Documentation OBELISdeutschlandnetz

Last Updated: Oct 15, 2024

General Information

The OBELISdeutschlandnetz API is organized around REST. Our API has predictable resource-oriented URLs, accepts JSON-encoded request bodies, returns JSON-encoded responses, and uses standard HTTP response codes, authentication, and verbs.

You can use the API in test mode, which doesn't affect your live data. The API URL and Client Credentials you use to authenticate the request determines whether the request is live mode or test mode.

The OBELISdeutschlandnetz API doesn't support bulk updates. You can work on only one object per request.

Authentication

The OBELISdeutschlandnetz API uses machine-to-machine authentication using the OAuth 2.0-Protocol to authenticate and authorize requests.
In order to obtain your client-token please get in contact with our team via e-mail.
Afterwards you will receive an e-mail containing the client credentials for your backend-application.
These credentials carry privileges to act on your behalf, so be sure to keep them secure. Don't share your secret API keys in publicly accessible areas such as public GitHub Repositories.

OBELIS Team noreply@obelis-deutschlandnetz.de via amazonses.com
to me

Guten Tag,

bitte verwenden Sie folgende Zugangsdaten für die Authentifizierung in der Test-Umgebung des OBELISdeutschlandnetz:

ClientId:
ClientSecret:
CPO-ID:

Mit freundlichen Grüßen
Ihr OBELIS-Team

OBELISdeutschlandnetz ist eine Online-Plattform der Nationalen Leitstelle Ladeinfrastruktur unter dem Dach der NOW GmbH
Nationale Leitstelle Ladeinfrastruktur
c/o NOW GmbH
Nationale Organisation Wasserstoff- und Brennstoffzellentechnologie
Fasanenstr. 5
10623 Berlin
Email: obelis@now-gmbh.de
Internet: www.now-gmbh.de | www.nationale-leitstelle.de

Example Registration Email

Authentication to the API is performed by requesting an access token inside your application.

Example Curl-Request to obtain a token:

```
curl --location 'TOKEN_ENDPOINT' \
--header 'Content-Type: application/json' \
```

```
--data '{
    "client_id": "YOUR_CLIENT_ID",
    "client_secret": "YOUR_CLIENT_SECRET",
    "cpo_id": "YOUR_CPO_ID"
}'
```

The token-endpoint answers with a jwt access token:

```
{
    "token": "YOUR_ACCESS_TOKEN",
    "expires_in": 3600,
    "token_type": "Bearer"
}
```

Your application then needs to extract the token and send it as a bearer token to the endpoints it wants to access. Example Curl-Request to an endpoint:

```
curl --location '{endpoint}' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer YOUR_ACCESS_TOKEN'
```

All access tokens are valid for one hour. The token itself contains the expiration time, which can manually be checked via jwt.io or automatically using a jwt-package in your preferred programming language. After the token has expired the application needs to request a new token.

PAYLOAD: DATA


```
{
  "sub": "6g9da9bnarkaa6o15bc36ac6vv",
  "token_use": "access",
  "scope": "transactions/obelis_offentlich",
  "auth_time": 1728293931,
  "iss": "https://cognito-idp.eu-central-
1.amazonaws.com/eu-central-1_KNrpR8omj",
  "exp": 1728297531,
  "iat": 1728293931,
  "version": 2,
  "jti": "38032faa-7d39-47d2-a8cf-7408f3f23258",
  "client_id": "6g9da9bnarkaa6o15bc36ac6vv"
}
```


You must make all API calls over HTTPS. Calls that you make over plain HTTP will fail.
API requests without authentication will also fail.

Errors

OBELISdeutschlandnetz uses conventional HTTP response codes to indicate the success or failure of an API request. In general: Codes in the `2xx` range indicate success. Codes in the `4xx` range indicate an error that failed given the information provided (e.g., a required parameter was omitted, a charge failed, etc.). Codes in the `5xx` range indicate an error with OBELISdeutschlandnetz's servers (these are rare).

Some `4xx` errors that could be handled programmatically (e.g., a date field is invalid) include an error code that briefly explains the error reported.

HTTP Status Code Summary

Status Code	Description
200 - OK	Everything worked as expected.
400 - Bad Request	The request was unacceptable, often due to missing a required parameter.
401 - Unauthorized	No valid OAuth 2.0 token provided.
403 - Forbidden	The token doesn't have permissions to perform the request.
404 - Not Found	The requested resource doesn't exist.
409 - Already Exists	The resource with specific parameters already exists
429 - Too Many Requests	Too many requests hit the API too quickly. We recommend an exponential backoff of your requests.
500, 502, 503, 504 - Server Errors	Something went wrong on OBELISdeutschlandnetz's end. (These are rare.)

Example Error (statusCode is optional):

```
{"statusCode": 400, "message": "coordinates are already assigned to a charging hub"}
```

Rate Limiting

The OBELISdeutschlandnetz API uses various safeguards against bursts of incoming traffic to help maximize its stability. Applications that send too many requests in quick succession might see error responses that show up as status code `429`. We have an **AWS Rate Limit WAF Rule** in place that limits the number of requests received by the API within any given second:

OBELISdeutschlandnetz allows up to 2.000 operations per minute.

Treat this limit as maximum and don’t generate unnecessary load. See Handling limiting gracefully for advice on handling 429s. If you suddenly see a rising number of rate limited requests, please contact support.

We may reduce limits to prevent abuse, or increase limits to enable high-traffic applications.

Common causes and mitigations

Rate limiting can occur under a variety of conditions, but it’s most common in these scenarios:

- Running a large volume of closely-spaced requests can lead to rate limiting. Often this is part of an analytical or migration operation. When engaging in these activities, you should try to control the request rate on the application side (see Handling limiting gracefully).
- A sudden increase in charge volume might result in rate limiting. We try to set our rates high enough that the vast majority of users are never rate limited for legitimate traffic, but if you suspect that an upcoming event might push you over the limits listed above, please contact support.

Handling limiting gracefully

A basic technique for integrations to gracefully handle limiting is to watch for `429` status codes and build in a retry mechanism. The retry mechanism should follow an exponential backoff schedule to reduce request volume when necessary. We’d also recommend building some randomness into the backoff schedule to avoid a thundering herd effect.

You can only optimize individual requests to a limited degree, so an even more sophisticated approach would be to control traffic to OBELISdeutschlandnetz at a global level, and throttle it back if you detect substantial rate limiting. A common technique for controlling rate

is to implement something like a token bucket rate limiting algorithm on the application-side. Ready-made and mature implementations for token bucket are available in almost any programming language.

Load testing

It's common for users to prepare for a major event by load testing their systems, with the OBELISdeutschlandnetz API as part of it. We generally discourage this practice because API limits are lower in test mode, so the load test is likely to hit limits that it wouldn't hit in production. Test mode is also not a perfect stand-in for live API calls, and that can be somewhat misleading. For example, requests to our test environment might result in significantly different latency profiles than requests to our live environment.

As an alternative, we recommend building integrations so that they have a configurable system for mocking out requests to the OBELISdeutschlandnetz API, which can be enabled for load tests. For realistic results, they should simulate latency by sleeping for a time that you determine by sampling the durations of real API calls, as seen from the perspective of the integration.

API Documentation

We kindly request you to thoroughly review the Swagger or OpenAPI YAML descriptions of the OBELISdeutschlandnetz API before making a personal inquiry.

These two technical documentations are continuously updated and should be considered definitive, meaning that functions and query options not listed there are currently (still) unavailable.

Please also understand that information concerning aspects that impact the overall stability and availability of the application and the API (such as firewall configuration), which goes beyond the information available in this document, cannot be disclosed or communicated upon request.

Using the OBELISdeutschlandnetz API

The OBELISdeutschlandnetz API allows you to access and interact with various resources. In this section, you will learn how to identify, create, store, update, and delete generated resources.

Creating Resources

To create a new resource, use the HTTP verb `POST`. Make sure to provide the required data and fields in your request body. The API will then provide you with a unique identification (ID) for the created resource.
Here is an example of creating a charging hub:

```
POST CHARGINGHUB_ENDPOINT
Content-Type: application/json
Authorization: Bearer YOUR_ACCESS_TOKEN

{
    "name": "Charging Hub",
    "address": "Dresdner Str. 136",
    "postalCode": "01640",
    "city": "Coswig",
    "state": "SN",
    "coordinates": {
        "latitude": 51.128490256558,
        "longitude": 13.55894958733462
    },
    "operator": "DEFAS",
    "owner": "Allego GmbH",
    "negligibleDamages": false,
    "lastUpdated": "2024-02-22T11:39:56.501Z"
}
```

In this request, you create a new charging hub with the provided data in the fields.
The API will respond with an ID for this new resource, which you can use for future interactions.


```
{
   "data":{
      "id":"191e7db7-1bea-44db-9e52-2ddab118d699",
      "name":"Charging Hub",
      "address":"Dresdner Str. 136",
      "postalCode":"01640",
      "city":"Coswig",
      "state":"SN",
      "coordinates":{
         "latitude":51.128490256558,
         "longitude":13.55894958733462
      },
      "operator":"DEFAS",
      "owner":"Allego GmbH",
      "negligibleDamages":false,
      "lastUpdated":"2024-02-22T11:39:56.501Z"
   },
   "timestamp":"2024-10-04T13:13:52.496Z",
   "message":"Request was processed successfully",
   "statusCode":200
}
```


In the OBELISdeutschlandnetz API, resources are a central component.
Each resource is represented by a unique identifier (ID) assigned to it during creation. This identifier is crucial for accessing, updating, or deleting the resource.
Please ensure that you carefully store the generated identifier, as it is required for the unique identification of the resource.

Updating Resources

To update an existing resource, use the following HTTP verbs, depending on your update requirements:

- **PUT (Full Update)**:
  When using PUT, you replace the entire resource with the new data provided in the request. You must send the complete representation of the resource, including all fields, in your update request. This completely replaces the existing resource with the new data.

```
PUT https://qa-api.obelis-deutschlandnetz.de/resource/{RESOURCE_ID}
Content-Type: application/json
Authorization: Bearer YOUR_ACCESS_TOKEN

{
    "field1": "new_value1",
    "field2": "new_value2"
}
```

In this pseudo request, the resource with ID `{RESOURCE_ID}` is updated and completely replaced. Please note that you must send the entire resource content in your update request. This means you need to send all required data along with the information you want to update. If you only want to update specific parts of the resource, you should use the `PATCH` verb and only send the relevant fields.

- **PATCH (Partial Update)**:
  When using `PATCH`, you perform partial updates to the resource. You only need to send the fields you want to update along with their new values. This allows for more precise and partial updates to the resource.

```
PATCH https://qa-api.obelis-deutschlandnetz.de/resource/{RESOURCE_ID}
```

```
Content-Type: application/json
Authorization: Bearer YOUR_ACCESS_TOKEN

{
    "field1": "new_value1"
}
```

In this request, only the field `field1` of the resource with ID `{RESOURCE_ID}` is updated to the new value `new_value1`, while all other fields remain unchanged.

Please consult the Swagger or OpenAPI documentation to determine which verbs are supported for a specific endpoint. This is important because not all endpoints necessarily support both verbs.

Updating Resources with Sub-Entities

> ⚠️ **Caution** ⚠️
>
> When updating resources that contain arrays of sub-entities, you should be careful. If these sub-entities do not have unique identifiers, and you send an update to this field with only one entity, all other items in the array will be deleted.

For illustration:

If you want to update the connectors of a charging point and you send only a new connector to the endpoint without mentioning the existing connectors, all existing connectors will be replaced by the new connector. To update individual connectors, you must send all current connectors along with the updated connector to ensure no data is lost.

Make sure to carefully review the API documentation and the structure of the resource to ensure you send the required data correctly when updating resources with complex structures.

If you have further questions about updating resources with complex structures, please feel free to contact our support team. We are here to assist you with your questions and provide support.

Deleting Resources

To delete a resource, use the HTTP verb `DELETE` and specify the identifier of the resource to be deleted in the endpoint.

```
DELETE https://qa-api.obelis-deutschlandnetz.de/resource/{RESOURCE_ID}
Authorization: Bearer YOUR_ACCESS_TOKEN
```

This request removes the resource with the ID `{RESOURCE_ID}` from the system.
Please act carefully with deletion operations, as deleted resources cannot be restored.

Request for recently modified Entities

An entity is considered modified when either the entity itself or a directly related sub-entity has been changed (e.h. charging points and connectors). However, an update does not necessarily mean that the fields of an entity have actually already changed.
If you intend to update the database regularly or at very short intervals, please note that there might be a time delay of approximately 15 minutes before a change is visible in the API. In the meantime, it is possible that the API may respond with a 404 error.

Monthly Reports

Please note that creating monthly reports (ratings) is only possible for the previous month. This means that you can only create reports for data from the previous month. Make sure your requests take this time constraint into account. This validation applies to the following endpoints:

- */ocpi/bob/2.2/debob-reports/hotline-ratings*

- */ocpi/bob/2.2/debob-reports/nondiscriminatory-access*
- */ocpi/bob/2.2/debob-reports/charginghub/{charginghubId}/ratings*
- */ocpi/bob/2.2/debob-reports/charginghub/{charginghubId}/ad-hoc-tariff*
- */ocpi/bob/2.2/debob-reports/charginghub/{charginghubId}/chargingpoint/{chargingpointId}/ratings*