoJSON 6](#_Toc71824664)

[Structure of JSGeofencing objects 9](#_Toc71824665)

[Type signatures 9](#_Toc71824666)

[Data Types 9](#_Toc71824667)

[UTCDate 9](#_Toc71824668)

[LocalDate 9](#_Toc71824669)

[Duration 10](#_Toc71824670)

[Display string localization 10](#_Toc71824671)

[The geofence extension 10](#_Toc71824672)

[Common JSGeofencing properties 10](#_Toc71824673)

[enabled 11](#_Toc71824674)

[consumed 11](#_Toc71824675)

[displayName 11](#_Toc71824676)

[description 11](#_Toc71824677)

[tags 11](#_Toc71824678)

[pre-imodifiers 12](#_Toc71824679)

[post-imodifiers 12](#_Toc71824680)

[Geofence 12](#_Toc71824681)

[@type 12](#_Toc71824682)

[@id 12](#_Toc71824683)

[@context 13](#_Toc71824684)

[comment 13](#_Toc71824685)

[shapes 13](#_Toc71824686)

[nodes 13](#_Toc71824687)

[alerts 13](#_Toc71824688)

[modifiers 13](#_Toc71824689)

[primitives 13](#_Toc71824690)

[iroots 13](#_Toc71824691)

[Shape 14](#_Toc71824692)

[Properties 14](#_Toc71824693)

[type 14](#_Toc71824694)

[radius 14](#_Toc71824695)

[elevation 14](#_Toc71824696)

[priority 15](#_Toc71824697)

[Nodes 15](#_Toc71824698)

[Properties 16](#_Toc71824699)

[@id 16](#_Toc71824700)

[iprimitive 16](#_Toc71824701)

[ishape 16](#_Toc71824702)

[translation 16](#_Toc71824703)

[rotation 16](#_Toc71824704)

[scale 16](#_Toc71824705)

[Alerts 17](#_Toc71824706)

[Scope of these definitions 17](#_Toc71824707)

[Properties 17](#_Toc71824708)

[relativeTo 17](#_Toc71824709)

[category 17](#_Toc71824710)

[severity 18](#_Toc71824711)

[message 18](#_Toc71824712)

[Modifier 19](#_Toc71824713)

[Commons Properties 19](#_Toc71824714)

[scope 19](#_Toc71824715)

[category 19](#_Toc71824716)

[priority 19](#_Toc71824717)

[Modifier objects 20](#_Toc71824718)

[Calendar 20](#_Toc71824719)

[Predicate 21](#_Toc71824720)

[Zone 21](#_Toc71824721)

[Primitives 22](#_Toc71824722)

[Area triggers 22](#_Toc71824723)

[Fence triggers 23](#_Toc71824724)

[Path triggers 24](#_Toc71824725)

[Control Point triggers 25](#_Toc71824726)

[Properties 25](#_Toc71824727)

[evalThreshold 25](#_Toc71824728)

[inodes 25](#_Toc71824729)

[ialerts 26](#_Toc71824730)

[distance 26](#_Toc71824731)

# Introduction

## Notational convention

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [[rfc2119]](https://tools.ietf.org/html/rfc2119).

The underlying format used for this specification is JSON. Consequently, the terms "object" and "array" as well as the four primitive types (strings, numbers, booleans, and null) are to be interpreted as described in Section 1 of[[rfc8259]](https://tools.ietf.org/html/rfc8259).

Some examples in this document contain "partial" JSON documents used for illustrative purposes. In these examples, three periods "..." are used to indicate a portion of the document that has been removed for compactness.

## Relation to GeoJSON

GeoJSON [[rfc7946](https://tools.ietf.org/html/rfc7946)] is an open standard format designed for representing simple geographical features, along with their non-spatial attributes. It is based on the JSON format.

The data for geofence or set of geofences is represented by Feature Object and FeatureCollection Object in the GeoJSON format. In Addition to it:

* The Geometry Object type can be a Point, MultiPoint, LineString, MultiLineString, Polygon, MultiPolygon, and GeometryCollection.
* Every feature properties MUST contain one, and only one, geofence attribute, which is used for carrying the geofence informations.
* Styling is **NOT** part of this specification and MAY be carry by another property of the geoJSON. Some tools already process such style within the properties attribute. For example, [[leaflet](https://leafletjs.com/reference-1.7.1.html#geojson)] use this approach and we MAY relay on few properties :
  + Width
  + LineColor
  + LineAlpha
  + FillColor
  + FillAlpha

Because these extensions are part of the GeoJSON specification, back end implementation **SHOULD** simply ignore it.

For example, application **MAY** add a specific tag to the GeoJson properties such

"properties": {

"style": {

"linecolor": "#02B8FA",

"width": 3

},

…

}

* Most of the serialization sample code sets the layout to "pretty-print" the JSON (with indentation and whitespace for human readability). For production use, you would typically accept the default value of false for this setting, since adding unnecessary whitespace may incur a noticeable, negative impact on performance and bandwidth usage.
* All coordinate values are represented as [longitude, latitude, elevation?] defined in **WGS84**. Optional elevation is expressed as meter above Sea Level.
* GeoJSON is 2D by design. The Geofence MAY represent also 3D volumes such Cylinder, Box, or even Mesh. In this last case, 3D format may rely on standard mesh format such Obj or even GLTF2.0

Following is a sample description body for a geofence represented as a circle geofence geometry in GeoJSON using a center point and a radius

{

"type": "Feature",

"geometry": {

"type": "Point",

"coordinates": [ [ 0, 0 ] ]

},

"properties": {

"geofence": {

"@id": "d42d7ff3-7012-4fbb-9fea-39fa2265b9a2",

"@type": "bf:geofence:1",

"@context": "./context",

"displayName": "DotVision Safe Zones",

"description": "Set a nightly safe zone against 'bad guys' and some monsters, around and above DotVision locations.",

"shapes": [

{

"type": "Circle",

"igeometry": 0,

"radius": 500,

"elevation": { "max": 1000 }

}

],

"nodes": [

{

"displayName": "Headquarter.",

"translation": [ 2.5741880846574667, 48.639946000000016 ],

"ishape": 0

},

{

"displayName": "Hawaii.",

"translation": [ -157.86073811843235, 21.310663208579705 ],

"ishape": 0

}

],

"alerts": [

{

"displayName": "Bad guy's alerts",

"relativeTo": [ "entering", "exiting", "inside", "outside", "above" ]

}

],

"modifiers": [

{

"type": "Predicate",

"displayName": "Bad guy's filter",

"description": "Watch only 'bad guys' or 'kaiju')",

"logic": {

"any": [

{ "var": "Sample.Tags" },

{ "or": [

{"==": [

{ "var": "" },

"bad guys"]

},

{"==": [

{ "var": "" },

"kaiju"]

}]}]}

},

{

"type": "Calendar",

"displayName": "Night",

"description": "Every day from 8pm to 8am D+1",

"expired": "2022-01-01T00:00:00Z",

"validityPeriod": [

{

"startTime": "2021-05-01T20:00:00Z",

"endTime": "2021-05-02T08:00:00Z",

"recurrenceType": "Daily"

}

]

}

],

"primitives": [

{

"displayName": "Safe zone",

"type": "Area",

"inodes": [ 0, 1 ],

"ialerts": [ 0 ],

"pre-imodifiers": [ 0, 1 ]

}

]

},

"style": {

"color": "#FF0000",

"width": 3

}

}

}

# Structure of JSGeofencing objects

A JSGeofencing object is a JSON object, which MUST be valid I-JSON (a stricter subset of JSON), as specified in  [[rfc8259]](https://tools.ietf.org/html/rfc8259). Property names and values are case-sensitive.

The object has a collection of properties, as specified in the following sections. Unless otherwise specified, all properties are optional; omitted properties MUST be treated identically to if that property had the value of null, unless otherwise specified.

## Type signatures

Types signatures are given for all JSON objects in this document. The following conventions are used:

* Foo: Any name that is not a native JSON type means an object for which the properties (and their types) are defined elsewhere within this document.
* Foo[]: An array of objects of type Foo.

## Data Types

In addition to the standard JSON data types, the following data types are used in this specification:

### UTCDate

This is a string in  [[rfc3339]](https://tools.ietf.org/html/rfc3339) date-time format, with the further restrictions that any letters MUST be in upper-case, the time component MUST be included and the time MUST be in UTC. Fractional second values MUST NOT be included unless non-zero and MUST NOT have trailing zeros, to ensure there is only a single representation for each date-time.

For example 2010-10-10T10:10:10.003Z is OK, but 2010-10-10T10:10:10.000Z is invalid and MUST be encoded as 2010-10-10T10:10:10Z.

In common notation, it should be of the form YYYY-MM-DDTHH:MM:SSZ.

### LocalDate

This is a date-time string with no time-zone/offset information. It is otherwise in the same format as UTCDate: YYYY-MM-DDTHH:MM:SS. The time-zone to associate the LocalDate with comes from an associated property, or if no time-zone is associated it defines floating time. Floating date-times are not tied to any specific time-zone. Instead, they occur in every timezone at the same wall-clock time (as opposed to the same instant point in time).

### Duration

A duration is represented by a subset of ISO8601 duration format, as specified by the following ABNF:

dur-secfrac = "." 1\*DIGIT

dur-second = 1\*DIGIT [dur-secfrac] "S"

dur-minute = 1\*DIGIT "M" [dur-second]

dur-hour = 1\*DIGIT "H" [dur-minute]

dur-time = "T" (dur-hour / dur-minute / dur-second)

dur-day = 1\*DIGIT "D"

duration = "P" (dur-day [dur-time] / dur-time)

In addition, the duration MUST NOT include fractional second values unless the fraction is non-zero. A zero duration MUST be represented as “P0D”.

## Display string localization

Some string properties in models are meant for display and, therefore, support localization. JSGeofencing models use JSON-LD's string internationalization support for localization. Each localizable property (e.g. displayName and description) is defined to be a JSON-LD language map ("@container": "@language"). The keys of the language map must be language tag strings (see [BCP 47](https://tools.ietf.org/html/bcp47)). [ISO 639](https://www.loc.gov/standards/iso639-2/php/code_list.php) provides a list of language tags. The default language for JSGeofencing documents is English.

# The geofence extension

The core of JSGeofencing is a JSON object that describe the structure and composition of a geofencing scene. The Top-level element of this object are :

* shapes : define the geometry used in the scene
* nodes: define node describing the link between primitives and shapes.
* modifiers : modifiers applied to other objects.
* alerts : the alert potentially triggered in the scene
* primitives : the geofencing rules, which use shapes and/or values to trigger alerts

These elements are contained in arrays. References between the objects are established by using their indices to lookup the objects in the arrays. The index are 0 zero based.

# Common JSGeofencing properties

This section describes the properties that are common to the various JSGeofencing object types. Specific JSGeofencing object types may only support a subset of these properties.

### enabled

Type : boolean

Required : No

Default : true

Define if the object is active or not. A disabled object will not currently take part of the geofencing pipeline flow session.

### consumed

Type : boolean

Required : No

Default : false

Define if the object is terminated. A consumed object will no longer take part of the geofencing pipeline flow current session.

### displayName

Type : string

Required : No

Default : Null

Localizable : yes

A localizable name for display

### description

Type : string

Required : No

Default : Null

Localizable : yes

A localizable description for display

### tags

Type : string[]

Required : No

Default : Null

A set of string for miscellaneous purpose such search or filtering

### pre-imodifiers

Type: int[]

Required : no

A set of zero index of the defined modifiers to be applied to the current object before the geofencing process. Note that some modifier may have non sense for specific object. This specification do NOT cover this aspect.

### post-imodifiers

Type: int[]

Required : no

A set of zero index of the defined modifiers to be applied to the current object after the geofencing process. Note that some modifier may have non sense for specific object. This specification do NOT cover this aspect.

# Geofence

The geofence object is the main object of the geofencing file. It is located into the properties element of the geo-json feature.

### @type

Type: IRI

Required : Yes

Specifies the type which this object represents. IRIs in DTDL are [JSON-LD IRIs](https://w3c.github.io/json-ld-syntax/#iris) and may be relative or absolute.

A valid geofence object MUST include this property.

### @id

Type: String

Required : Yes

A globally unique identifier, used to associate the object as the same across different systems, geofencing models and views. The value of this property MUST be unique across **ALL** JSGeofencing objects, even if they are of different type. [[RFC4122]](https://tools.ietf.org/html/rfc4122) describes a range of established algorithms to generate universally unique identifiers (UUID), and the random or pseudo-random version is recommended to use.

A valid geofence object MUST include this property.

## @context

Type: IRI

Required : Yes, (at least once in the doc)

The context to use when processing this interface. For this version, it must be set to "bfrst:geofencing:context;1"

## comment

Type: string

Required : No

A comment for geofence authors.

## shapes

Type: Shape Array

Required : Yes

## nodes

Type: Node Array

Required : Yes

## alerts

Type: Alert Array

Required : Yes

## modifiers

Type: Modifier Array

Required : Yes

## primitives

Type: Primitive Array

Required : Yes

## iroots

Type : integer[]

Required: No

Default :[0]

The geofence MAY have a roots definition index which design the roots nodes of the scene. Default is node at index Zero.

# Shape

Shape are the geofencing node relative to geometries. This include but not limited to type of

* Circle
* Rectangle
* Path
* Line
* Polygon

## Properties

### type

Type : string

Required : Yes

### radius

Type : number

Required : No

Default : 0

Shapes with underlying geometry of type Point, MultiPoint, LineString, MultiLineString **MAY** contain radius in properties. radius value is measured in meters, the radius value ranges from 1 to 10000.

Shapes with underlying geometry of type Polygon and MultiPolygon type does **NOT** have a radius property.

### elevation

Type : range

Required : No

Default : None

Shapes **MAY** contain absolute elevation range in properties. This range act as kind of extrusion of the underlying geometry. For example, a circle MAY become a cylinder.

Range values are measured in meters and are expressed as meter above or below Sea Level. A negative value indicate below Sea Level.

The optional elevations of the underlying geometries are not take in account.

A range object has the following properties

* min : number
* max: number

### priority

Type : number

Required : No

Default : 0

Some geometries may overlap. And in this case sorting the geometries with a priority value **MAY** be useful.

priority can range between 0 (lowest priority) and 127 (highest priority).

# Nodes

A node is representing the object referenced by the geofence container. It mainly intended to bind a primitive and a shape.

Each of the nodes can contain an array of indices of its children. This allows modelling a simple hierarchy.

A node may contain a **local** transform. This can be given as a column-major matric array, or with separate translation, rotation and scale properties, where the rotation is given as quaternion.

The local transform matrix is then computed as

**M = T \* R \* S**

The global transform of a node is given by the product of all local transform on the path from the root to the respective node.

Translations are represented as [longitude, latitude, elevation] defined in **WGS84**

Each node MAY refer to a shape, using indices that point into the shape array.

## Properties

### @id

Type: String

Required : Yes

A globally unique identifier, used to associate the object as the same across different systems, geofencing models and views. The value of this property MUST be unique across **ALL** JSGeofencing objects, even if they are of different type. [[RFC4122]](https://tools.ietf.org/html/rfc4122) describes a range of established algorithms to generate universally unique identifiers (UUID), and the random or pseudo-random version is recommended to use.

***Note : this node id is may used as cache key to uniquely identify a logic pair of asset state within this geofencing item.***

### iprimitive

Type: int

Required : Yes

### ishape

Type: int

Required : Yes

### translation

Type: number[3]

Required : No

Default : [0,0,0]

### rotation

Type: number[4]

Required : No

Default : [0,0,0, 1]

### scale

Type: number[3]

Required : No

Default : [1,1,1]

# Alerts

Geofencing pipeline is defined as Location flow associated with an user.

At each new user location, this new location and eventually segments (two consecutive location) are processed against primitives.

Depending the result, one or several alerts MAY be raised.

This alerts are defined according the type of Geofencing primitive.

Note that

* these alerts **MAY** be altered by modifiers.
* each Alert has its own set of Modifiers and has a default behavior.

## Scope of these definitions

* Alert’s definition serve the sole objective of this specification. The mechanism AND/OR format to transmit these alerts is out of scope of this specification. Event format **MUST** be defined through external specification such alarm and telemetry.
* It is common to merge threshold Alert within Geofence. For example a system MAY be interested by speed limit or to identify asset immobility. However, sole threshold alerts **MUST** be defined as outer scope such alarm and telemetry and are **NOT** part of this specification.

## Properties

### relativeTo

Type : string[]

Required : yes

The type of the event that the alarm is related to.

A valid Alert object MUST include this property.

### category

Type : string

Required : yes

Default : geography

The category to which this alert belongs. Alert Categories define groupings of alerts supported by an Geofencing Event server

Standard categories are:

* motion
* geography

Vendors **MAY** add categories, which **MUST** be prefixed with vendor namespace

<namespace>:<category>

### severity

Type : number

Required : No

Default : 1

The urgency of the event. This may be a value in the range of 1 – 1000, with 1 being the lowest severity and 1000 being the highest. Typically, a severity of 1 would indicate in event which is informational in nature, while a value of 1000 would indicate an event of catastrophic nature which could potentially result in severe financial loss or loss of life. It is expected that few server implementations will support 1000 distinct severity levels. Therefore, server developers are responsible for distributing their severity levels across the 1 – 1000 range in such a manner that clients can assume a linear distribution. n. For example, a client wishing to present five severity levels to a user could implement the severity level as shown in the following Figure

|  |  |
| --- | --- |
| Severity | JSGeofencing Severity |
| HIGH | 801 – 1000 |
| MEDIUM HIGH | 601 – 800 |
| MEDIUM | 401 – 600 |
| MEDIUM LOW | 201 – 400 |
| LOW | 1 – 200 |

### message

Type : string

Required : No

Default : None

Localizable : yes

A localizable message for display

# Modifier

JSGeofencing object are extensible using the concept of modifier. A modifier is an object which when applied to another object change his properties and/or behavior. Along this definition, modifiers are mainly use as pre or post filter to primitive, or alerts.

Modifiers can contains modifiers

Every JSGeofencing object **MAY** have one or more modifier.

Standard Modifiers could be of type of :

* Calendar
* Zone
* Predicate

## Commons Properties

### scope

Type : string

Required : No

Default : “asset”

The scope of the modifier. Value can be :

* static : the modifier act as global level and any status/action are related to the model. For example, we can invalidate a zone if ANY of user exit it.
* asset : the modifier act as user level and any status/action are related and specific to this user. For example, a count modifier will store different information for different user and will act accordingly.

### category

Type : string

Required : No

Default : null

The category to which this modifier belongs. Modifier Categories define groupings of modifier supported by Geofencing tools.

### priority

Type : number

Required : No

Default : 0

Some modifiers may be call prior to other.

priority can range between 0 (lowest priority) and 127 (highest priority).

## Modifier objects

### Calendar

Define the expiration and validity of the underlying node.

#### Properties

##### expire

Type : UTCDate | LocalDate

Required : No

The absolute expiration.

Once a Node is expired, the consumed property is set to true.

##### validityPeriods

Type : Array of Period

Required : No

Define the validity periods of the underlying node. A Period object has the following properties

* startTime: UTCDate | LocalDate
* endTime: UTCDate | LocalDate
* recurrence: String. MUST be one of the following values:
  + week
  + day
  + hour

##### count

Type : int

Required : No

Default : 0

The number of time the recurrence repeat.

### Predicate

Define a Boolean expression to be applied over different scenario.

### Zone

Define a geographic area for inclusion or exclusion of geofencing object.

# Primitives

The Geofencing extension MUST contains primitives. Primitive are the core of the geofencing rules. Main Primitives are of types

* Area
* Fence
* Path
* Control Point

Each type of primitive define a set of triggers. Alerts define which triggers are enabled, and various parameters such trigger threshold (see Alerts)

## Area triggers

|  |  |  |
| --- | --- | --- |
| trigger | variables | Description |
| Entering |  | When entering the area. |
| Inside |  | When inside the area. |
| Above |  | When inside but above max height. |
| Under |  | When Inside but under min height. |
| Exiting |  | When exiting the area. |
| Outside |  | When outside the area. |
| Approaching | distance | When near the fence with direction toward to the fence. |
| Leaving | distance | When near the fence with direction away from the fence. |



Note that threshold is Also applied to external fence defined by radius + distance

## Fence triggers

|  |  |  |
| --- | --- | --- |
| trigger | variables | Description |
| Crossing |  | When crossing the fence. |
| Approaching | distance | When near the fence with direction toward to the fence. |
| Leaving | distance | When near the fence with direction away from the fence. |



Note that threshold is Also applied to external fence defined by radius + distance

## Path triggers

|  |  |  |
| --- | --- | --- |
| trigger | variables | Description |
| Approaching | distance | When near the path with direction toward to the path. |
| On Track | distance | When near the path. |
| Leaving | distance | When near the path with direction away from the path. |
| Off Track | distance | When too far from the path. |



Note that threshold is Also applied to external fence defined by radius + distance

## Control Point triggers

|  |  |  |
| --- | --- | --- |
| trigger | variables | Description |
| Approaching | distance | When ON TRACK and near a **C**ontrol **P**oint with direction toward to the CP. |
| Check in |  | When ON TRACK and at **C**ontrol **P**oint. |
| Leaving | distance | When ON TRACK AND near a **C**ontrol **P**oint with direction away from the CP. |

## Properties

### evalThreshold

Type : number

Required : no

Default : 0

This is the delta use to evaluating a numeric condition. This avoid a condition from oscillating between two states.



For example, the threshold could be a radius of 500 meters where the evaluation threshold could be 50 meters in order to absorb GPS inaccuracy in weak signal area. huge evaluation threshold could lead in loss of alerts. This has to be adapted case by case.

Note : this functionality implies that the primitive MUST be stateful for each asset, so stateless implementation may choose to NOT support this property.

Note : This property is hold by the primitive in order to be applied to every related alert.

### inodes

Type : int[]

Required : no

An array of zero based index to the nodes defined into the extension.

### ialerts

Type : int[]

Required : no

An array of zero based index to the alerts defined into the extension.

### distance

Type : single

Required : no

Used by primitive, to trigger Approaching and Leaving events.