



## Developer guide

v.0.9.4 – 2 Feb 2015

<b>1. INTRODUCTION</b>	<b>3</b>
<b>2. AUTHENTICATING</b>	<b>3</b>
<b>3. ENVIRONMENTS</b>	<b>3</b>
<b>4. MAKING REQUESTS</b>	<b>3</b>
<b>5. API ACTIONS WITHIN THE BREATHEHR SYSTEM</b>	<b>4</b>
<b>6. REQUEST AND RESPONSE FORMAT</b>	<b>5</b>
6.1 SPECIFYING THE PARAMETERS (HTTP OR JSON)	6
6.2 DATE AND TIME FORMATS	6
6.3 NESTED OBJECTS	6
6.4 THE ROOT OBJECT	7
6.5 PAGINATION	8
<b>7. ERROR RESPONSES</b>	<b>8</b>
<b>8. RATE LIMITING</b>	<b>9</b>

## 1. Introduction

This guide is intended to give developers an overview of how to integrate with the breatheHR API. There is also detailed, interactive online documentation available on our developer centre at <http://developer.breathehr.com>

You will notice that the API documentation page has a space for you to put in an API key in the top right. Place your **sandbox** key here and press the button, and you can try out sandboxed API requests from within the documentation itself! The details of the sandbox and the production environments are explained below.

## 2. Authenticating

Authentication to the breatheHR API is achieved with a token, sent in the HTTP headers of every request. The **X-API-KEY** header should contain the string shown in your breatheHR > settings > modules > API setup.

## 3. Environments

You are provided with a sandbox environment in which to test your API consuming code. When you enable the API, the system uses different keys for the production and sandbox environments, to prevent accidentally using the wrong environment. The production keys and prefixed with **prod-** and the sandbox keys are prefixed with **sandbox-**.

In addition, the production and sandbox environments reside on different domains. The sandbox environment is at [api.sandbox.breathehr.com](http://api.sandbox.breathehr.com)

## 4. Making Requests

The breatheHR API uses the JSON-over-HTTP architecture that is widely used within web-based API system. Broadly, this means that:

- The type of action to be carried out is determined by a request's URL and its HTTP method
- Any data to be submitted is sent in the form of a JSON document in the body of a POST request
- Additional parameters may be sent in the URL query string
- All responses are in the form of JSON documents
- The HTTP response code will give additional information as to the outcome of the request

Over the following pages we will detail the conventions used within the breatheHR API.

## 5. API Actions within the breatheHR system

Some actions within breatheHR are tied to a specific user, such as approving a leave request. When these actions are carried out via the API, they will show as having been performed by an “API Employee.” This “employee” does not actually exist within the system and does not count towards any account limits or other statistical context, and exists purely as a placeholder where the originator of an action must be recorded.

## 6. Request and Response Format

Here is an example of a request to create a new employee, made using the popular cURL tool, and the response:

*Listing 1:*

```
1) $ curl https://api.breathehr.com/v1/employees \  
2)  -H 'X-API-KEY: prod-10cTwKW8ln5ARz04IjsrJEQTZgP3Ir1sWfyu0LdqC1Y' \  
3)  -d employee[first_name]=foo \  
4)  -d employee[last_name]=bar \  
5)  -d employee[email]=foo@bar.com \  
6)  -d employee[job_title]=Director \  
7)  -d employee[join_date]=2014/10/01  
8)  
9) {  
10) "employees": [  
11)   {  
12)     "length_of_service_in_months":3,  
13)     "age":null,  
14)     "full_or_part_time":null,  
15)     "notice_period":null,  
16)     "working_pattern":{  
17)       "id":1,  
18)       "name":"Default working week M-F 8hrs",  
19)       "total_hours":"40.0",  
20)       "default":true  
21)     },  
22)     "holiday_allowance":{  
23)       "name":"Full Time Hours M-F 200+64 (25+8)",  
24)       "id":1  
25)     },  
26)     "line_manager":null,  
27)     "holiday_approver":null,  
28)     "department":null,  
29)     "receives_statutory_holidays":false,  
30)     "status":"Current employee",  
31)     "company_join_date":"2014-10-01",  
32)     "job_start_date":"2015-01-22",  
33)     "job_title":"Director",  
34)     "hr":false,  
35)     "id":5,  
36)     "account_id":1,  
37)     "first_name":"foo",  
38)     "last_name":"bar",  
39)     "email":"foo@barzz.com",  
40)     "created_at":"2015-01-22T10:44:35Z",  
41)     "updated_at":"2015-01-22T10:44:35Z"  
42)   }  
43) ]  
44) }
```

Please note that the response has been edited for brevity, in order to highlight some important attributes. In the following sub section we will examine the request and response in more detail.

## 6.1 Specifying the parameters (HTTP or JSON)

Here they are sent in the HTTP form parameter style. The square bracket notation seen on lines 3-7 implies a nested object structure; this technique may be familiar to you from other web frameworks and APIs. The same request could be made by POSTing a whole JSON object, similar to the response:

*Listing 2:*

```
1) $ curl https://api.breathehr.com/v1/employees \  
2) -H 'Content-Type:application/json' \  
3) -H 'X-API-KEY: prod-10cTwKW8ln5ARz04IjsrJEQTZgP3Ir1sWfyu0LdqC1Y' -d '  
4) {  
5)   "employee": {  
6)     "first_name": "foo",  
7)     "last_name": "bar",  
8)     "email": "foo@bar.com",  
9)     "job_title": "Director",  
10)    "join_date": "2014/10/01"  
11)  }  
12) }'  
13)  
14) <RESPONSE NOT SHOWN>
```

Note that in *listing 2*, the `Content-Type` header is supplied with a value of `application/json`. This header is required for this alternative request format to function.

## 6.2 Date and Time formats

You can see examples of date and time formats in both listings, eg. the join date on line 10 of *listing 2*. You can see an example of the date/time format on line 40 of *listing 1*. These are based on the ISO 8601 standard. Details of this format are available online, and the date/time functions in your language of choice should be able to output dates and times in this way with no problem. Broadly speaking, dates are specified as `YYYY-MM-DD` while dates while times are specified as `YYYY-MM-DDTHH:MM:SSZ`.

## 6.3 Nested objects

Two examples of nested objects in the above example are the `working_pattern` and the `holiday_allowance`. In the breatheHR API, where an objects attribute is also another attribute available within the system, it appears as a nested object in API responses. In the *listing 1* above, the `working_pattern` attribute is an object with a small set of its own attributes, including an `id`. We can make another request to the working patterns resource, which will give us the full details of all working patterns, including this one:

**Listing 3:**

```
1) $ curl https://api.breathehr.com/v1/working_patterns \  
2) -H 'X-API-KEY: prod-10cTwKW8ln5ARz04IjsrJEQTZgP3IrisWfyu0LdqC1Y'  
3) {  
4)   "working_patterns": [  
5)     {  
6)       "default": true,  
7)       "id": 1,  
8)       "name": "Default working week M-F 8hrs",  
9)       "total_hours": "40.0",  
10)      "mon": true,  
11)      "tue": true,  
12)      "wed": true,  
13)      "thu": true,  
14)      "fri": true,  
15)      "sat": false,  
16)      "sun": false,  
17)      "mon_hr": "8.0",  
18)      "tue_hr": "8.0",  
19)      "wed_hr": "8.0",  
20)      "thu_hr": "8.0",  
21)      "fri_hr": "8.0",  
22)      "sat_hr": "0.0",  
23)      "sun_hr": "0.0",  
24)      "created_at": "2012-09-11T13:38:26Z",  
25)      "updated_at": "2013-12-09T16:19:20Z"  
26)    },  
27) <FULL RESPONSE NOT SHOWN>
```

## 6.4 The root object

All requests and responses consist of a JSON object with a single key containing all the data. When creating an object, the value is an object representing the entity to be created, and they key is named for the type of object being created. This can be seen on line 5 of *listing 2* and in the square bracket nesting syntax of *listing 1* as previously discussed.

This pattern is repeated for the response objects. As before there is a root object, with a single key. The value is an array containing all the objects returned. Please note that the breatheHR API will always return an array of objects, even if only a single object is contained within. Correspondingly, the key will be the pluralised version of the object type.

For example, if we are requesting a list of employees, the API will return an object with a key `employees` pointing to an array of `employee` objects. However, if our request returns a single `employee` object, such as when we create a new `employee`, the API will also return an object with an `employees` key pointing to an array containing our single newly created `employee`. This behaviour is implemented to ensure consistency across API responses, reducing the number of special cases your consuming code will have to deal with.

## 6.5 Pagination

Results from the API which return collections of objects are paginated, ie. split up into pages. We use a system which is compliant with the proposed RFC-5988 standard for web linking.

Paginated responses will return `Link` and `Total` HTTP headers. Here is an example:

*Listing 4:*

```
1) $ curl https://api.breathehr.com/v1/employees?page=2 -i -H 'X-API-Key:
   prod-10cTwKW8ln5ARz04IjsrJEQTZgP3Ir1sWfyu0LdqC1Y'
2) HTTP/1.1 200 OK
3) Content-Type: application/json
4) Link: <https://api.breathehr.com/v1/employees?page=1>; rel="first", <https://api.breathehr.com/v1/employees?
   page=1>; rel="prev", <https://api.breathehr.com/v1/employees?page=23>; rel="last", <https://api.breathehr.com/
   v1/employees?page=3>; rel="next"
5) Total: 572
6)
7) <RESPONSE NOT SHOWN>
```

The `Link` header contains the URLs for the first, previous, next and last pages of the collection being returned. The `Total` header contains the total number of *records* in the collection, not the total number of pages. (The total number of pages is found in the URL for the last page.) These headers can help simplify your client code.

As you may have noticed, we tell the API which page to return with a `page` query parameter. Similarly, we can request more items per page with a `per_page` parameter, up to a maximum of 100. The default is 25 records per page.

## 7. Error Responses

When we submit an invalid request to the breatheHR API, we get a standardised error response in the form of a JSON object:

*Listing 5:*

```
1) $ curl https://api.breathehr.com/v1/employees -H 'X-API-KEY: prod-10cTwKW8ln5ARz04IjsrJEQTZgP3Ir1sWfyu0LdqC1Y' \
2) -d employee[first_name]=foo
3) {
4)   "error":{
5)     "type":"Record Invalid",
6)     "employee[last_name]":[
7)       "missing"
8)     ],
9)     "employee[email]":[
10)      "missing"
11)    ]
12)  }
13) }
```

In this case, since we were trying to create an `employee` object but we did not submit all the required fields, we get an `error` object which mirrors the structure of an actual `employee` object to some extent, except that each field points to an array of error messages. There is also a `type` field describing the type of error. The HTTP error code will be `400`.

## 8. Rate Limiting

The breatheHR API supports a maximum of 12 requests in 60 seconds per customer, so roughly one request every 5 seconds. If you exceed this limit, you will receive the following response:

*Listing 6:*

```
14) {  
15)   "error": {  
16)     "type": "Rate Limit Reached"  
17)   }  
18) }
```

The HTTP status code will be [429](#).