OHDM-API

Documentation

Author: David Hadizadeh
E-Mail: david.hadizadeh@student.htw-berlin.de
Date: 05.03.2015

Contents

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.setRequestProperty("Authorization", "Bearer␣" + token);
4  con.setRequestMethod("GET");
5  con.setRequestProperty("Content-Type",
6           "application/json;␣charset=utf-8");
7
8  int responseCode = con.getResponseCode();
9  BufferedReader in = new BufferedReader(
10       new InputStreamReader(con.getInputStream()));
11 String inputLine;
```

```
StringBuffer response = new StringBuffer();

while ((inputLine = in.readLine()) != null) {
   response.append(inputLine);
}
in.close();
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 1.1) Test: `http://ohsm.f4.htw-berlin.de:8080/OhdmApi/`

URL (version 1.1) Productive: `http://ohdm.f4.htw-berlin.de:8080/OhdmApi/`

2.1 Authentication

Authentication-Server:
Test: `http://ohsm.f4.htw-berlin.de/authentication/`
Productive: `http://ohdm.f4.htw-berlin.de/authentication/`

The OHDM-API uses Bearer tokens for authentication. You need to register an account at the authentication-server which you can find above (/register.php). After registration you have to login (/login.php). After logging in you will find an access token. This access token is used in the example of 1.1.

The line

```
1 con.setRequestProperty("Authorization", "Bearer " + token);
```

is responsible for the authorization.

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

```
{
   "id":<ID of the external source>,
   "title":"<title␣of␣the␣source>",
   "text":"<text␣of␣he␣source>"
}
```

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

```
{
   "id":1,
   "title":"OHDM␣Iphone␣Editor",
   "text":"Version␣1.0␣of␣the␣Iphone␣Editor"
}
```

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

```
{
   "id":<ID of the external source>,
   "title":"<title of the source>",
   "text":"<text of he source>"
}
```

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

```
{
   "id":1,
   "title":"OHDM Iphone Editor",
   "text":"Version 1.0 of the Iphone Editor"
}
```

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

```
{
    "title":"<title of the source>",
    "text":"<text of he source>"
}
```

Response

```
{
    "key":<created key>
}
```

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/`

```
{
    "title":"OHDM Iphone Editor",
    "text":"Version 1.0 of the Iphone Editor"
}
```

Exemplary Response

```
{
    "key":1
}
```

3.1.4 Removing an external source

Removing an external source.

HTTP-Method

DELETE

Resource

/externalSource/<externalSourceId>

Response

Empty response

HTTP status codes

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

Exemplary Call

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

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

```
{
   "originalId":<originalID>,
   "attributes":{
      "<key>":"<value>"
   },
   "externalSourceId":<externalSourceId>,
   "externalSource":{
      "id":<ID of the external source>,
      "<title of the source>",
      "text":"<text of he source>"
   },
   "geoBlobDates":null,
```

```
    "tagDates":[
        {
            "tagDateId":<tagDateId>,
            "tags":{
                "<key>":"<value>"
            },
            "valid":{
                "since":"<dateText>",
                "until":"<dateText>"
            }
        }
    ],
    "geometricObjects":[
        {
            "geometricObjectId":<geometricObjectId>,
            "valid":{
                "since":"<dateText>",
                "until":"<dateText>"
            },
            "multipoint":"<geometryText>",
            "multilinestring":"<geometryText>",
            "multipolygon":"<geometryText>"
        }
    ]
}
```

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


```
{
   "geographicObjectId":1,
   "attributes":{

   },
   "externalSourceId":0,
   "externalSource":{
      "id":0,
      "title":"OSM_nodes",
      "text":"OSM_geometries_with_the_id_nodes"
   },
   "originalId":59968148,
   "geoBlobDates":null,
   "tagDates":[
      {
         "tagDateId":35206,
         "tags":{
            "ref":"241",
            "source:ref":"interpolation",
            "source:position":"Yahoo",
            "osm_timestamp":"2010-12-06T15:12:48Z",
            "osm_user":"bahnpirat",
            "power":"tower",
            "osm_version":"6",
```

```
            "osm_uid":"13203"
         },
         "valid":{
            "since":"0001-01-01",
            "until":"3000-01-01"
         }
      }
   ],
   "geometricObjects":[
      {
         "geometricObjectId":996410,
         "valid":{
            "since":"0001-01-01",
            "until":"3000-01-01"
         },
         "multipoint":"SRID=4326;MULTIPOINT(12.907364398203306␣
            52.5968660947524␣0)",
         "multilinestring":null,
         "multipolygon":null
      }
   ]
}
```


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>/<longitude>/<latitude>

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 selected point (lng, lat) to the required geographic objects
- longitude: position where the geographic objects have to be located
- latitude: position where the geographic objects have to be located

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 should usually be set to 1. This will create a point with a radius of 1 meter. Every polygon which includes the point and every linestring or point which intersect the point will be delivered. They will be sorted by their size.

Exemplary Call

GET `http://domain.com/OhdmApi/geographicObject/nearObjects/highway=primary,highway=traffic_signals/since/0001-01-01/until/3000-01-01/1/13.513375234473285/52.42776656340422`

Exemplary Response


```
[
   {
      "geographicObjectId":2177858,
      "attributes":{

      },
      "externalSourceId":0,
      "externalSource":{
         "id":0,
         "title":"OSM_nodes",
         "text":"OSM geometries with the id nodes"
      },
      "originalId":620874781,
      "geoBlobDates":null,
      "tagDates":[
         {
            "tagDateId":10166910,
```

```
            "tags":{
               "highway":"traffic_signals",
               "osm_timestamp":"2012-09-09T12:37:42Z",
               "osm_user":"Balgofil",
               "osm_version":"3",
               "osm_uid":"95702"
            },
            "valid":{
               "since":"0001-01-01",
               "until":"3000-01-01"
            }
         }
      ],
      "geometricObjects":[
         {
            "geometricObjectId":627278,
            "valid":{
               "since":"0001-01-01",
               "until":"3000-01-01"
            },
            "multipoint":"SRID=4326;MULTIPOINT
               (13.513375298118923␣52.42776659475573␣0)",
            "multilinestring":null,
            "multipolygon":null
         }
      ]
   }
]
```


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 exist since the start date will be returned.

HTTP-Method

GET

Resource

/geographicObject/nearObjects/<requiredKeys>/since/<since>/<distance>/<longitude>/<latitude>

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 selected point (lng, lat) to the required geographic objects
- longitude: position where the geographic objects have to be located
- latitude: position where the geographic objects have to be located

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 should usually be set to 1. This will create a point with a radius of 1 meter. Every polygon which includes the point and every linestring or point which intersect the point will be delivered. They will be sorted by their size.

Exemplary Call

GET `http://domain.com/OhdmApi/geographicObject/nearObjects/highway=primary,highway=traffic_signals/since/0001-01-01/1/13.513375234473285/52.42776656340422`

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 exist until the end date will be returned.

HTTP-Method

GET

Resource

/geographicObject/nearObjects/<requiredKeys>/until/<until>/<distance>/<longitude>/<latitude>

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 selected point (lng, lat) to the required geographic objects
- longitude: position where the geographic objects have to be located
- latitude: position where the geographic objects have to be located

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 should usually be set to 1. This will create a point with a radius of 1 meter. Every polygon which includes the point and every linestring or point which intersect the point will be delivered. They will be sorted by their size.

Exemplary Call

GET `http://domain.com/OhdmApi/geographicObject/nearObjects/highway=primary,highway=traffic_signals/until/3000-01-01/1/13.513375234473285/52.42776656340422`

Exemplary Response

Same structure as in 3.2.2

3.2.5 Requesting nearby geographic objects of any 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>/<longitude>/<latitude>

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 ”,”.
- distance: radius from the selected point (lng, lat) to the required geographic objects
- longitude: position where the geographic objects have to be located

- latitude: position where the geographic objects have to be located

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 should usually be set to 1. This will create a point with a radius of 1 meter. Every polygon which includes the point and every linestring or point which intersect the point will be delivered. They will be sorted by their size.

Exemplary Call

GET `http://domain.com/OhdmApi/geographicObject/nearObjects/highway=primary,highway=traffic_signals/1/13.513375234473285/52.42776656340422`

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

```
{
   "originalId":<originalID>,
   "attributes":{
      "<key>":"<value>"
   },
   "externalSourceId":<externalSourceId>,
   "geoBlobDates":[
      {
         "valid":{
            "since":"<dateText>",
            "until":"<dateText>"
         }
      }
   ],
   "tagDates":[
      {
         "tags":{
            "<key>":"<value>"
         },
```

```
            "valid":{
               "since":"<dateText>",
               "until":"<dateText>"
            }
         }
      ],
      "geometricObjects":[
         {
            "valid":{
               "since":"<dateText>",
               "until":"<dateText>"
            },
            "multipoint":"<geometryText>",
            "multilinestring":"<geometryText>",
            "multipolygon":"<geometryText>"
         }
      ]
}
```

Response

```
{
   "key":<created key>
}
```

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 and multipolygon 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/`

```
{
   "originalId":"12345",
   "attributes":{
      "key1":"value1",
      "key2":"value2",
      "key3":"value3"
   },
   "externalSourceId":400,
   "geoBlobDates":[
      {
         "valid":{
            "since":"2013-12-01",
            "until":"2014-05-14"
         }
      },
      {
         "valid":{
            "since":"2013-10-01",
            "until":"2014-05-11"
         }
      }
```

```
    ],
    "tagDates":[
        {
            "tags":{
                "key1":"value1",
                "key2":"value2",
                "key3":"value3"
            },
            "valid":{
                "since":"2013-12-01",
                "until":"2014-05-14"
            }
        }
    ],
    "geometricObjects":[
        {
            "valid":{
                "since":"2013-12-01",
                "until":"2014-05-14"
            },
            "multipoint":"MULTIPOINT(1␣2,1␣2)"
        }
    ]
}
```

Exemplary Response

```
{
    "key":100
}
```

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

```
{
   "originalId":<originalID>,
   "attributes":{
      "<key>":"<value>"
   },
   "externalSourceId":<externalSourceId>,
   "geoBlobDates":[
      {
         "valid":{
            "since":"<dateText>",
            "until":"<dateText>"
         }
      }
   ],
```

```
    "tagDates":[
        {
            "tags":{
                "<key>":"<value>"
            },
            "valid":{
                "since":"<dateText>",
                "until":"<dateText>"
            }
        }
    ],
    "geometricObjects":[
        {
            "valid":{
                "since":"<dateText>",
                "until":"<dateText>"
            },
            "multipoint":"<geometryText>",
            "multilinestring":"<geometryText>",
            "multipolygon":"<geometryText>"
        }
    ]
}
```

Response

Empty response. Details are included in the HTTP header.

HTTP status codes

- 200: Update executed successfully successfull
- 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`

```
{
   "originalId":"12345",
   "attributes":{
      "key1":"value1",
      "key2":"value2",
      "key3":"value3"
   },
   "externalSourceId":400,
   "geoBlobDates":[
      {
         "valid":{
            "since":"2013-12-01",
            "until":"2014-05-14"
         }
      },
      {
         "valid":{
            "since":"2013-10-01",
            "until":"2014-05-11"
         }
      }
   ],
   "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

```
{
   "valid":{
      "since":"<dateText>",
      "until":"<dateText>"
   },
   "multipoint":"<geometryText>",
   "multilinestring":"<geometryText>",
   "multipolygon":"<geometryText>"
}
```

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

```
{
   "valid":{
      "since":"2013-12-01",
      "until":"2014-05-14"
   },
   "multipoint":"MULTIPOINT(1␣2,1␣2)"
}
```

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

```
[
    {
        "valid":{
            "since":"<dateText>",
            "until":"<dateText>"
        },
        "multipoint":"<geometryText>",
        "multilinestring":"<geometryText>",
        "multipolygon":"<geometryText>"
    }
]
```

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

```
[
    {
        "valid":{
            "since":"2013-12-01",
            "until":"2014-05-14"
        },
        "multipoint":"MULTIPOINT(1 2,1 2)"
    }
]
```

3.3.3 Adding a geometric object

Adding a geometric object with multipoint, multilinestring and multipolygon 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

```
{
   "valid":{
      "since":"<dateText>",
      "until":"<dateText>"
   },
   "multipoint":"<geometryText>",
   "multilinestring":"<geometryText>",
   "multipolygon":"<geometryText>"
}
```

Response

```
{
   "key":<created key>
}
```

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`

```
{
    "valid":{
        "since":"2013-12-01",
        "until":"2014-05-14"
    },
    "multipoint":"MULTIPOINT(1␣2,1␣2)"
}
```

Exemplary Response

```
{
    "key":100
}
```

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

```
{
   "valid":{
      "since":"<dateText>",
      "until":"<dateText>"
   },
   "multipoint":"<geometryText>",
   "multilinestring":"<geometryText>",
   "multipolygon":"<geometryText>"
}
```

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`

```
{
    "valid":{
        "since":"2013-12-01",
        "until":"2014-05-14"
    },
    "multipoint":"MULTIPOINT(1 2,1 2)"
}
```

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

```
{
    "tagDateId":<tagDateId>,
    "tags":{
        "<key>":"<value>"
    },
    "valid":{
        "since":"<dateText>",
        "until":"<dateText>"
    }
}
```

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


```
{
   "tagDateId":35206,
   "tags":{
      "ref":"241",
      "source:ref":"interpolation",
      "source:position":"Yahoo",
      "osm_timestamp":"2010-12-06T15:12:48Z",
      "osm_user":"bahnpirat",
      "power":"tower",
      "osm_version":"6",
      "osm_uid":"13203"
   },
   "valid":{
      "since":"0001-01-01",
      "until":"3000-01-01"
   }
```

```
}
```

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

```
[
    {
        "tagDateId":<tagDateId>,
        "tags":{
            "<key>":"<value>"
        },
        "valid":{
            "since":"<dateText>",
            "until":"<dateText>"
        }
    }
```

```
]
```

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


```
[
   {
      "tagDateId":35206,
      "tags":{
         "ref":"241",
         "source:ref":"interpolation",
         "source:position":"Yahoo",
         "osm_timestamp":"2010-12-06T15:12:48Z",
         "osm_user":"bahnpirat",
         "power":"tower",
         "osm_version":"6",
         "osm_uid":"13203"
      },
      "valid":{
         "since":"0001-01-01",
         "until":"3000-01-01"
      }
```

```
    }
]
```

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

```
{
    "tags":{
        "<key>":"<value>"
    },
    "valid":{
        "since":"<dateText>",
        "until":"<dateText>"
    }
}
```

Response

```
{
   "keys":[<created keys>]
}
```

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`

```
{
   "tags":{
      "key1":"value1",
      "key2":"value2",
      "key3":"value3"
   },
   "valid":{
      "since":"2013-12-01",
      "until":"2014-05-14"
   }
}
```

Exemplary Response

```
{
    "keys":[100,101,101]
}
```

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

```
[
    {
        "tags":{
            "<key>":"<value>"
        },
        "valid":{
            "since":"<dateText>",
```

```
            "until":"<dateText>"
        }
    }
]
```

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`

```
[
    {
        "tags":{
            "key1":"value1",
            "key2":"value2",
            "key3":"value3"
        },
        "valid":{
            "since":"2013-12-01",
            "until":"2014-05-14"
        }
    }
]
```

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`