Facilitor HELP

Online Facilitor Documentatie

General API usage

All services of this Facilitor-API can be used with either JSON or XML formatted content.The result of a GET request can be used for a PUT/POST request, the structures used are the same. All references to other entities are provided by a sub-“node” with 2 items:

- id: Numeric value (max. 10 digits) and unique identification of entity it defines (eg. building, appointment, reservableroom, reservation, etc.).
- name: The human readable presentation of that entitiy, like a code or a name

The id could be used for a subsequent call to get more details of the referred entity.

Invoke the API

In general request will look like this:
`GET https://XXXX.facilitor.nl/api2/<xxx>.json`
and
`GET https://XXXX.facilitor.nl/api2/<xxx>/<id>.json`
where <xxx> is the name of the entity involved eg. ‘buildings’.

Limited view of data

The response of a GET will always be restricted to data to which the user associated with the API-key (the *API-user* for short) has access to. The API-user must be granted access and authorizations by the application managers in the same way regular users are authorized. This is also true for the PUT, POST and DELETE requests.

Fetching associated data

You have to explicitly specify the associations you want to be included in the query result by appending the include parameter to the query url. For example when requesting an appointment.json the url can be extended with &include=visitors to include visitor information as well.

- Adding includes on a POST without using the include parameter will also result in adding the included data. The response will show the included node(s).

- Adding includes on a PUT without using the include parameter will also result in changing the included data. The response will show the included node(s).

Ranges

With the GET-services filters can be added to the request, like ?location=2 to filter building on a specific location. Currently three types of filters are supported:

Exact: filter must be exact, e.g. &name=ENS4 will get you the data of the building named ENS4

Like: filter acts like a wildcard, e.g. &name=ENS will yield all building with ENS in the name

Range: (currently for dates only) you may use the start_ and end_ filters. If you just filter on the property, it will behave like you used start_. E.g. ..issues.json?start_issuedate=XXX&end_issuedate=YYY will give you all issues that are issued between XXX (0:00) and YYY (23:59), inclusive, and ..issues.json?issuedate=XXX will give you the issues starting with XXX (0:00).

Dates

All dates are used by the ISO standard format, e.g. 2014-08-31T22:00:00Z

Authentication

For authorization an API-key is provided, for example 'iXsooFqjhXgezGQpXOLXZFkHDzYofRQJm'. This API key is unique for each application accessing a Facilitor environment.

The API key may be attached to each request in one of the following way:

- Passed in as a "X- Facilitor -API-Key" HTTP header (preferred)
- Passed in as a username with a random password via HTTP Basic authentication
- If that is not possible for the calling application provisions can be made to pass the API-key in the querystring as &apikey=iXsooFqjhXezGQpXOLXZFkHDzYofRQJm

Response

Response codes

- 200: OK
- 204: No Content
- 304: Not modified
- 401: Unauthorized
- 404: Not Found
- 405: Method not allowed

- 412: Invalid X-Facilitor-Switch-User header
- 501: Not implemented

Error response syntax

The above response codes can be returned with or without result-body. Without a result -body the response code speaks for itself (eg. No Content or Not Found). Otherwise the result-body contains the response as documented for the concerned entity (in case of 200). There are two exceptions:

A 200-response code in combination with a general failure:

```
{
"success":false,
"message":"XXX"
}
```

A specific error response (usually 500) of which most should never occur in production (after being tested). All error responses of this sort, but most important the ones that can occur have a result-body which contains a user-friendly message:

```
{
"error": {
"code": 500,
"message": "XXX"
}
}
```

Examples:
General failure

```
{
"success":false,
"message":"Op grond van de u toegekende rechten heeft u hiervoor geen bevoegdheid.\n\nIndien u van mening bent dat dit niet terecht is kunt u het beste contact opnemen met uw Facilitor contactpersoon."
}
```

Specific error responses

```
{
"error":
{
"code": 500,
"message": " Deze ruimte of voorziening is niet beschikbaar. Kies een andere ruimte of een ander tijdstip!"
}
}
```

```
{
"error": {
"code": 500,
"message": "Aiai\nEr is iets fout gegaan, maar dat ligt waarschijnlijk niet aan u.\nDeze fout wordt automatisch geregistreerd en onderzocht.\nU kunt wellicht het beste uw browser verversen en verder gaan met uw werk.\nFacilitor 2014-09-08 12:06:54, https://xxxx.facilitorlabs.nl"
}
}
{
"error": {
"code": 500,
"message": "Error parsing input JSON: Expected identifier, string or number"
}
}
{
"error": {
"code": 500,
"message": "No 'reservation' found in input"
}
}
```