

# OHDM-API

## **Documentation**

Author: David Hadizadeh

E-Mail: david.hadizadeh@student.htw-berlin.de

Date: 05.03.2015

# Contents

<b>1</b>	<b>General information</b>	<b>1</b>
1.1	Java Example for a client . . . . .	1
1.2	Important note: Beta version . . . . .	2
<b>2</b>	<b>Availability and connection data</b>	<b>3</b>
2.1	Authentication . . . . .	3
<b>3</b>	<b>API calls</b>	<b>4</b>
3.1	External Sources . . . . .	4
3.1.1	Requesting an external source by a geographic object . . . . .	4
3.1.2	Requesting an external source by its ID . . . . .	6
3.1.3	Adding an external source . . . . .	7
3.2	Geographic Objects . . . . .	10
3.2.1	Requesting a geographic object by its ID . . . . .	10
3.2.2	Requesting nearby geographic objects by start and end date . . . . .	13
3.2.3	Requesting nearby geographic objects by start date . . . . .	16
3.2.4	Requesting nearby geographic objects by end date . . . . .	18
3.2.5	Requesting nearby geographic objects of all time . . . . .	20
3.2.6	Adding a geographic object . . . . .	22
3.2.7	Updating an existing geographic object with new data . . . . .	26
3.2.8	Removing an existing geographic object . . . . .	29
3.3	Geometric Objects . . . . .	31
3.3.1	Requesting a geometric object by its ID . . . . .	31
3.3.2	Requesting all geometric objects of a geographic object . . . . .	32
3.3.3	Adding a geometric object . . . . .	34
3.3.4	Updating an existing geometric object with new data . . . . .	36
3.3.5	Removing an existing geometric object . . . . .	38

## Contents

---

3.3.6	Removing all geometric objects of a geographic object . . . . .	39
3.4	Tag Dates . . . . .	41
3.4.1	Requesting a single tag date . . . . .	41
3.4.2	Requesting all tag dates of a geographic object . . . . .	43
3.4.3	Adding a tag date to a geographic object . . . . .	45
3.4.4	Updating the tag dates of a geographic object . . . . .	47
3.4.5	Removing a single tag date . . . . .	49
3.4.6	Removing all tag dates of a geographic object . . . . .	50

# 1 General information

The API accepts and response a JSON document (content type application/json). For every insert method the API provides the URL to the added element in the header. If you get a HTTP status code 500 the reason is mostly that the JSON format is wrong. Check out the examples in this documentation and compare them with your JSON string.

## 1.1 Java Example for a client

This is an example of how to request near geographic objects. This listing can be adapted easily for every other API call.

```
1 URL url = new URL("http://domain.com/OhdmApi/geographicObject/
  nearObjects/highway=primary,highway=traffic_signals/1/"+
  URLEncoder.encode("MULTILINESTRING( (13.513375234473285_
  52.42776656340422_0,13.513293936940071_52.42797876424454_0) )
  ", "UTF-8").replace("+", "%20"));
2 HttpURLConnection con = (HttpURLConnection) url.openConnection()
  ;
3 con.setRequestMethod("GET");
4 con.setRequestProperty("Content-Type",
5   "application/json;_charset=utf-8");
6
7 int responseCode = con.getResponseCode();
8 BufferedReader in = new BufferedReader(
9   new InputStreamReader(con.getInputStream()));
10 String inputLine;
11 StringBuffer response = new StringBuffer();
```

```
12
13 while ((inputLine = in.readLine()) != null) {
14     response.append(inputLine);
15 }
16 in.close();
17 System.out.println(response.toString());
```

## 1.2 Important note: Beta version

In this beta version the calls for getting near geographic objects are not finalized. Use them carefully and do not use big radius values else the API will take a lot of time to process these requests.

## **2 Availability and connection data**

URL (version 1.0) Test: <http://ohsm.f4.htw-berlin.de:8080/OhdmApi/>

URL (version 1.0) Productive: <http://ohdm.f4.htw-berlin.de:8080/OhdmApi/>

### **2.1 Authentication**

Not implemented yet.

## 3 API calls

### 3.1 External Sources

#### 3.1.1 Requesting an external source by a geographic object

Requests an external source with title and description by its geographic object ID.

##### HTTP-Method

GET

##### Resource

/externalSource/geographicObject/<geographicObjectId>

##### Parameters

- geographicObjectId: ID of the geographic object

## Response

```
1 {
2   "id":<ID of the external source>,
3   "title": "<title_of_the_source>",
4   "text": "<text_of_he_source>"
5 }
```

## HTTP status codes

- 200: Request executed successfully
- 404: No external source for given geographic object available
- 500: Error while reading the source

## Notes

- Use this call if you do not have the exact ID of the external source

## Exemplary Call

GET <http://domain.com/OhdmApi/externalSource/geographicObject/20>

## Exemplary Response

```
1 {
2   "id":1,
3   "title": "OHDM_Iphone_Editor",
4   "text": "Version_1.0_of_the_Iphone_Editor"
5 }
```



### 3.1.2 Requesting an external source by its ID

Requesting an external source by its automatically generated ID.

#### HTTP-Method

GET

#### Resource

/externalSource/<externalSourceId>

#### Parameters

- externalSourceId: ID of the external source

#### Response

```
1 {
2   "id":<ID of the external source>,
3   "title": "<title_of_the_source>",
4   "text": "<text_of_he_source>"
5 }
```

#### HTTP status codes

- 200: Request executed successfully
- 404: External source with the given ID not found
- 500: Error while reading the source

### Notes

- Use this call if you have the exact ID of the external source and want to get the title and description of it or you want to verify the ID.

### Exemplary Call

GET `http://domain.com/OhdmApi/externalSource/4`

### Exemplary Response

```
1 {
2   "id":1,
3   "title":"OHDM_Iphone_Editor",
4   "text":"Version_1.0_of_the_Iphone_Editor"
5 }
```

### 3.1.3 Adding an external source

Adding a new external source to OHDM. This can be sources like OSM, a book or an old map.

#### HTTP-Method

PUT

#### Resource

`/externalSource/`

## Payload

```
1 {
2   "title": "<title_of_the_source>",
3   "text": "<text_of_he_source>"
4 }
```

## Response

```
1 {
2   "key": <created key>
3 }
```

## HTTP status codes

- 201: External source created successfully
- 500: Error while adding the source

## Notes

- Editor developers should create an external source just once and use the received ID in their application.

## Exemplary Call

PUT <http://domain.com/OhdmApi/externalSource/>

```
1 {
2   "title": "OHDM_Iphone_Editor",
3   "text": "Version_1.0_of_the_Iphone_Editor"
4 }
```

## Exemplary Response

```
1 {  
2   "key":1  
3 }
```

## 3.2 Geographic Objects

### 3.2.1 Requesting a geographic object by its ID

Requests a geographic object and the geometries by its identifier.

#### HTTP-Method

GET

#### Resource

/geographicObject/<geographicObjectId>

#### Parameters

- geographicObjectId: ID of the geographic object

#### Response

```
1 {
2   "originalId":<originalID>,
3   "attributes":{
4     "<key>": "<value>"
5   },
6   "externalSourceId":<externalSourceId>,
7   "externalSource":{
8     "id":<ID of the external source>,
9     "<title_of_the_source>",
10    "text": "<text_of_he_source>"
11  },
12  "geoBlobDates":null,
```

```
13     "tagDates": [
14         {
15             "tagDateId": <tagDateId>,
16             "tags": {
17                 "<key>": "<value>"
18             },
19             "valid": {
20                 "since": "<dateText>",
21                 "until": "<dateText>"
22             }
23         }
24     ],
25     "geometricObjects": [
26         {
27             "geometricObjectId": <geometricObjectId>,
28             "valid": {
29                 "since": "<dateText>",
30                 "until": "<dateText>"
31             },
32             "multipoint": "<geometryText>",
33             "multilinestring": "<geometryText>",
34             "multipolygon": "<geometryText>",
35             "geometryCollection": "<geometryText>"
36         }
37     ]
38 }
```

### HTTP status codes

- 200: Request executed successfully
- 500: Error while requesting the geographic object
- 404: Geographic object not found

## Notes

- Use this method if you want to get all detailed data of a geographic object.

## Exemplary Call

GET <http://domain.com/OhdmApi/geographicObject/1>

## Exemplary Response

```
1 {
2   "geographicObjectId":1,
3   "attributes":{
4
5   },
6   "externalSourceId":0,
7   "externalSource":{
8     "id":0,
9     "title":"OSM_nodes",
10    "text":"OSM_geometries_with_the_id_nodes"
11  },
12  "originalId":59968148,
13  "geoBlobDates":null,
14  "tagDates":[
15    {
16      "tagDateId":35206,
17      "tags":{
18        "ref":"241",
19        "source:ref":"interpolation",
20        "source:position":"Yahoo",
21        "osm_timestamp":"2010-12-06T15:12:48Z",
22        "osm_user":"bahnpirat",
23        "power":"tower",
24        "osm_version":"6",
```

```
25         "osm_uid":"13203"
26     },
27     "valid":{
28         "since":"0001-01-01",
29         "until":"3000-01-01"
30     }
31 }
32 ],
33 "geometricObjects":[
34     {
35         "geometricObjectId":996410,
36         "valid":{
37             "since":"0001-01-01",
38             "until":"3000-01-01"
39         },
40         "multipoint":"SRID=4326;MULTIPOINT(12.907364398203306_
41             52.5968660947524_0) ",
42         "multilinestring":null,
43         "multipolygon":null,
44         "geometryCollection":null
45     }
46 ]
}
```

### 3.2.2 Requesting nearby geographic objects by start and end date

Requesting nearby geographic objects by a known distance and a geometry. It is also possible to transfer the needed key-value pairs which have to be included in the tag dates. This method provides to handle a start date and an end date of the geographic object. Objects which exist since the start date and until the end date will be returned.



## HTTP-Method

GET

## Resource

/geographicObject/nearObjects/<requiredKeys>/since/<since>/until/<until>/<distance>/<geometry>

## Parameters

- **requiredKeys:** A dynamic map which describes different keys and values. Values are optional and can be left open. This parameter determines which keys and values has to be included in the tagDates of the result. Keys and values are devided by ”=”. Each pair is splitted with ”,”.
- **since:** date string of the lower limit where the geographic object exists
- **until:** date string of the upper limit where the geographic object exists
- **distance:** radius from the given geometry to the required geographic objects
- **geometry:** geometry which describes the position where the geographic objects have to be located; can be any geometry (linestring, point, polygon etc.)

## Response

Array of geographic objects. Detailed structure is described in 3.2.1.

## HTTP status codes

- 200: Request executed successfully
- 500: Error while requesting the objects
- 404: There are no geographic objects for this filter

## Notes

- The `requiredKeys` parameter can be used for example to get all streets and to filter the geographic object by a type.
- The `distance` can be set to 1 if you want to get the objects in the given geometry with no more range.

## Exemplary Call

```
GET http://domain.com/OhdmApi/geographicObject/nearObjects/highway=
primary,highway=traffic_signals/since/0001-01-01/until/3000-01-
01/1/MULTILINESTRING((13.51337523447328552.427766563404220,13.
51329393694007152.427978764244540))
```

## Exemplary Response

```
1  [
2    {
3      "geographicObjectId":2177858,
4      "attributes":{
5
6      },
7      "externalSourceId":0,
8      "externalSource":{
9        "id":0,
10       "title":"OSM_nodes",
11       "text":"OSM_geometries_with_the_id_nodes"
12     },
13     "originalId":620874781,
14     "geoBlobDates":null,
15     "tagDates":[
16       {
17         "tagDateId":10166910,
```

```
18     "tags":{
19         "highway":"traffic_signals",
20         "osm_timestamp":"2012-09-09T12:37:42Z",
21         "osm_user":"Balgofil",
22         "osm_version":"3",
23         "osm_uid":"95702"
24     },
25     "valid":{
26         "since":"0001-01-01",
27         "until":"3000-01-01"
28     }
29 }
30 ],
31 "geometricObjects":[
32     {
33         "geometricObjectId":627278,
34         "valid":{
35             "since":"0001-01-01",
36             "until":"3000-01-01"
37         },
38         "multipoint":"SRID=4326;MULTIPOINT
39             (13.513375298118923_52.42776659475573_0)",
40         "multilinestring":null,
41         "multipolygon":null,
42         "geometryCollection":null
43     }
44 ]
45 ]
```

### 3.2.3 Requesting nearby geographic objects by start date

Requesting nearby geographic objects by a known distance and a geometry. It is also possible to transfer the needed key-value pairs which have to be included in the tag dates. This method provides to handle

a start date and an open end date of the geographic object. Objects which are existend since the start date will be returned.

### HTTP-Method

GET

### Resource

/geographicObject/nearObjects/<requiredKeys>/since/<since>/<distance>/<geometry>

### Parameters

- **requiredKeys:** A dynamic map which describes different keys and values. Values are optional and can be left open. This parameter determines which keys and values has to be included in the tagDates of the result. Keys and values are devided by "=" . Each pair is splitted with ",".
- **since:** date string of the lower limit where the geographic object exists
- **distance:** radius from the given geometry to the required geographic objects
- **geometry:** geometry which describes the position where the geographic objects have to be located; can be any geometry (linestring, point, polygon etc.)

### Response

Array of geographic objects. Detailed structure is described in 3.2.1.

### HTTP status codes

- 200: Request executed successfully
- 500: Error while requesting the objects
- 404: There are no geographic objects for this filter

## Notes

- The `requiredKeys` parameter can be used for example to get all streets and to filter the geographic object by a type.
- The `distance` can be set to 1 if you want to get the objects in the given geometry with no more range.

## Exemplary Call

```
GET http://domain.com/OhdmApi/geographicObject/nearObjects/  
highway=primary,highway=traffic_signals/since/0001-01-01/  
1/MULTILINESTRING((13.51337523447328552.427766563404220,13.  
51329393694007152.427978764244540))
```

## Exemplary Response

Same structure as in 3.2.2

### 3.2.4 Requesting nearby geographic objects by end date

Requesting nearby geographic objects by a known distance and a geometry. It is also possible to transfer the needed key-value pairs which have to be included in the tag dates. This method provides to handle an end date and an open start date of the geographic object. Objects which are existend until the end date will be returned.

## HTTP-Method

GET

## Resource

`/geographicObject/nearObjects/<requiredKeys>/until/<until>/<distance>/<geometry>`

### Parameters

- **requiredKeys:** A dynamic map which describes different keys and values. Values are optional and can be left open. This parameter determines which keys and values has to be included in the tagDates of the result. Keys and values are devided by "=" . Each pair is splitted with ",".
- **until:** date string of the upper limit where the geographic object exists
- **distance:** radius from the given geometry to the required geographic objects
- **geometry:** geometry which describes the position where the geographic objects have to be located; can be any geometry (linestring, point, polygon etc.)

### Response

Array of geographic objects. Detailed structure is described in 3.2.1.

### HTTP status codes

- 500: Error while requesting the objects
- 404: There are no geographic objects for this filter

### Notes

- The requiredKeys parameter can be used for example to get all streets and to filter the geographic object by a type.
- The distance can be set to 1 if you want to get the objects in the given geometry with no more range.

### Exemplary Call

```
GET http://domain.com/OhdmApi/geographicObject/nearObjects/  
highway=primary,highway=traffic_signals/until/3000-01-01/  
1/MULTILINESTRING((13.51337523447328552.427766563404220,13.  
51329393694007152.427978764244540))
```

### Exemplary Response

Same structure as in 3.2.2

## 3.2.5 Requesting nearby geographic objects of all time

Requesting nearby geographic objects by a known distance and a geometry. It is also possible to transfer the needed key-value pairs which have to be included in the tag dates. This method returns the near objects of all time.

### HTTP-Method

GET

### Resource

/geographicObject/nearObjects/<requiredKeys>/<distance>/<geometry>

### Parameters

- **requiredKeys:** A dynamic map which describes different keys and values. Values are optional and can be left open. This parameter determines which keys and values has to be included in the tagDates of the result. Keys and values are divided by "=". Each pair is splitted with ",".
- **distance:** radius from the given geometry to the required geographic objects

- geometry: geometry which describes the position where the geographic objects have to be located; can be any geometry (linestring, point, polygon etc.)

### Response

Array of geographic objects. Detailed structure is described in 3.2.1.

### HTTP status codes

- 200: Request executed successfully
- 500: Error while requesting the objects
- 404: There are no geographic objects for this filter

### Notes

- The `requiredKeys` parameter can be used for example to get all streets and to filter the geographic object by a type.
- The distance can be set to 1 if you want to get the objects in the given geometry with no more range.

### Exemplary Call

```
GET http://domain.com/OhdmApi/geographicObject/nearObjects/  
highway=primary,highway=traffic_signals/1/MULTILINESTRING((13.  
51337523447328552.427766563404220,13.51329393694007152.  
427978764244540))
```

### Exemplary Response

Same structure as in 3.2.2



### 3.2.6 Adding a geographic object

Adding a new geographic object with all geometries and meta data.

#### HTTP-Method

PUT

#### Resource

/geographicObject/

#### Payload

```
1 {
2   "originalId":<originalID>,
3   "attributes":{
4     "<key>":"<value>"
5   },
6   "externalSourceId":<externalSourceId>,
7   "geoBlobDates":[
8     {
9       "valid":{
10        "since":"<dateText>",
11        "until":"<dateText>"
12      }
13    }
14  ],
15  "tagDates":[
16    {
17      "tags":{
18        "<key>":"<value>"
19      },
```

```
20     "valid":{
21         "since": "<dateText>",
22         "until": "<dateText>"
23     }
24 }
25 ],
26 "geometricObjects": [
27     {
28         "valid":{
29             "since": "<dateText>",
30             "until": "<dateText>"
31         },
32         "multipoint": "<geometryText>",
33         "multilinestring": "<geometryText>",
34         "multipolygon": "<geometryText>",
35         "geometryCollection": "<geometryText>"
36     }
37 ]
38 }
```

## Response

```
1 {
2     "key":<created key>
3 }
```

## HTTP status codes

- 201: Successfully added
- 500: Error while adding the geographic object

## Notes

- originalID can be the ID of the object in OpenStreetMap
- multipoint, multilinestring, multipolygon and geometryCollection are optional. You can ignore the geometries which you do not want to use.
- tag dates are optional. If the geographic object does not have any tag dates you can ignore this parameter.

## Exemplary Call

PUT <http://domain.com/OhdmApi/geographicObject/>

```
1 {
2   "originalId": "12345",
3   "attributes": {
4     "key1": "value1",
5     "key2": "value2",
6     "key3": "value3"
7   },
8   "externalSourceId": 400,
9   "geoBlobDates": [
10    {
11      "valid": {
12        "since": "2013-12-01",
13        "until": "2014-05-14"
14      }
15    },
16    {
17      "valid": {
18        "since": "2013-10-01",
19        "until": "2014-05-11"
20      }
21    }
22  ]
23 }
```

```
22 ],
23   "tagDates": [
24     {
25       "tags": {
26         "key1": "value1",
27         "key2": "value2",
28         "key3": "value3"
29       },
30       "valid": {
31         "since": "2013-12-01",
32         "until": "2014-05-14"
33       }
34     }
35 ],
36   "geometricObjects": [
37     {
38       "valid": {
39         "since": "2013-12-01",
40         "until": "2014-05-14"
41       },
42       "multipoint": "MULTIPOINT(1_2,1_2) "
43     }
44 ]
45 }
```

### Exemplary Response

```
1 {
2   "key": 100
3 }
```

### 3.2.7 Updating an existing geographic object with new data

Updating an existing geographic object. Updates the external source ID, the original ID, the attributes, geometries and tag dates.

#### HTTP-Method

POST

#### Resource

/geographicObject/<geographicObjectId>

#### Parameters

- geographicObjectId: ID of the geographic object

#### Payload

```
1 {
2   "originalId":<originalID>,
3   "attributes":{
4     "<key>":"<value>"
5   },
6   "externalSourceId":<externalSourceId>,
7   "geoBlobDates":[
8     {
9       "valid":{
10        "since":"<dateText>",
11        "until":"<dateText>"
12      }
13    }
14  ],
```

```
15     "tagDates": [
16         {
17             "tags": {
18                 "<key>": "<value>"
19             },
20             "valid": {
21                 "since": "<dateText>",
22                 "until": "<dateText>"
23             }
24         }
25     ],
26     "geometricObjects": [
27         {
28             "valid": {
29                 "since": "<dateText>",
30                 "until": "<dateText>"
31             },
32             "multipoint": "<geometryText>",
33             "multilinestring": "<geometryText>",
34             "multipolygon": "<geometryText>",
35             "geometryCollection": "<geometryText>"
36         }
37     ]
38 }
```

## Response

Empty response. Details are included in the HTTP header.

## HTTP status codes

- 200: Update executed successfully
- 500: Error while processing the update

## Notes

- This call will not change the ID of the geographic object.
- Old geometries will be removed.
- Tag dates which are not in use anymore will be removed.

## Exemplary Call

POST <http://domain.com/OhdmApi/geographicObject/1>

```
1 {
2   "originalId": "12345",
3   "attributes": {
4     "key1": "value1",
5     "key2": "value2",
6     "key3": "value3"
7   },
8   "externalSourceId": 400,
9   "geoBlobDates": [
10    {
11      "valid": {
12        "since": "2013-12-01",
13        "until": "2014-05-14"
14      }
15    },
16    {
17      "valid": {
18        "since": "2013-10-01",
19        "until": "2014-05-11"
20      }
21    }
22  ],
23  "tagDates": [
```

```
24     {
25         "tags":{
26             "key1":"value1",
27             "key2":"value2",
28             "key3":"value3"
29         },
30         "valid":{
31             "since":"2013-12-01",
32             "until":"2014-05-14"
33         }
34     }
35 ],
36 "geometricObjects":[
37     {
38         "valid":{
39             "since":"2013-12-01",
40             "until":"2014-05-14"
41         },
42         "multipoint":"MULTIPOINT(1_2,1_2) "
43     }
44 ]
45 }
```

### 3.2.8 Removing an existing geographic object

Removing an geographic object and all assigned data.

#### HTTP-Method

DELETE



## Resource

/geographicObject/<geographicObjectId>

## Parameters

- geographicObjectId: ID of the geographic object

## Response

Empty response. Details are included in the HTTP header.

## HTTP status codes

- 200: Geographic object successfully removed
- 404: No object with the given ID found
- 500: Removing the object failed

## Notes

- This call will also remove the tag dates which are not in use anymore.

## Exemplary Call

DELETE <http://domain.com/OhdmApi/geographicObject/1>

## 3.3 Geometric Objects

### 3.3.1 Requesting a geometric object by its ID

Requesting a geometric object by its ID.

#### HTTP-Method

GET

#### Resource

/geometricObject/<geometricObjectId>

#### Parameters

- geometricObjectId: ID of the geometric object

#### Response

```
1 {
2   "valid": {
3     "since": "<dateText>",
4     "until": "<dateText>"
5   },
6   "multipoint": "<geometryText>",
7   "multilinestring": "<geometryText>",
8   "multipolygon": "<geometryText>",
9   "geometryCollection": "<geometryText>"
10 }
```

## HTTP status codes

- 200: Request executed successfully
- 404: Geometric object with the given ID not found
- 500: Error while processing the request

## Notes

- Each geometry is optional. Usually a geometric object has just one of them. For example just a multipoint and not a multilinestring.

## Exemplary Call

GET <http://domain.com/OhdmApi/geometricObject/1>

## Exemplary Response

```
1 {
2   "valid":{
3     "since":"2013-12-01",
4     "until":"2014-05-14"
5   },
6   "multipoint":"MULTIPOINT(1_2,1_2) "
7 }
```

### 3.3.2 Requesting all geometric objects of a geographic object

Requesting all geometric objects with geometries of a given geographic object ID.

## HTTP-Method

GET

## Resource

/geometricObject/geographicObject/<geographicObjectId>

## Parameters

- geographicObjectId: ID of the geographic object

## Response

```
1  [
2    {
3      "valid":{
4        "since":"<dateText>",
5        "until":"<dateText>"
6      },
7      "multipoint":"<geometryText>",
8      "multilinestring":"<geometryText>",
9      "multipolygon":"<geometryText>",
10     "geometryCollection":"<geometryText>"
11   }
12 ]
```

## HTTP status codes

- 200: Request executed successfully
- 404: No geometric objects found for the given geographic object ID

- 500: Error while processing the request

## Notes

- Use this call if you know the geographic object and want to get all of its geometries.

## Exemplary Call

GET <http://domain.com/OhdmApi/geometricObject/geographicObject/1>

## Exemplary Response

```
1  [  
2    {  
3      "valid":{  
4        "since":"2013-12-01",  
5        "until":"2014-05-14"  
6      },  
7      "multipoint":"MULTIPOINT (1_2,1_2) "  
8    }  
9  ]
```

### 3.3.3 Adding a geometric object

Adding a geometric object with multipoint, multilinestring, multipolygon and geometrycollection to a given geographic object.

#### HTTP-Method

POST

## Resource

/geometricObject/<geographicObjectId>

## Parameters

- geographicObjectId: ID of the geographic object where the geometry should be added

## Payload

```
1 {
2   "valid":{
3     "since":"<dateText>",
4     "until":"<dateText>"
5   },
6   "multipoint":"<geometryText>",
7   "multilinestring":"<geometryText>",
8   "multipolygon":"<geometryText>",
9   "geometryCollection":"<geometryText>"
10 }
```

## Response

```
1 {
2   "key":<created key>
3 }
```

## HTTP status codes

- 201: Successfully added
- 500: Error while adding the geometric object

## Notes

- Each geometry is optional. Usually a geometric object has just one of them. For example just a multipoint and not a multilinestring.

## Exemplary Call

POST <http://domain.com/OhdmApi/geometricObject/1>

```
1 {
2   "valid":{
3     "since":"2013-12-01",
4     "until":"2014-05-14"
5   },
6   "multipoint":"MULTIPOINT(1_2,1_2) "
7 }
```

## Exemplary Response

```
1 {
2   "key":100
3 }
```

### 3.3.4 Updating an existing geometric object with new data

Updating an existing geometric object. Updates all geometries and removes old linked geometries which are not longer in use.

#### HTTP-Method

POST

## Resource

/geometricObject/geometry/<geometricObjectId>

## Parameters

- `geometricObjectId`: ID of the geometric object which should be updated

## Payload

```
1 {
2   "valid": {
3     "since": "<dateText>",
4     "until": "<dateText>"
5   },
6   "multipoint": "<geometryText>",
7   "multilinestring": "<geometryText>",
8   "multipolygon": "<geometryText>",
9   "geometryCollection": "<geometryText>"
10 }
```

## Response

Empty response

## HTTP status codes

- 200: Update executed successfully
- 404: No geometric object found for the given ID
- 500: Error while processing the update



## Notes

- Each geometry is optional. Usually a geometric object has just one of them. For example just a multipoint and not a multilinestring.

## Exemplary Call

POST `http://domain.com/OhdmApi/geometricObject/geometry/1`

```
1 {
2   "valid": {
3     "since": "2013-12-01",
4     "until": "2014-05-14"
5   },
6   "multipoint": "MULTIPOINT(1_2,1_2) "
7 }
```

### 3.3.5 Removing an existing geometric object

Removing an existing geometric object and all assigned data.

#### HTTP-Method

DELETE

#### Resource

`/geometricObject/<geometricObjectId>`

#### Parameters

- `geometricObjectId`: ID of the geometric object which should be removed

## Response

Empty response

## HTTP status codes

- 200: Data removed successfully
- 500: Error while removing the data

## Exemplary Call

DELETE `http://domain.com/OhdmApi/geometricObject/1`

### 3.3.6 Removing all geometric objects of a geographic object

Removing all geometric objects of a given geographic object ID. This call will also remove all assigned geometries.

## HTTP-Method

DELETE

## Resource

`/geometricObject/geographicObject/<geographicObjectId>`

## Parameters

- `geographicObjectId`: ID of the geographic object which geometric objects should be removed

## Response

Empty response

## HTTP status codes

- 200: Data removed successfully
- 500: Error while removing the data

## Exemplary Call

```
DELETE http://domain.com/OhdmApi/geometricObject/geographicObject/  
1
```

## 3.4 Tag Dates

### 3.4.1 Requesting a single tag date

Requesting a tag date by its ID.

#### HTTP-Method

GET

#### Resource

/tagDate/<tagDateId>

#### Parameters

- tagDateId: ID of the tag date

#### Response

```
1 {
2   "tagDateId":<tagDateId>,
3   "tags":{
4     "<key>": "<value>"
5   },
6   "valid":{
7     "since": "<dateText>",
8     "until": "<dateText>"
9   }
10 }
```

## HTTP status codes

- 200: Request executed successfully
- 404: Tag date with this ID not found
- 500: Error while requesting the tag date

## Notes

- Use this call if you know the exact ID of a tag date

## Exemplary Call

GET <http://domain.com/OhdmApi/tagDate/35206>

## Exemplary Response

```
1 {
2   "tagDateId":35206,
3   "tags":{
4     "ref":"241",
5     "source:ref":"interpolation",
6     "source:position":"Yahoo",
7     "osm_timestamp":"2010-12-06T15:12:48Z",
8     "osm_user":"bahnpirat",
9     "power":"tower",
10    "osm_version":"6",
11    "osm_uid":"13203"
12  },
13  "valid":{
14    "since":"0001-01-01",
15    "until":"3000-01-01"
16  }
```

```
17 }
```

### 3.4.2 Requesting all tag dates of a geographic object

Requesting all tag dates of a geographic object by its ID.

#### HTTP-Method

GET

#### Resource

/tagDate/geographicObject<geographicObjectId>

#### Parameters

- geographicObjectId: ID of the geographic object

#### Response

```
1  [  
2    {  
3      "tagDateId":<tagDateId>,  
4      "tags":{  
5        "<key>": "<value>"  
6      },  
7      "valid":{  
8        "since": "<dateText>",  
9        "until": "<dateText>"  
10     }  
11  }
```

```
12 ]
```

## HTTP status codes

- 200: Request executed successfully
- 404: No tag dates found
- 500: Error while requesting the tag dates

## Exemplary Call

GET <http://domain.com/OhdmApi/tagDate/geographicObject/1>

## Exemplary Response

```
1  [
2    {
3      "tagDateId":35206,
4      "tags":{
5        "ref":"241",
6        "source:ref":"interpolation",
7        "source:position":"Yahoo",
8        "osm_timestamp":"2010-12-06T15:12:48Z",
9        "osm_user":"bahnpirat",
10       "power":"tower",
11       "osm_version":"6",
12       "osm_uid":"13203"
13     },
14     "valid":{
15       "since":"0001-01-01",
16       "until":"3000-01-01"
17     }
18   }
19 ]
```

```
18     }
19 ]
```

### 3.4.3 Adding a tag date to a geographic object

Adding a tag date to a given geographic object ID.

#### HTTP-Method

POST

#### Resource

/tagDate/<geographicObjectId>

#### Parameters

- geographicObjectId: ID of the geographic object where the tag date should be added

#### Payload

```
1 {
2   "tags":{
3     "<key>": "<value>"
4   },
5   "valid":{
6     "since": "<dateText>",
7     "until": "<dateText>"
8   }
9 }
```



## Response

```
1 {
2   "key":<created key>
3 }
```

## HTTP status codes

- 201: Tag date created successfully
- 404: Geographic object does not exist
- 500: Error while creating the tag date

## Notes

- You can find the URL to the created tag date in the response header.

## Exemplary Call

POST <http://domain.com/OhdmApi/tagDate/1>

```
1 {
2   "tags":{
3     "key1":"value1",
4     "key2":"value2",
5     "key3":"value3"
6   },
7   "valid":{
8     "since":"2013-12-01",
9     "until":"2014-05-14"
10  }
11 }
```

## Exemplary Response

```
1 {
2   "key":100
3 }
```

### 3.4.4 Updating the tag dates of a geographic object

Updating the tag dates of a geographic object by its ID.

#### HTTP-Method

POST

#### Resource

/tagDate/geographicObject<geographicObjectId>

#### Parameters

- geographicObjectId: ID of the geographic object

#### Payload

```
1 [
2   {
3     "tags":{
4       "<key>":"<value>"
5     },
6     "valid":{
7       "since":"<dateText>",
```

```
8     "until": "<dateText>"
9   }
10 }
11 ]
```

## Response

Empty response

## HTTP status codes

- 201: Tag dates updated successfully
- 404: Geographic object does not exist
- 500: Error while updating the tag dates

## Exemplary Call

POST <http://domain.com/OhdmApi/tagDate/geographicObject/1>

```
1 [
2   {
3     "tags": {
4       "key1": "value1",
5       "key2": "value2",
6       "key3": "value3"
7     },
8     "valid": {
9       "since": "2013-12-01",
10      "until": "2014-05-14"
11    }
12  }
13 ]
```

### 3.4.5 Removing a single tag date

Removing a single tag date by its ID

#### HTTP-Method

DELETE

#### Resource

/tagDate/<tagDateId>

#### Parameters

- tagDateId: ID of the tag date

#### Response

Empty response

#### HTTP status codes

- 200: Tag date removed successfully
- 500: Error while removing the tag date

#### Notes

- Use this method to remove a single tag date.

## Exemplary Call

DELETE `http://domain.com/OhdmApi/tagDate/1`

### 3.4.6 Removing all tag dates of a geographic object

Removing all tag dates of a geographic object by its ID

#### HTTP-Method

DELETE

#### Resource

`/tagDate/geographicObject/<geographicObjectId>`

#### Parameters

- `geographicObjectId`: ID of the geographic object

#### Response

Empty response

#### HTTP status codes

- 200: Tag dates removed successfully
- 500: Error while removing the tag dates

### Notes

- Use this method to remove all tag dates for a single geographic object. This can be usefull if you want to insert them tag by tag.

### Exemplary Call

DELETE `http://domain.com/OhdmApi/tagDate/geographicObject/1`