REST API

Some of the features found on the Sigfox backend's website can also be accessed programmatically using a webservice API. This API uses the HTTP protocol, following the REST principles.

All API endpoints return data in the JSON format, with the corresponding «application/json» content type header.
If a property has no value, it won't appear in the result.

General protocol information

Authentication and security

The API is accessible only using HTTPS, and all API endpoints require authentication credentials (login and password). Since all communication is encrypted, the authentication is performed using the simple HTTP Basic scheme.

Authentication credentials are associated to a group, which define the available device types that will be available with those credentials. A group initially does not have associated credentials, and it's up to you to create them using the «Generate credentials» buttons in the group edition screen.

If the API credentials ever get compromised, new ones can be regenerated at any moment, invalidating the previous ones.

CORS and JSONP are intentionally unsupported
Your credentials must remain private: exposing them to third parties is a bad practice.
CORS and JSONP JavaScript techniques tends to expose your credentials to your users.
If you really need to call Sigfox API from JavaScript in the browser, you must set a reverse proxy on your website.
Be careful not to blindly proxify all request to Sigfox backend but select only needed ones.

HTTP status codes

The Sigfox API uses a limited number of HTTP status codes:

- 200: request was successfully processed, and its result is sent in the response body
- 400: bad request, some required parameters are missing or their value is not valid
- 401: authentication credentials were not present or invalid
- 403: access denied, the request is not authorized with your credentials or the action is forbidden owing to some constraints
- 404: resource was not found for the given request
- 500: an unexpected error occured

Using the API from the command line with `curl`

`curl` is a popular command line utility to send HTTP requests. To use `curl` with the API credentials, use the following parameters.

Assuming an API login " `123login` " and password " `456passwd` ":

Get request

```
curl -u 123login:456passwd https://backend.sigfox.com/api/devicetypes
```

Post request

```
curl -u 123login:456passwd -H 'Content-Type:
application/json'
https://backend.sigfox.com/api/devicetypes/1fbc3d4
-d '{
"prefix" : "test-device-",
  "ids" : [
    { "id" : "C031", "pk":
"16b3cc86699ab4d3091a05ee" }
  ]
}'
```

Groups API

Group list

Lists all children groups of your group.

DEPRECATION notice : Until the 30th of June 2016, the API will list all groups (without paging) when neither `limit` nor `offset` query parameters are provided. After this date, specifying neither `limit` nor `offset`will return paginated results using default values for those parameters

Request

```
GET https://backend.sigfox.com/api/groups
```

Parameters:

Optionally, the request can also have the following parameters (see `next` response field below):

- `limit`: maximum number of groups to return.
- `offset`: number of groups skipped.
- `parentId`: id of the group for which the group list must be retrieved.

Response

```
{
    "data" : [ {
        "id":"510b848ee4b0ca47869752b5",
        "name":"Group 1",
        "nameCI":"group 1",
        "description":"Group 1 description text",
        "path":[
             "51f13454bc54518c7bae7d4d",
             "50f13484b846618c7bae77b7"
        ],
        "billable":true,
        "bssId": "bss-48631656321"
    }, {
        "id":"489b848ee4b0ca4786945614",
        "name":"Group 2",
        "nameCI":"group 2",
        "description":"Group 2 description text",
        "path":[
            "51f13454bc54518c7bae7d4d",
            "50f13484b846618c7bae77b7"
        ],
        "billable":false,
        "bssId": "bss-4686131845"
    } ],
    "paging": {
        "next":
"https://backend.sigfox.com/api/groups?limit=2&off
set=2"
    }
}
```

Fields:

- `data`: array of group information records
- `id`: the group's identifier
- `name`: the group's name
- `nameCI`: the group's name case insensitive
- `description`: description of this group
- `path`: Array with the group's parents' ids
- `billable`: true if the group is billable
- `bssId`: the group BSS id
- `paging`: paging information, if more groups are available
- `next`: URL of the next «page» of groups

Group information

Get the description of a particular group

Request

```
GET https://backend.sigfox.com/api/groups/{group-
id}
```

- `group-id`: the group identifier as returned by the `/api/groups` endpoint.

Response

```
{
    "id":"489b848ee4b0ca4786945614",
    "name":"Group 1",
    "nameCI":"group 1",
    "description":"Group 1 description text",
    "path":[
           "51f13454bc54518c7bae7d4d",
           "50f13484b846618c7bae77b7"
     ],
    "billable":true,
    "bssId": "bss-48631656321"
}
```

Fields:

- `id`: the group’s identifier
- `name`: the group’s name
- `nameCI`: the group’s name case insensitive
- `description`: description of this group
- `path`: Array with the group’s parents’ ids
- `billable`: true if the group is billable
- `bssId`: the group BSS id

Device types API

Device types list

Lists all device types available to your group.

Request

```
GET https://backend.sigfox.com/api/devicetypes
```

Parameters:

- None

Optionally, the request can also have the following parameter:

- `includeSubGroups`: `true` if you need also device types from child groups.
- `contractInfoId`: id of the contract info associated to the device types to retrieve

Response

```
{
  "data" : [ {
    "id" : "4d3091a05ee16b3cc86699ab",
    "name" : "Sigfox test device",
    "group" : "4d39a4c9e03e6b3c430e2188",
    "description" : "Little things in the black
boxes",
    "keepAlive" : 7200,
    "payloadType" : "None",
    "contract" : "523b1d10d777d3f5ae038a02"
  }, {
    "id" : "4d52cde8b4e81b6a0db3a00e",
    "name" : "Geolocated device",
    "group" : "4dc11c1a2f0f0590ce6d4879",
    "description" : "A device with a GPS",
    "keepAlive" : 0,
    "payloadType" : "Geolocation",
    "contract" : "547eab8a9336a3d6150ec144"
  } ]
}
```

Fields:

- `data`: array of device type information records
- `id`: the device type's identifier
- `name`: the device type's name
- `group`: the user group that owns this device type
- `description`: description of this device type
- `keepAlive`: the keep alive in minutes (0 mean inactive)
- `payloadType`: the payload type. Here are the values that can be set :
  - `"None"` if no specific type of payload must be set
  - `"String"` to display the data as an ascci string
  - `"Custom"` to display the frame as a custom format. The format must be defined with a field `payloadConfig` containing the line config.
  - `"Geolocation"` to display the data as a location with a certain format. It must be associated with a `geolocPayloadConfigId` field containing the id of the geolocation configuration. The id can be obtained with the `/api/geoloc-config/list` API
- `contract`: the contract id associated to this device type

Device type information

Get the description of a particular device type

Request

```
GET
https://backend.sigfox.com/api/devicetypes/{device
type-id}
```

- `devicetype-id`: the device type identifier as returned by the `/api/devicetypes` endpoint.

Response

```
{
  "id" : "4d3091a05ee16b3cc86699ab",
  "name" : "Sigfox test device",
  "group" : "4d39a4c9e03e6b3c430e2188",
  "description" : "Little things in the black
boxes",
  "keepAlive" : 0,
  "payloadType" : "Geolocation",
  "contract" : "523b1d10d777d3f5ae038a02"
}
```

Fields:

- `id`: the device type's identifier
- `name`: the device type's name
- `group`: the user group that owns this device type
- `description`: description of this device type
- `keepAlive`: the keep alive in minutes (0 mean inactive)
- `payloadType`: the payload type. Here are the values that can be set :
  - `"None"` if no specific type of payload must be set
  - `"String"` to display the data as an ascci string
  - `"Custom"` to display the frame as a custom format. The format must be defined with a field `payloadConfig` containing the line config.
  - `"Geolocation"` to display the data as a location with a certain format. It must be associated with a `geolocPayloadConfigId` field containing the id of the geolocation configuration. The id can be obtained with the `/api/geoloc-config/list` API
- `contract`: the contract id associated to this device type

Device type edition

Edit a device type

Request

```
    POST
https://backend.sigfox.com/api/devicetypes/edit

    {
        "id" : "deadbeef0486300cbebef070",
        "name" : "dtname",
        "description" : "the description",
        "keepAlive" : 3000,
        "alertEmail" : "alert@email.com",
        "payloadType" : "None",
        "downlinkMode" : 0,
        "downlinkDataString" : "deadbeefcafebabe",
    }
```

Fields:

If a field is not present, its value will not be changed.

- `id`: the id of the device type to edit. This id must correspond to an existing device type.
- `name`: the name of the device type.
- `description`: the description of the device type.
- `keepAlive`: the keep alive in minutes. Must be either 0 or >= 1800s.
- `payloadType`: set the payload type. Here are the values that can be set :
  - `"None"` if no specific type of payload must be set
  - `"String"` to display the data as an ascci string
  - `"Custom"` to display the frame as a custom format. The format must be defined with a field `payloadConfig` containing the line config.
  - `"Geolocation"` to display the data as a location with a certain format. It must be associated with a `geolocPayloadConfigId` field containing the id of the geolocation configuration. The id can be obtained with the `/api/geoloc-config/list` API
- `downlinkMode`: 0 for direct downlink, 1 for callback downlink mode.
- `downlinkDataString`: in mode 0 (direct), set the data that will be sent to the devices members of this device type. It must be an 8 hexadecimal bytes string.

Callback creation

Create a callback

Request

```
    POST
https://backend.sigfox.com/api/devicetypes/{device
type-id}/callbacks/new

    [{
        "channel" : "URL",
        "callbackType" : 0,
        "callbackSubtype" : 2,
        "url" :
"http://myserver.com/sigfox/callback",
        "httpMethod" : "POST",
        "enabled" : true,
        "sendDuplicate" : false,
        "sendSni": false,
        "payloadConfig" : "var1::bool:1",
        "bodyTemplate" : "device : {device} /
{customData#var1}",
        "headers" : {
            "time" : "{time}"
        },
        "contentType" : "text/plain"
    }, {
        "channel" : "BATCH_URL",
        "callbackType" : 0,
        "callbackSubtype" : 2,
        "url" :
"http://myserver.com/sigfox/callback/batch",
        "linePattern" : "{device};{data};",
        "enabled" : true,
        "sendDuplicate" : false,
        "sendSni": false
    }]
```

Parameters:

- `devicetype-id`: **the device type identifier as returned by the** `/api/devicetypes` **endpoint.**

Fields:

The callback definitions json format depends on the `channel` **field :**

- `"URL"`
  - `url` **the callback url**
  - `httpMethod` **the HTTP method used to send the callback (GET, POST or PUT)**
  - `sendSni (optional)` **Send SNI (Server Name Indication) for SSL/TLS connections**
  - `bodyTemplate` **the body template of the request. Only if** `httpMethpd` **is set to POST Or PUT. It can contain predefined and custom variables.**
  - `contentType` **the content type of the request. Only taken into account for POST requests. Available content types are**
    - **text/plain**
    - **application/json**
    - **application/x-www-form-urlencoded, last but not least, the default one.**
  - `headers` **headers of the request. The header value can contain a variable (predefined or custom).**
- `"BATCH_URL"`
  - `url` **the callback url**
  - `linePattern` **the line pattern representing a message.**
  - `sendSni (optional)` **Send SNI (Server Name Indication) for SSL/TLS connections**
- `"EMAIL"`
  - `subject` **the mail subject**
  - `recipient` **the recipient of the email. Must be a valid email address.**
  - `message` **the content of the message.**

The common fields for all the callback definitions are :

- `channel` **"URL", "BATCH_URL" or "EMAIL"**

- `callbackType`
  - `0` DATA

    Subtypes :

    - `2` UPLINK callback for an uplink message
    - `3` BIDIR callback for a bidirectional message
  - `1` SERVICE

    Subtypes :

    - `0` STATUS callback sending information about the status of a device
    - `4` ACKNOWLEDGE callback sent on a downlink acknowledge message
    - `5` REPEATER callback triggered when a repeater sends an OOB (available for SERVICE callbacks)
  - `2` ERROR
- `callbackSubtype` the subtype code, see the code definition above. The subtype must be valid against its type.
- `enabled` true to enable the callback, false else
- `sendDuplicate` true to duplicates callback, false else
- `enabled` true to enable the callback, false else
- `payloadConfig` this is a configuration to define custom variables that you then can use in the `bodyTemplate`, `url`, `headers`.

Custom message type decoding grammar

The "custom format" grammar is as follows :

```
format = field_def [" " field_def]* ;
field_def = field_name ":" byte_index ":" type_def ;
field_name = (alpha | digit | "#" | "_")* ;
byte_index = [digit*] ;
type_def = bool_def | char_def | float_def | uint_def ;
bool_def = "bool:" ("0" | "1" | "2" | "3" | "4" | "5" | "6" | "7") ;
char_def = "char:" length ("0" | "1" | "2" | "3" | "4" | "5" | "6" | "7");
float_def = "float:" ("32" | "64") [ ":little-endian" | ":big-endian" ] ("0" | "1" | "2" | "3" | "4" | "5" | "6" |
"7");
uint_def = "uint:" ["1" - "64"] [ ":little-endian" | ":big-endian" ] ("0" | "1" | "2" | "3" | "4" | "5" | "6" | "7");
int_def = "int:" ["1" - "64"] [ ":little-endian" | ":big-endian" ] ("0" | "1" | "2" | "3" | "4" | "5" | "6" | "7");
length = number* ;
digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

A field is defined by its name, its position in the message bytes, its length and its type :

- the field name is an identifier including letters, digits and the '-' and '_' characters.
- the byte index is the offset in the message buffer where the field is to be read from, starting at zero. If omitted, the position used is the current byte for boolean fields, the next byte for all other types if the previous element has no bit offset and the last byte used if the previous element has a bit offset. For the first field, an omitted position means zero (start of the message buffer)

Next comes the type name and parameters, which varies depending on the type :

- **boolean** : parameter is the bit position in the target byte
- **char** : parameter is the number of bytes to gather in a string, and optionally the bit offset where to start the reading of the first byte, Default value is 7 for the offset
- **float** : parameters are the length in bits of the value, which can be either 32 or 64 bits, optionally the endianness for multi-bytes floats, and optionally the bit offset where to start the reading of the first byte. Default is big endian and 7 for the offset. Decoding is done according to the IEEE 754 standard.

- **uint** (unsigned integer) : parameters are the number of bits to include in the value, optionally the endianness for multi-bytes integers, and optionally the bit offset where to start the reading of the first byte. Default is big endian and 7 for the offset.
- **int** (signed integer) : parameters are the number of bits to include in the value, optionally the endianness for multi-bytes integers, and optionally the bit offset where to start the reading of the first byte. Default is big endian and 7 for the offset.

Examples :

Format	Message (in hex)
int1::uint:8 int2::uint:8	1234
b1::bool:7 b2::bool:6 i1:1:uint:16	C01234
b1::bool:7 b2::bool:6 i1:1:uint:16:little-endian	801234
b1::bool:7 b2::bool:6 i1:1:uint:16:little-endian i2::uint:8	80123456
str::char:6 i1::uint:16 i2::uint:32	41424344454601234567890A
str::char:6 i1::uint:16 i2::uint:32:2	41424344454601234567890A

Response

```
    [
        "57285afd061b7015ef7d39b4",
        "5717383fa2154c0d19592424",
        "57285b02061b7015ef7d39b5"
    ]
```

List of identifiers of the created callbacks

Callback list

List the callbacks for a device type

Request

```
    POST
https://backend.sigfox.com/api/devicetypes/{device
type-id}/callbacks
```

Parameters:

- `devicetype-id`: **the device type identifier as returned by the** `/api/devicetypes` **endpoint.**

Response

```
    {
        data : [{
            "id" : "deadbeeffacecafebabecafe"
            "channel" : "URL",
            "callbackType" : 0,
            "payloadConfig" : "int1::uint:8
int2::uint:8",
            "callbackSubtype" : 0,
            "urlPattern" :
"http://myserver.com/sigfox/callback",
            "httpMethod" : "POST",
            "headers" : { "key1" : "value1",
                          "key2" : "value2" },
            "enabled" : true,
            "sendDuplicate" : false,
            "dead":false,
            "downlinkHook":false

        }, {
            "id" : "beefbeefcafecafebabecafe"
            "channel" : "BATCH_URL",
            "callbackType" : 0,
            "callbackSubtype" : 0,
            "urlPattern" :
"http://myserver.com/sigfox/callback/batch",
            "httpMethod" : "POST",
            "linePattern" : "{device};{data};",
            "enabled" : true,
            "sendDuplicate" : false,
            "dead":false,
            "downlinkHook":false

        }],
        service: [{
            "id" : "beefbeeffaceb00cbabecafe"
            "channel" : "URL",
            "callbackType" : 1,
            "callbackSubtype" : 0,
            "urlPattern" :
"http://myserver.com/sigfox/callback/service",
            "httpMethod" : "GET",
            "enabled" : true,
            "sendDuplicate" : false,
            "dead":false,
            "downlinkHook":false

        }]
```

Fields:

The callback definitions json format depends on the `channel` field :

- `"URL"`
  - `urlPattern` **the callback url**
  - `httpMethod` **the http method use to send callback, GET or POST.**
  - `downlinkHook` **A boolean indicating if the callback is used to set a response .**
  - `headers` **the headers of the http request to send, as an object with key:value.**
- `"BATCH_URL"`
  - `urlPattern` **the callback url**
  - `httpMethod` **the http method use to send callback, GET or POST.**
  - `linePattern` **the line pattern representing a message.**
- `"EMAIL"`
  - `subject` **the mail subject**
  - `recipient` **the recipient of the email.**
  - `message` **the content of the message.**

- `channel` **"URL", "BATCH_URL" or "EMAIL"**
- `callbackType`
  - `0` **DATA**

    Subtypes :

    - `2` **UPLINK callback for an uplink message**
    - `3` **BIDIR callback for a bidirectional message**
  - `1` **SERVICE**

    Subtypes :

    - `0` **STATUS callback sending information about the status of a device**
    - `4` **ACKNOWLEDGE callback sent on a downlink acknowledge message**
    - `5` **REPEATER callback triggered when a repeater sends an OOB (available for SERVICE callbacks)**
  - `2` **ERROR**
- `callbackSubtype` **the subtype code, see the code definition above. The subtype must be valid against its type.**
- `payloadConfig` **The custom payload configuration. Only for DATA callbacks**
- `enabled` **true to enable the callback, false else**
- `sendDuplicate` **true to duplicates callback, false else**
- `dead` **true if last use of the callback fails, false else**
- `downlinkHook` **true if this callback is used for downlink, false else**

Callback deletion

Delete a callback

Request

```
    POST
https://backend.sigfox.com/api/devicetypes/{device
type-id}/callbacks/{callback-id}/delete
```

Parameters:

- `devicetype-id`: **the device type identifier as returned by the** `/api/devicetypes` **endpoint.**
- `callback-id`: **the id of the callback as found in the** `/api/devicetypes/{devicetype-id}/callbacks` **endpoint.**

Response

- 200 if it was OK
- 404 when the device type does not exist, or when the callback id does not correspond to this device type.

Callback enable/disable

Enable or disable a callback

Request

```
    POST
https://backend.sigfox.com/api/devicetypes/{device
type-id}/callbacks/{callback-
id}/enable?enabled=true
```

Parameters:

- `devicetype-id`: **the device type identifier as returned by the** `/api/devicetypes` **endpoint.**
- `callback-id`: **the id of the callback as found in the** `/api/devicetypes/{devicetype-id}/callbacks` **endpoint.**
- `enabled`: **true to enable the callback, false to disable it.**

Response

- 200 if it was OK
- 404 when the device type does not exist, or when the callback id does not correspond to this device type.

Callback downlink selection

Select a downlink callback. The given callback will be selected as the downlink one, the one that was previously selected will be no more be selected for downlink.

Request

```
    POST
https://backend.sigfox.com/api/devicetypes/{device
type-id}/callbacks/{callback-id}/downlink
```

Parameters:

- `devicetype-id`: the device type identifier as returned by the `/api/devicetypes` endpoint.
- `callback-id`: the id of the callback as found in the `/api/devicetypes/{devicetype-id}/callbacks` endpoint.

Response

- 200 if it was OK
- 404 when the device type does not exist, or when the callback id does not correspond to this device type.

Error status events sent for a list of devices

Get the communication down events that were sent for a list of devices belonging to the same device type, in reverse chronological order (most recent events first).

Request

```
GET
https://backend.sigfox.com/api/devicetypes/{device
type-id}/status/error
```

Parameters:

- `devicetype-id`: the device type identifier as returned by the `/api/devicetypes` endpoint.

Optionally, the request can also have the following parameter (see `next` response field below):

- `limit`: maximum number of status events to return, default 100.
- `before`: return status events sent before this timestamp in milliseconds since Unix Epoch.
- `since`: return status events sent since this timestamp in milliseconds since Unix Epoch.
- `offset`: number of events to skip (between 0 and 5000). Normally you should not have to worry about this parameter as the app will set this parameter automatically in the response, in the URL of the next page (see field `next` in response)

Response

```
{
  "data" : [ {
    "deviceId" : "0235",
    "time" : 1381410600026,
    "message" : "No message received since 2013-
10-08 15:36:21",
    "severity" : "ERROR",
    "deviceTypeId" : "5256c4d6c9a871b80f5a2e50",
    "callbacks" : [ {
      "url" :
"http://host/path?id=0235&time=1381410600",
      "status" : 600,
      "info" : "Connection refused: host/path"
    }, {
      "subject" : "some subject",
      "message" : "some messages",
      "status" : 200
    } ]
  }, {
    "deviceId" : "4830",
    "time" : 1381300600026,
    "message" : "No message received since 2013-
10-08 15:36:21",
    "severity" : "ERROR",
    "deviceTypeId" : "5256c4d6c9a871b80f5a2e50",
    "callbacks" : [ {
      "url" :
"http://host/path?id=4830&time=1381300600",
      "status" : 200
    }, {
      "subject" : "some subject",
      "message" : "some messages",
      "status" : 200
    } ]
  } ],
  "paging" : {
    "next" :
"https://backend.sigfox.com/api/devicetypes/5256c4
d6c9a871b80f5a2e50/status/error?limit=100&offset=1
00"
  }
}
```

Fields:

- `data`: the array of device communication down events
- `deviceId`: the device identifier which has been detected down
- `time`: the time where the event was sent, in milliseconds since the Unix Epoch
- `message`: the reason of this event

- `severity`: ERROR
- `deviceTypeId`: the device type identifier
- `callbacks`: list of ERROR callbacks if any, see doc for information about ERROR callbacks.
  - Email:
    - `subject`: the subject of the mail which have been sent
    - `message`: the body of the mail which have been sent
    - `status`: present only if the response has already been processed, 200 if OK, otherwise 600.
  - HTTP:
    - `url`: the URL called when this event has been emitted
    - `status`: present only if the response has already been processed, contains the HTTP Response status or 600 if an other error occurred.
    - `info`: present only if the response has already been processed, contains additional information.
- `paging`: paging information, if more events are available
- `next`: URL of the next «page» of events

Warning status events sent for a list of devices

Get the network issues events that were sent for a list of devices belonging to the same device type, in reverse chronological order (most recent events first).

Request

```
GET
https://backend.sigfox.com/api/devicetypes/{device
type-id}/status/warn
```

Parameters:

- `devicetype-id`: the device type identifier as returned by the `/api/devicetypes` endpoint.

Optionally, the request can also have the following parameter (see `next` response field below):

- `limit`: maximum number of status events to return, default 100.
- `before`: return status events sent before this timestamp in milliseconds since Unix Epoch.
- `since`: return status events sent since this timestamp in milliseconds since Unix Epoch.

Response

```
{
  "data" : [ {
    "deviceIds" : [ "0235", "023A", "4830" ],
    "time" : 1381410600026,
    "message" : "Sigfox network experiencing
issues that could cause message loss or delay for
some devices.",
    "severity" : "WARN",
    "deviceTypeId" : "5256c4d6c9a871b80f5a2e50",
    "callbacks" : [ {
      "url" :
"http://host/path?id=0235&time=1381410600",
      "status" : 600,
      "info" : "Connection refused: host/path"
    }, {
      "subject" : "some subject",
      "message" : "some messages",
      "status" : 200
    } ]
  }],
  "paging" : {
    "next" :
"https://backend.sigfox.com/api/devicetypes/5256c4
d6c9a871b80f5a2e50/status/warn?limit=100&offset=10
0"
  }
}
```

Fields:

- `data`: the array of device communication down events
- `deviceIds`: contains an array of device identifiers
- `time`: the time where the event was sent, in milliseconds since the Unix Epoch
- `message`: the reason of this event
- `severity`: WARN
- `deviceTypeId`: the device type identifier
- `callbacks`: list of ERROR callbacks if any, see doc for information about ERROR callbacks.
  - Email:
    - `subject`: the subject of the mail which have been sent
    - `message`: the body of the mail which have been sent
    - `status`: present only if the response has already been processed, 200 if OK, otherwise 600.
  - HTTP:
    - `url`: the URL called when this event has been emitted
    - `status`: present only if the response has already been processed, contains the HTTP Response status or 600 if an other error occurred.
    - `info`: present only if the response has already been processed, contains additional information.
- `paging`: paging information, if more events are available
- `next`: URL of the next «page» of events

List custom geoloc configurations

Get the list of the custom geoloc configurations available for the given group

Request

```
 (DEPRECATED - It will be removed on the 30th of
March 2016)
```

```
GET
https://backend.sigfox.com/api/devicetypes/geolocs
-config
```

Parameters:

- `groupId`: the id of the given group.

Response

```
    [ {
        "id" : "521f7cf3e4b56d12a7b82ac4",
        "name" : "Sigfox",
        "groupName" : "Sigfox"
    }, {
        "id" : "52243fa7e4b87e1b8587bd5a",
        "name" : "Test",
        "groupName" : "Sigfox"
    } ]
```

Fields:

- `id`: the config id
- `name`: the name of the config
- `groupName`: the name of the group the config is member of

Messages sent by the devices of a device type

Get the messages that were sent by all the devices of a device type, in reverse chronological order (most recent messages first). All device messages are listed, including those from device are not associated to this device type anymore.

Request

```
GET
https://backend.sigfox.com/api/devicetypes/{device
type-id}/messages
```

Parameters:

- `devicetype-id` the identifier of the device type, as returned by the `/api/devicetypes` endpoint.

Optionally, the request can also have the following parameter (see `next` response field below):

- `limit`: maximum number of messages to return
- `before`: return messages sent before this timestamp (in seconds since the Unix Epoch).
- `since`: return messages sent since this timestamp (in seconds since the Unix Epoch).

- `offset`: number of messages to skip (between 0 and 5000). Normally you should not have to worry about this parameter as the app will set this parameter automatically in the response, in the URL of the next page (see field `next` in response)

Response

```
{
  "data" : [ {
    "device" : "002C",
    "time" : 1343321977,
    "data" : "3235353843fc",
    "computedLocation": {
      "lat" : 43.45,
      "lng" : 6.54,
      "radius": 500
    },
    "linkQuality" : "GOOD"
    }, {
      "tap" : "003D",
      "snr" : "38.2",
      "lat" : "43",
      "lng" : "6",
      "cbStatus" : [ {
            "status" : 200,
            "info" : "OK",
            "cbDef" : "[GET]
https://myHost.com/callback?id={device}&data={data
}&lat={lat}&lng={lng}&ack={ack}",
            "time" : 1343333000
        } ],
    } ]
  } ],
  "paging" : {
    "next" :
"https://backend.sigfox.com/api/devicetypes/51086f
cde4b0b1b30a01c491/messages?limit=100&before=13388
03969&offset=100"
  }
}
```

Fields:

- `data`: the array of device messages
- `device`: device identifier

- `time`: time the message was sent, in seconds since the Unix Epoch
- `data`: message content, hex encoded
- : the best SNR of the messages received by the network so far. (DEPRECATED - It will be removed on the 30th of March 2016. Replaced by `linkQuality`)
- `computedLocation`: contains the estimated position of the device within a circle based on the GPS data or the Sigfox Spot'It service. GPS data is used if the device has a device type with payload type "Geolocation", while Sigfox Spot'It service is used if the device is attached to a contract with the geolocation option enabled.
  - `lat`: the estimated latitude
  - `lng`: the estimated longitude
  - `radius`: the radius of the circle
- `linkQuality`: The link quality indicator value (LIMIT, AVERAGE, GOOD or EXCELLENT)
- `paging`: paging information, if more messages are available
- `next`: URL of the next «page» of messages

Disengage sequence number

Disengage sequence number check for next message of each device of the device type.

Request

```
GET
https://backend.sigfox.com/api/devicetypes/{device
type-id}/disengage
```

Parameters:

- `devicetype-id`: the device type identifier as returned by the `/api/devicetypes` endpoint.

Response

The response has no body. A status code 200 is returned when the disengagment have been done successfully.

Devices API

Devices for a given device type

Lists the devices associated to a specific device type.

Request

```
GET
https://backend.sigfox.com/api/devicetypes/{device
type-id}/devices?snr={snr}
```

Parameters:

- `devicetype-id`: the device type identifier as returned by the `/api/devicetypes` endpoint
- `snr`: optional, filter the device list according to the average signal to noise ratio of the last 25 received messages. Values can be :

    - 1, for SNR values from 0 to 10 dB
    - 2, for SNR values from 10 to 13 dB
    - 3, for SNR values from 13 to 20 dB
    - 4, for SNR values above 20 dB
- `limit`: maximum number of status events to return, default 100.
- `offset`: number of devices to skip (between 0 and 5000).

Response

```
{
  "data" : [ {
    "id" : "0B59",
    "name" : "Test 0B59",


    "type" : "4d3091a05ee16b3cc86699ab",
    "last" : 0,
    "averageSignal": 8.457,
    "averageSnr": 8.457,
    "averageRssi": -125.967,
    "state": 0,
    "lat" : 43.45,
     "lng" : 1.54,
     "computedLocation": {
       "lat" : 43.45,
       "lng" : 6.54,
       "radius": 500
     },
    "activationTime": 1404096340556,
    "pac": "545CB3B17AC98BA4",
    "tokenType": "CONTRACT",
     "contractId": "7896541254789654aedfba4c",
     "tokenEnd": 1449010800000,
    "preventRenewal": false
  }, {
    "id" : "0009",
    "name" : "Paris 1",

    "type" : "4d3091a05ee16b3cc86699ab",
    "last" : 1345013112,
    "averageSignal": 9.045,
    "averageSnr": 9.045,
    "averageRssi": -114.971,
    "state": 0,
        "lat" : 43.45,
         "lng" : -1.54,
         "computedLocation": {
           "lat" : 43.45,
           "lng" : 6.54,
           "radius": 500
         },
        "activationTime": 1404064540586,
        "pac": "545CB3B17AC98BA4",
        "tokenType": "CONTRACT",
        "contractId": "7896541254789654aedfba4c",
        "tokenEnd": 1449010800000,
        "preventRenewal": false
      }, ... ],
      "paging" : {
        "next" :
"https://backend.sigfox.com/api/devicetypes/5256c4
d6c9a871b80f5a2e50/devices?snr=1&limit=100&offset=
100"
      }
    }
```

Fields:

- `data`: the array of device information records
- `id`: the device's identifier
- `name`: the device's friendly name
- `type`: identifier of the device type this device belongs to
- `last`: last time when this device has sent a message, in seconds since the Unix Epoch. A value of zero means the device has never sent a message
- (DEPRECATED - It will be removed on the 30th of March 2016. Replaced by `averageSnr`)
- `averageSnr`: the average signal computed from the last 25 received messages (in dB)
- `averageRssi`: the average RSSI computed from the last 25 received messages (in dBm)
- `state`: the device state
  - `0`: OK
  - `1`: KO – Communication timeout
  - `2`: Off contract – Communication forbidden
  - `3`: Disabled – The device is about to be deleted
  - `4`: Warn – Network issues
- `lat`: the provided latitude of the device
- `lng`: the provided longitude of the device
- `computedLocation`: contains the estimated position of the device within a circle based on the GPS data or the Sigfox Spot'It service. GPS data is used if the device has a device type with payload type "Geolocation", while Sigfox Spot'It service is used if the device is attached to a contract with the geolocation option enabled.
  - `lat`: the estimated latitude
  - `lng`: the estimated longitude

    - `radius`: the radius of the circle
- `activationTime`: time of the contract token taking, in milliseconds since the Unix Epoch. no value is returned if the device does not have a token
- `pac`: the device's PAC
- `tokenType`: the type of token taken by this device. This field is only present if a token exists. There are curently two types of tokens :
    - `CONTRACT`: a classical token, with a start date and an end date.
    - `FREE`: a token offering test messages With such a token a device is not considered as activated, so not taken into account in billing.
- `contractId`: the id of the contract of the token. This field is only present if a token exists.
- `freeMessages`: the number of free messages left on this token. This field is only present if the token exists and is of type FREE.
- `tokenEnd`: a timestamp in milliseconds expressing the time the token expires. This field is only present if the token exists and is of type CONTRACT.
- `preventRenewal`: «true» if token renewal is deactivated.

Device information

Gets information about a device.

Request

```
GET
https://backend.sigfox.com/api/devices/{device-id}
```

Parameters:

- `device-id` the identifier of the device, as returned by the `/api/devicetypes/{devicetype-id}/devices` endpoint.

Response

```
    {
      "id" : "002C",
      "name" : "Labege 4",

      "type" : "4d3091a05ee16b3cc86699ab",
      "last" : 1343321977,
      "averageSignal": 8.065601,
      "averageSnr": 8.065601,
      "averageRssi": -122.56,
      "state": 0,
      "lat" : 43.45,
      "lng" : 1.54,
      "computedLocation": {
        "lat" : 43.45,
        "lng" : 6.54,
        "radius": 500
      },
      "activationTime": 1404096340556,
      "pac": "545CB3B17AC98BA4",
      "tokenType": "CONTRACT",
      "contractId": "7896541254789654aedfba4c",
      "tokenEnd": 1449010800000,
      "preventRenewal": false

    }
```

- `id`: the device's identifier
- `name`: the device's friendly name
- `type`: identifier of the device type this device belongs to
- `last`: last time when this device has sent a message, in seconds since the Unix Epoch. A value of zero means the device has never sent a message
- `averageRssi`: the average RSSI computed from the last 25 received messages (in dBm)
- (DEPRECATED - It will be removed on the 30th of March 2016. Replaced by `averageSnr`)
- `averageSnr`: the average signal computed from the last 25 received messages (in dB)
- `state`: the device state
  - `0`: OK
  - `1`: KO – Communication timeout
  - `2`: Off contract – Communication forbidden
  - `3`: Disabled – The device is about to be deleted
  - `4`: Warn – Network issues
- `lat`: the provided latitude of the device
- `lng`: the provided longitude of the device
- `computedLocation`: contains the estimated position of the device within a circle based on the GPS data or the Sigfox Spot'It service. GPS data is used if the device has a device type with payload type "Geolocation", while Sigfox Spot'It service is used if the device is attached to a contract with the geolocation option enabled.
  - `lat`: the estimated latitude
  - `lng`: the estimated longitude
  - `radius`: the radius of the circle
- `activationTime`: time of the contract token taking, in milliseconds since the Unix Epoch. no value is returned if the device does not have a token.
- `pac`: the device's PAC
- `tokenType`: the type of token taken by this device. This field is only present if a token exists. There are curently two types of tokens :
  - `CONTRACT`: a classical token, with a start date and an end date.
  - `FREE`: a token offering test messages With such a token a device is not considered as activated, so not taken into account in billing.
- `contractId`: the id of the contract of the token. This field is only present if a token exists.
- `freeMessages`: the number of free messages left on this token. This field is only present if the token exists and is of type FREE.
- `tokenEnd`: a timestamp in milliseconds expressing the time the token expires. This field is only present if the token exists and is of type CONTRACT.
- `preventRenewal`: «true» if token renewal is deactivated.

Device token state

Gets information about a device's token.

Request

```
GET
https://backend.sigfox.com/api/devices/{device-
id}/token-state
```

Parameters:

- `device-id` the identifier of the device, as returned by the `/api/devicetypes/{devicetype-id}/devices` endpoint.

Response

```
    {
      "code" : 1,
      "detailMessage" : "Off contract",
      "tokenType": "CONTRACT",
      "contractId": "7896541254789654aedfba4c",
      "tokenEnd": 1418673953200,
    }
```

- `code`: the device’s token state
  - `0`: OK
  - `1`: KO
  - `2`: N/A
- `detailMessage`: Detail message that could be:
  - `"Off contract"`: Device is off contract
  - `"Not communicated"`: There’s none token on this device
  - `"Invalid token"`: Token end time has been exceeded
  - `"Ok"`: Token is ok
- `tokenType`: the type of token taken by this device. This field is only present if a token exists. There are curently two types of tokens :
  - `CONTRACT`: a classical token, with a start date and an end date.
  - `FREE`: a token offering test messages With such a token a device is not considered as activated, so not taken into account in billing.
- `contractId`: the id of the contract of the token. This field is only present if a token exists.
- `freeMessages`: the number of free messages left on this token. This field is only present if the token exists and is of type FREE.
- `tokenEnd`: a timestamp in milliseconds expressing the time the token expires. This field is only present if the token exists and is of type CONTRACT.

Messages sent by a device

Get the messages that were sent by a device, in reverse chronological order (most recent messages first).

Request

```
GET
https://backend.sigfox.com/api/devices/{device-
id}/messages
```

Parameters:

- `device-id` the identifier of the device, as returned by the `/api/devicetypes/{devicetype-id}/devices` endpoint.

Optionally, the request can also have the following parameter (see `next` response field below):

- `limit`: maximum number of messages to return
- `before`: return messages sent before this timestamp (in seconds since the Unix Epoch).
- `since`: return messages sent since this timestamp (in seconds since the Unix Epoch).
- `offset`: number of messages to skip (between 0 and 5000). Normally you should not have to worry about this parameter as the app will set this parameter automatically in the response, in the URL of the next page (see field `next` in response)

Response

```
{
  "data" : [ {
    "device" : "002C",
    "time" : 1343321977,
    "data" : "3235353843fc",
    "snr" : "38.2",
    "computedLocation": {
      "lat" : 43.45,
      "lng" : 6.54,
      "radius": 500
    },
    "linkQuality" : "GOOD"
    }, {
      "tap" : "003D",
      "lat" : "43",
      "lng" : "6",
      "cbStatus" : [ {
            "status" : 200,
            "info" : "OK",
            "cbDef" : "[GET]
https://myHost.com/callback?id={device}&data={data
}&lat={lat}&lng={lng}&ack={ack}",
            "time" : 1343332000
        } ],
      "freq" : "N/A"
    } ]
  } ],
  "paging" : {
    "next" :
"https://backend.sigfox.com/api/devices/002C/messa
ges?limit=100&before=1338803969"
  }
}
```

Fields:

- `data`: the array of device messages
- `device`: device identifier
- `time`: time the message was sent, in seconds since the Unix Epoch
- `data`: message content, hex encoded
- : the best SNR of the messages received by the network so far. (DEPRECATED - It will be removed on the 30th of March 2016. Replaced by `linkQuality`)
- `computedLocation`: contains the estimated position of the device within a circle based on the GPS data or the Sigfox Spot'It service. GPS data is used if the device has a device type with payload type "Geolocation", while Sigfox Spot'It service is used if the device is attached to a contract with the geolocation option enabled.
  - `lat`: The estimated latitude
  - `lng`: The estimated longitude
  - `radius`: The radius of the circle
- `linkQuality`: The link quality indicator value (LIMIT, AVERAGE, GOOD or EXCELLENT)
- `paging`: paging information, if more messages are available
- `next`: URL of the next «page» of messages

Messages location for one device

Get the messages location from the following sources: GPS data or the Sigfox Spot'It service in reverse chronological order (most recent messages first). GPS data is used if the device has a device type with payload type "Geolocation", while Sigfox Spot'It service is used if the device is attached to a contract with the geolocation option enabled.

Request

```
GET
https://backend.sigfox.com/api/devices/{device-
id}/locations
```

Parameters:

- `device-id` the identifier of the device, as returned by the `/api/devicetypes/{devicetype-id}/devices` endpoint.

Optionally, the request can also have the following parameter (see `next` response field below):

- `limit`: maximum number of messages to return
- `before`: return messages sent before this timestamp (in seconds since the Unix Epoch).
- `since`: return messages sent since this timestamp (in seconds since the Unix Epoch).

- `offset`: number of messages to skip (between 0 and 5000). Normally you should not have to worry about this parameter as the app will set this parameter automatically in the response, in the URL of the next page (see field `next` in response)

Response

```
{
  "data" : [ {
    "time" : 1343321977000,
    "valid" : true,
    "lat" : 42.4631156,
    "lng" : 1.5652321,
    "radius" : 360,
  } ],
  "paging" : {
    "next" :
"https://backend.sigfox.com/api/devices/002C/locat
ions?limit=100&before=1338803969"
  }
}
```

Fields:

- `data`: the array of device messages location
- `time`: time the message was sent, in milliseconds since the Unix Epoch
- `valid`: true, if a valid estimation for this message is available (GPS or RSSI), false otherwise
- `lat`: latitude of the estimated position of the device, if a valid estimation exists for this message
- `lng`: longitude of the estimated position of the device, if a valid estimation exists for this message
- `radius`: radius around the estimated position in which the device can be, if a valid estimation exists for this message. For an exact position (GPS) radius is set to 0
- `paging`: paging information, if more messages are available
- `next`: URL of the next «page» of messages

Error status events sent for a device

Get the communication down events that were sent for a device, in reverse chronological order (most recent events first).

Request

```
GET
https://backend.sigfox.com/api/devices/{device-
id}/status/error
```

Parameters:

- `device-id` the identifier of the device, as returned by the `/api/devicetypes/{devicetype-id}/devices` endpoint.

Optionally, the request can also have the following parameter (see `next` response field below):

- `limit`: maximum number of status events to return, default 100.
- `before`: return status events sent before this timestamp in milliseconds since Unix Epoch.
- `since`: return status events sent since this timestamp in milliseconds since Unix Epoch.
- `offset`: number of events to skip (between 0 and 5000). Normally you should not have to worry about this parameter as the app will set this parameter automatically in the response, in the URL of the next page (see field `next` in response)

Response

```
{
  "data" : [ {
    "deviceId" : "4830",
    "time" : 1381300600026,
    "message" : "No message received since 2013-
10-08 15:36:21",
    "severity" : "ERROR",
    "deviceTypeId" : "5256c4d6c9a871b80f5a2e50",
    "callbacks" : [ {
      "url" :
"http://host/path?id=4830&time=1381300600",
      "status" : 200
    }, {
      "subject" : "some subject",
      "message" : "some messages",
      "status" : 200
    } ]
  } ],
  "paging" : {
    "next" :
"https://backend.sigfox.com/api/devices/4830/statu
s/error?limit=100&offset=100"
  }
}
```

Fields:

- `data`: the array of device communication down events
- `deviceId`: the device identifier which has been detected down
- `time`: the time where the event was sent, in milliseconds since the Unix Epoch
- `message`: the reason of this event
- `severity`: ERROR
- `deviceTypeId`: the device type identifier
- `callbacks`: list of ERROR callbacks if any, see doc for information about ERROR callbacks.
  - Email:

    - `subject`: the subject of the mail which have been sent
    - `message`: the body of the mail which have been sent
    - `status`: present only if the response has already been processed, 200 if OK, otherwise 600.
  - HTTP:
    - `url`: the URL called when this event has been emitted
    - `status`: present only if the response has already been processed, contains the HTTP Response status or 600 if an other error occurred.
    - `info`: present only if the response has already been processed, contains additional information.
- `paging`: paging information, if more events are available
- `next`: URL of the next «page» of events

Warning status events sent for a device

Get the network issues events that were sent for a device, in reverse chronological order (most recent events first).

Request

```
GET
https://backend.sigfox.com/api/devices/{device-
id}/status/warn
```

Parameters:

- `device-id` the identifier of the device, as returned by the `/api/devicetypes/{devicetype-id}/devices` endpoint.

Optionally, the request can also have the following parameter (see `next` response field below):

- `limit`: maximum number of status events to return, default 100.
- `before`: return status events sent before this timestamp in milliseconds since Unix Epoch.
- `since`: return status events sent since this timestamp in milliseconds since Unix Epoch.
- `offset`: number of events to skip (between 0 and 5000). Normally you should not have to worry about this parameter as the app will set this parameter automatically in the response, in the URL of the next page (see field `next` in response)

Response

```
{
  "data" : [ {
    "deviceIds" : [ "0235", "023A", "4830" ],
    "time" : 1381410600026,
    "message" : "Sigfox network experiencing
issues that could cause message loss or delay for
some devices.",
    "severity" : "WARN",
    "deviceTypeId" : "5256c4d6c9a871b80f5a2e50",
    "callbacks" : [ {
      "url" :
"http://host/path?id=4830&time=1381410600",
      "status" : 600,
      "info" : "Connection refused: host/path"
    }, {
      "subject" : "some subject",
      "message" : "some messages",
      "status" : 200
    } ]
  } ],
  "paging" : {
    "next" :
"https://backend.sigfox.com/api/devices/4830/statu
s/warn?limit=100&offset=100"
  }
}
```

Fields:

- `data`: the array of device communication down events
- `deviceIds`: contains an array of device identifiers
- `time`: the time where the event was sent, in milliseconds since the Unix Epoch
- `message`: the reason of this event
- `severity`: WARN
- `deviceTypeId`: the device type identifier
- `callbacks`: list of ERROR callbacks if any, see doc for information about ERROR callbacks.
  - Email:
    - `subject`: the subject of the mail which have been sent
    - `message`: the body of the mail which have been sent
    - `status`: present only if the response has already been processed, 200 if OK, otherwise 600.
  - HTTP:
    - `url`: the URL called when this event has been emitted
    - `status`: present only if the response has already been processed, contains the HTTP Response status or 600 if an other error occurred.
    - `info`: present only if the response has already been processed, contains additional information.
- `paging`: paging information, if more events are available
- `next`: URL of the next «page» of events

Network status of a device

Return the network status for a specific device. The value is based on the computed state of all the base stations which received the last messages for this device.

Request

```
GET
https://backend.sigfox.com/api/devices/{id}/networ
kstate
```

Parameters:

- `id` the identifier of the device.

Response

```
{
  "networkStatus" : "OK|NOK"
}
```

Fields:

- `networkStatus`: the network status for the device. OK, the network works, NOK otherwise.

Device message metrics

Returns the total number of device messages for one device, this day, this week and this month.

Request

```
GET
https://backend.sigfox.com/api/devices/{id}/messag
es/metric
```

Parameters:

- `id` the identifier of the device (in hexadecimal).

Response

```
{
  "lastDay": 47,
  "lastWeek": 276,
  "lastMonth": 784
}
```

Fields:

- `lastDay`: The number of message received from this device during the last 24 hours.
- `lastWeek`: The number of message received from this device during the last 7 days.
- `lastMonth`: The number of message received from this device during the last 30 days.

Device consumptions

Get a Device’s consumptions for a year

Request

```
GET https://backend.sigfox.com/api/device/{device-id}/consumptions/{year}
```

Parameters:

- `device-id`: the identifier of the device (in hexa).
- `year`: the year of consumption.

Response

```
{
    "consumption": {
        "id" : "110160_2015",
        "consumptions": [
            {
                "frameCount": 12,
                "downlinkFrameCount": 3
            },
            ...
        ]
    }
}
```

Fields:

- `id`: The consumption's identifier (format : {device-id}_{year})
- `consumptions`: Array of consomption per day (index in this array is number of the day in year, eg : 10 = 9th january)
  - `frameCount`: Number of uplink messages received
  - `downlinkFrameCount`: Number of downlink messages received

Misc APIs

Messages not sent this month

Returns device messages where **at least one callback has failed**, in reverse chronological order (most recent message first).

```
GET
https://backend.sigfox.com/api/callbacks/messages/error
```

Optionally, the request can also have the following parameter (see `next` response field below):

- `limit`: maximum number of messages to return, default 100.
- `offset`: number of messages skipped, used when paginated.

- `since`: return messages sent since this timestamp (in milliseconds since the Unix Epoch), default to a month ago.
- `before`: return messages sent before this timestamp (in milliseconds since the Unix Epoch).
- `hexId`: device identifier.
- `deviceTypeId`: device type identifier if no device identifier provided.
- `groupId`: group identifier if no device type identifier or device identifier provided.

Response

```
{
  "data" : [ {
    "device" : "00A0",
    "deviceType" : "4fe9cc70e4b014be5cc90246",
    "time" : 1381766225,
    "data" : "3b3616201e100000",
    "snr" : 10.5,
    "status" : 404,
    "message" : "Page Not Found",
    "callback" : {
      "url" :
"http://host/batch?data=00A0%3B1381766225%3B3b3616
201e100000%3B"
    },
    "parameters" : {
      "time" : 1381766225,
      "data" : "3b3616201e100000",
      "device" : "00A0"
    }
  }, {
    "device" : "00A1",
    "deviceType" : "4fe9cc70e4b014be5cc90246",
    "time" : 1381766225,
    "data" : "3b3616201e100000",
    "snr" : 10.5,
    "status" : 600,
    "message" : "Connection refused",
    "callback" : {
      "url" :
"http://host/single?device=00A1&time=1381766225&du
plicate=false&snr=10.50&station=001F&data=3b361620
1e100000",
      "headers": {
        "Authorization" : "Basic
U0lHRk9YOndpbGxfcnVsZSB0aGUgd29ybGQ="
```

```
      },
      "method" : "POST",
      "body":
"device=00A1&time=1381766225&duplicate=false&snr=1
0.50&station=001F&data=3b3616201e100000",
      "contentType": "application/x-www-form-
urlencoded"
    },
    "parameters" : {
      "time" : 1381766225,
      "duplicate" : false,
      "station" : "001F",
      "data" : "3b3616201e100000",
      "device" : "00A1",
      "snr" : 10.5
    }
  }, {
    "device" : "00A1",
    "deviceType" : "4fe9cc70e4b014be5cc90246",
    "time" : 1381766225,
    "data" : "3b3616201e100000",
    "snr" : 10.5,
    "status" : 600,
    "message" : "Could not send email to
abcd@test.com",
    "callback" : {
      "subject" : "00A1 - 1381766225 ",
      "message" : "3b3616201e100000 - 001F -
10.50"
    },
    "parameters" : {
      "time" : 1381766225,
      "station" : "001F",
      "data" : "3b3616201e100000",
      "device" : "00A1",
      "snr" : 10.5
    } ],
  "paging" : {
    "next" :
"https://backend.sigfox.com/api/callbacks/messages
/error?offset=3&since=1381766000&deviceTypeId=4fe9
cc70e4b014be5cc90246"
```

```
    }
}
```

Fields:

- `paging`: paging information, if more messages are available
  - `next`: URL of the next «page» of messages
- `data`: the array of device messages error
  - `device`: device identifier
  - `deviceType`: device type identifier
  - `time`: time the message was sent, in seconds since the Unix Epoch
  - `data`: message content, hex encoded
  - `snr`: the SNR of the messages received by the network so far

  Following fields will be present only for errors occurring after the backend release 2.8.0

  - `status`: contains the callback response status.
  - `message`: contains additional information on the response.
  - `callback`: information depends on the callback type.
    - EMAIL:
      - `subject`: the subject of the mail which have been sent
      - `message`: the body of the mail which have been sent
    - HTTP:
      - `url`: the URL called when this message has been processed
      - `headers`: the headers sent in the request. If no header is defined, this field is not present.
      - `body`: the body of the request, if any. It is only present if the request method is POST.
      - `contentType`: the content type of the request. It is only present if the request is a POST.
      - `method`: the HTTP method, currently only GET or POST.
  - `parameters`: All the parameters which have served to build the callback, see CALLBACK doc for an exhaustive list.

Coverage API

Redundancy

Get base station redundancy for a given latitude and longitude.

Request

```
GET
https://backend.sigfox.com/api/coverages/redundanc
y?lat={lat}&lng={lng}&mode={mode}
```

Parameters:

- `lat`: the latitude.
- `lng`: the longitude.
- `mode`: the coverage mode (UNDERGROUND, INDOOR, OUTDOOR). This parameter is optional, default is INDOOR.
  - OUTDOOR: max link budget

- INDOOR: link budget with dB margin
- UNDERGROUND: link budget with dB margin

Response

```
{
"redundancy":3
}
```

Fields:

- `redundancy`: the base station redundancy : 0 = none, 1 = 1 base station, 2 = 2 base stations, 3 = 3 base stations or more

Downlink data variables

Downlink data can contain variables. The variables must be written between '{' and '}'. 3 variables are available :

- **rssi :** the rssi of the uplink message. If the station does not provide it, the value will be equal to 0.
- **tapId :** the id of the station that emitted the downlink answer
- **time :** the time of the downlink message. Note that this variable is not available in roaming because it cannot be computed accurately. It is replaced by 0 in roaming.

User API

User list by Group

Lists all users registered with a role associated to a specific group

Request

```
GET
https://backend.sigfox.com/api/users?groupId={grou
pId}
```

Parameters:

- `groupId`: id of the group used to filter the users to return

Optionally, the request can also have the following parameters (see `next` response field below):

- `limit`: maximum number of messages to return, default 100.
- `offset`: number of messages skipped, used when paginated.

Response

```
{
  "data" : [ {
    "firstName" : "Michel",
    "lastName" : "Dupont",
    "email" : "michel.dupont@sigfox.com",
    "timezone" : "Europe/Paris",
    "creationTime" : 1392812363644,
    "creationDate" : "Wed Feb 19 13:19:23 CET
2014",
    "lastLogin" : 1448351837467,
    "lastLoginDate" : "Tue Nov 24 08:57:17 CET
2015",
    "userRoles" : [ {
      "group" : {
        "id" : "babecafebabecafebabecafe",
        "name" : "Root",
        "nameCI" : "root",
        "description" : "Master Group",
        "path" : [ ],
        "billable" : false
      },
      "customRole" : {
        "id" : "51d19e7ce4b067e859e4c2c1",
        "name" : "SUPPORT_CORP"
      }
    } ]
  }, {
    "firstName" : "Claude",
    "lastName" : "Martin",
    "email" : "claude.martin@sigfox.com",
    "timezone" : "Europe/Paris",
    "creationTime" : 1394118285929,
    "creationDate" : "Thu Mar 06 16:04:45 CET
2014",
    "lastLogin" : 1446456643002,
    "lastLoginDate" : "Mon Nov 02 10:30:43 CET
2015",
    "userRoles" : [ {
      "group" : {
        "id" : "babecafebabecafebabecafe",
        "name" : "Root",
        "nameCI" : "root",
        "description" : "Master Group",
        "path" : [ ],
        "billable" : false
      },
      "customRole" : {
        "id" : "5138e7dfa2f1fffaf25fd407",
        "name" : "SNO [R]"
      }
    }, {
      "group" : {
        "id" : "deadbeeffacecafebabecafe",
        "name" : "SIGFOX_France",
        "nameCI" : "sigfox_france",
        "description" : "Sigfox France SNO Group",
        "path" : [ "babecafebabecafebabecafe" ],
        "billable" : false
      },
      "customRole" : {
        "id" : "539ea35893364266e6202179",
        "name" : "OPT_DOWNLINK"
      }
    } ]
  } ],
  "paging" : {
    "next" :
"https://127.0.0.1:8080/api/users?limit=3&offset=6
"
  }
}
```

Fields:

- `data`: **array of users information records**
- `firstName`: **the user's first name**
- `lastName`: **the user's last name**
- `email`: **the user's email**
- `timezone`: **the user's timezone**
- `creationTime`: **the user's creation date in time**
- `creationDate`: **the user's creation date**
- `lastLogin`: **the user's last connection date in time**
- `lastLoginDate`: **the user's last connection date**
- `userRoles`: **array of custom roles associated to this user through groups**
  - `group`: **the group's information**
    - `id`: **the group's identifier**
    - `name`: **the group's name**
    - `nameCI`: **the group's name case insensitive**
    - `description`: **description of this group**

    - `path`: Array with the group’s parents’ identifier
    - `billable`: true if the group is billable
  - `customRole`: the custom role’s information
    - `id`: the custom role’s identifier
    - `name`: the custom role’s name
- `paging`: paging information, if more users are available
- `next`: URL of the next «page» of users