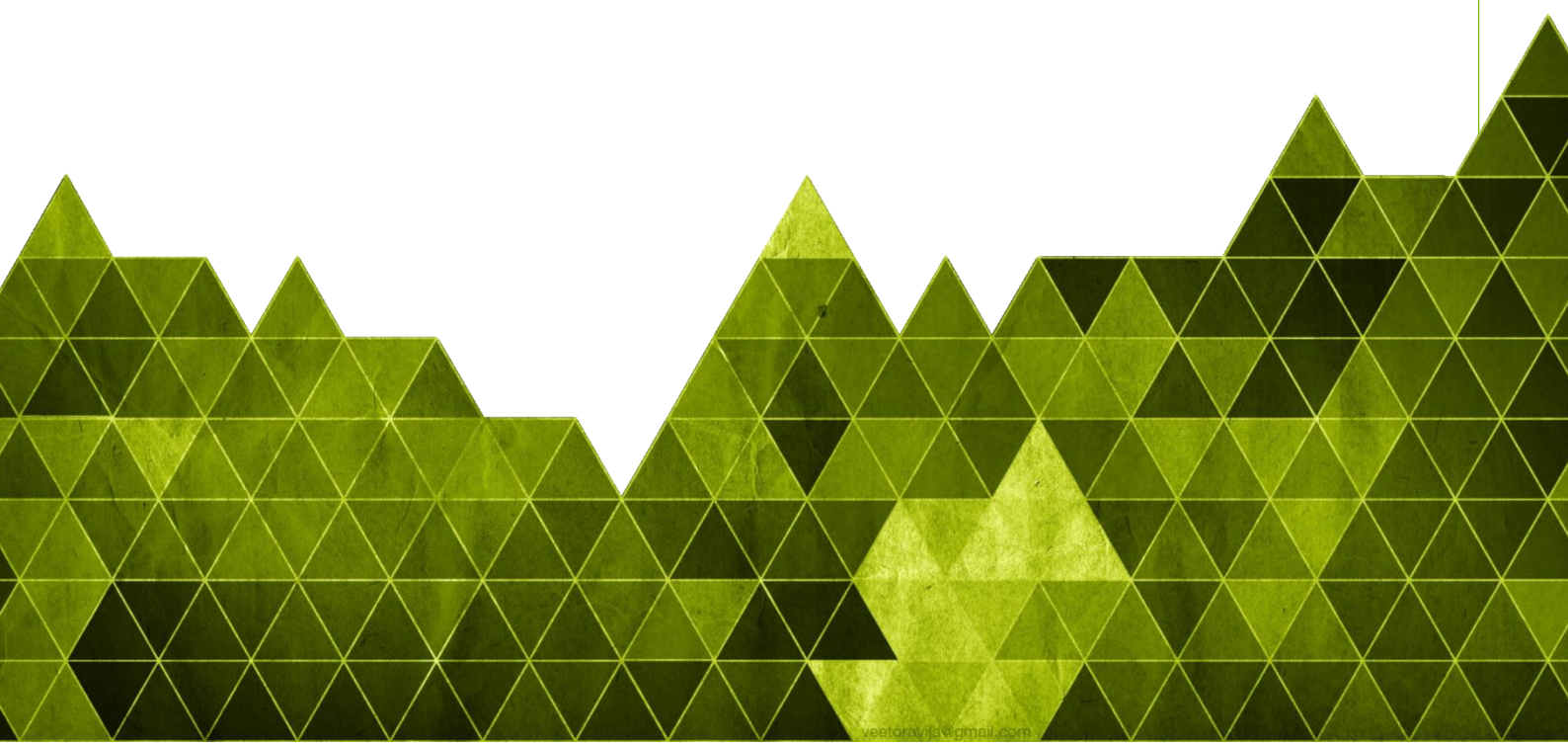
ioThink Solutions

API documentation

API Version: 1.16.1



Index

1 Introduction

Kheiron Service Platform API exposes a set of HTTPS web services based on Kheiron Service Platform application. As the API is a dedicated feature, it must be activated by the administrator before being accessible. Each request to a non-activated API feature will result a 404 Not Found or 401 Unauthorized response.

> **Important note**
>
> Since v1.12.4: the API restrict resource accesses matching access group configurations. Please make sure that the user using the API his configured with the right access group.

2 Generalities

2.1 API path

All URL requests must start with the following path

```
https://api.kheiron-sp.io/v1
```

In order to simplify the notation in this document, we will indicate this root path using the '~' char.

2.2 Authentication token

KSP API use bearer token based authentication to proceed requests. In order to get a valid token, you need to authenticate with a valid user account (**username** and **password**) by requesting the token at the following URL with one of the following methods.

HTTP

```
POST /token HTTP/1.1
Host: api.kheiron-sp.io
Content-Type: application/x-www-form-urlencoded
Cache-Control: no-cache

grant_type=password&username=username&password=password
```

cURL

```
curl -X POST \
  https://api.kheiron-sp.io/token \
  -H 'Cache-Control: no-cache' \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  -d 'grant_type=password&username=username&password=password'
```

RESPONSE (JSON)

```
{
   "access_token": "rHzHRiPrH3t6V5Trxi64……RbmiacBuCNgA47RPZ",
   "token_type": "bearer",
   "expires_in": 86399,
   "userName": "username",
   ".issued": "Mon, 12 Mar 2018 15:11:08 GMT",
   ".expires": "Tue, 13 Mar 2018 15:11:08 GMT"
}
```

2.3 Requests authentication

After getting the token, each request to the API must include the **bearer token** in the Header of the request using one of the following formats.

HTTP
Authorization: bearer **token**

cURL
-H 'Authorization: bearer **token**

2.4 Responses

2.4.1 Formats

KSP API supports both JSON and XML formats. You can specify the desired response format using header parameter **accept** with **application/xml** or **application/json** values.

2.4.2 Status

The following status codes can be return.

- OK (200): Request succeed. The request body contains the result of the request as described in topics below.
- BAD REQUEST (400): The request was not proceeded due to invalid parameters. The body contains a more detailed description of the error.
- UNAUTHORIZED (401): The request was not proceeded due to access violation.
- NOT FOUND (404): The request was not proceeded. The URL was not found or the API feature is not activated for your account and/or contract.
- INTERNAL SERVER ERROR (500): The request was not proceeded due to internal server failure.

2.5 Timestamp

All timestamps are based on 1st January 2000 at 00:00:00. When specifying a timestamp as parameter or retrieving a timestamp from the response, be sure that you refer to the 1st January 2000 at 00:00:00.

Timestamps can refer to Local time or UTC time. Reference is specified on methods descriptions below.

3 Services

3.1 Contracts

Section relative to contract API methods.

> **Important note**
>
> Naming convention has changed and relatively to this documentation the term **Contract** designates an **Application**.

3.1.1 GET Contracts

GET	~/contracts

Parameters: None

REPONSE (MODEL)

Contracts: Array[Contract]

Contract:

- Id: (string) Contract unique identifier
- Reference: (string) Contract reference
- Company: (string) Company name
- Details: (string) Details
- ContactEmail: (string) Contract contact email

RESPONSE (JSON)

```
{
   "contracts": [
      {
        "id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx",
        "reference": "CT001",
        "company": "Contonso",
        "details": "Contonso .Inc Contract",
        "contactEmail": "admin@contonso.com"
      },
      {
        "id": " xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx ",
        "reference": "MI001",
        "company": "Mira",
        "details": "Mira Contract",
        "contactEmail": "it@mira.com"
      }
   ]
}
```

3.2 Devices

Section relative to device API methods.

> **Important note**
>
> Naming convention has changed and relatively to this documentation the term **Device** designates an **DigitalTwin**.

3.2.1 GET devices

GET	~/devices

Parameters:

- contractId: (string) The contract identifier

REPONSE (MODEL)

Devices: Array[Device]

Device:

- Id: (string) Device unique identifier
- Name: (string) Device name
- Details: (string) Device details
- Status: (int) Device status [0 = active, 1 = inactive, 2 = suspended]
- Timezone: (string) Device timezone reference

RESPONSE (JSON)

```
{
   "devices": [
      {
         "id": "3219875446",
         "name": "MyDevice1",
         "details": "My device 1 details",
         "status": 0,
         "timezone": "UTC"
      },
      {
         "id": "14587",
         "name": "MyDevice2",
         "details": "My device 2 details",
         "status": 0,
         "timezone": "(UTC+01:00) Bruxelles, Copenhague, Madrid, Paris"
      }
   ]
}
```

3.2.2 GET device

GET	~/device

Parameters:

- contractId: (string) The contract identifier
- deviceId: (string) The device identifier

REPONSE (MODEL)

Device: Device

Device:

- Id: (string) Device unique identifier
- Name: (string) Device name
- Details: (string) Device details
- Status: (int) Device status [0 = active, 1 = inactive, 2 = suspended]
- Timezone: (string) Device timezone reference

RESPONSE (JSON)

```
{
   "device": {
      "id": "3219875446",
      "name": "MyDevice1",
      "details": "My device 1 details",
      "status": 0,
      "timezone": "UTC"
   }
}
```

3.2.3 GET device real-time logs

This method returns the current real-time logs for the corresponding device id. The real-time logs values returned are the last proceeded by KSP. Values returned must not be considered has absolute real-time values has it depends on communication between the device and the platform.

GET	~/devices/realtimes

Parameters:

- contractId: (string) The contract identifier
- deviceId: (string) The device identifier
- tagReferences: (Array<string>) The tag reference collection [Optional]

REPONSE (MODEL)

Logs: Array[RealtimeLog]

RealtimeLog:

- TagReference: (string) tag reference
- Value: (string) value
- Timestamp: (int64) refresh timestamp (UTC time)
- IsEvent: (Boolean) true is the tag is defined as event

RESPONSE (JSON)

```
{
   "logs": [
      {
         "tagReference": "TAG_1",
         "value": "24",
         "timestamp": 578590638,
         "isEvent": false
      },
      {
         "tagReference": "TAG_2",
         "value": "2587",
         "timestamp": 578590638,
         "isEvent": false
      }
   ]
}
```

3.2.4 GET multiple devices real-time logs

This method returns a set of current real-time logs for the corresponding device identifiers. The real-time logs values returned are the last proceeded by KSP. Values returned must not be considered has absolute real-time values has it depends on communication between the device and the platform.

GET	~/devices/realtimes

Parameters:

- contractId: (string) The contract identifier
- deviceIdentifiers: (Array<string>) the devices identifier collection
- tagReferences: (Array<string>) The tag reference collection [Optional]

REPONSE (MODEL)

Logs: Array[RealtimeLog]

RealtimeLog:

- DeviceIdentifier: (string) The identifier of the corresponding device
- TagReference: (string) tag reference
- Value: (string) value
- Timestamp: (int64) refresh timestamp (UTC time)
- IsEvent: (Boolean) true is the tag is defined as event

RESPONSE (JSON)

```
{
   "logs": [
      {
         "deviceIdentifier": "device1",
         "tagReference": "TAG_1",
         "value": "24",
         "timestamp": 578590638,
         "isEvent": false
      },
      {
         "deviceIdentifier": "device2",
         "tagReference": "TAG_2",
         "value": "2587",
         "timestamp": 578590638,
         "isEvent": false
      }
   -
```

3.2.5 GET device historic logs

Historics request return a set of historics for the corresponding device id and the set times (start and optionally end). As the result can contain large amount of data, the request result can be batched. The batch size is defined to 10000 logs. If the result log count exceeds 10000 logs, the parameter "Next" is added to the response. This parameter contains the URL for the next request which will return the next logs statements.

GET	~/devices/historics

Parameters:

- contractId: (string) the contract identifier
- deviceId: (string) the device identifier
- startTime: (string) start timestamp (Local time)
- endTime: (string) end timestamp [Optional] (Local time)
- tagReferences: (Array<string>) the tag reference collection [Optional]

REPONSE (MODEL)

Historics: Array[HistoricLog]
Next: (string) the next URL request

HistoricLog:

- TagReference: (string) The tag reference
- Logs: Array[Log]

Log:

- Value: (string) value
- Timestamp: (int64) log timestamp (Local time)
- Source: (int) log source [1 = Local (device), 2 = Remote (platform)]

RESPONSE (JSON)

```
{
   "historics": [
      {
         "tagReference": "TAG_1",
         "logs": [
            {
               "value": "1224",
               "timestamp": 577929633,
               "source": 1
            },
            ...
         ]
      },
      ...
   ]
}
```

3.2.6 GET multiple devices historic logs

Historics request return a set of historics for all corresponding devices identifiers and the set times (start and optionally end). As the result can contain large amount of data, the request result can be batched. The batch size is defined to 10000 logs. If the result log count exceeds 10000 logs, the parameter "Next" is added to the response. This parameter contains the URL for the next request which will return the next logs statements.

GET	~/devices/historics

Parameters:

- contractId: (string) the contract identifier
- deviceIdentifiers: (Array<string>) the devices identifier collection
- startTime: (string) start timestamp (Local time)
- endTime: (string) end timestamp [Optional] (Local time)
- tagReferences: (Array<string>) the tag reference collection [Optional]

REPONSE (MODEL)

Historics: Array[HistoricLog]
Next: (string) the next URL request

HistoricLog:

- TagReference: (string) The tag reference
- DeviceIdentifier: (string) The identifier of the corresponding device
- Logs: Array[Log]

Log:

- Value: (string) value
- Timestamp: (int64) log timestamp (Local time)

RESPONSE (JSON)

```
{
   "historics": [
      {
         "tagReference": "TAG_1",
         "deviceIdentifier": "device1",
         "logs": [
            {
               "value": "1224",
               "timestamp": 577929633,
               "source": 1
            },
            ...
         ]
      },
      ...
   ]
}
```

3.2.7 GET device event logs

Events request return a set of events for the corresponding device id and the set times (start and optionally end). As the result can contain large amount of data, the request result can be batched. The batch size is defined to 10000 logs. If the result log count exceeds 10000 logs, the parameter "Next" is added to the response. This parameter contains the URL for the next request which will return the next logs statements.

GET	~/devices/events

Parameters:

- contractId: (string) the contract identifier
- deviceId: (string) the device identifier
- startTime: (string) start timestamp (Local time)
- endTime: (string) end timestamp [Optional] (Local time)
- tagReferences: (Array<string>) the tag reference collection [Optional]

REPONSE (MODEL)

Events: Array[EventLog]
Next: (string) the next URL request

EventLog:

- TagReference: (string) tag reference
- Logs: Array[Log]

Log:

- Value: (bool) value (true = active, false = inactive)
- Timestamp: (int64) log timestamp (Local time)
- Source: (int) log source [1 = Local (device), 2 = Remote (platform)]
- Status: (int) event status [0 = none, 1 = Acknowledged]

RESPONSE (JSON)

```
{
  "events": [
    {
      "tagReference": "e_TAG_1",
      "logs": [
        {
          "value": false,
          "timestamp": 560598370,
          "source": 2,
          "status": 0
        },
        ...
      ]
    }
  ]
}
```

3.2.8 POST tag writing

This method allows device tag write. This enable External tag write the same way as it's performed through Kheiron Dashboard. Consider that a call to this method can generate downlink messages.

POST	~/devices/tags/write

Parameters:

- contractId: (string) The contract identifier
- deviceId: (string) The device identifier
- tagRef: (string) The tag reference
- value: (string) The value

REPONSE (OK)

Operation can take several seconds if a downlink message must be proceeded. If operation succeed method return HTTP Code 200 (OK).

3.2.9 POST tag writing multiple devices

This method allows tag writing for multiple devices at the same time. This enable External tag write the same way as it's performed through Kheiron Dashboard. Consider that a call to this method can generate downlink messages.

POST	~/devices/tags/write

Parameters:

- contractId: (string) The contract identifier

REQUEST BODY (MODEL)

RealTimeLog:

- TagReference: (string) The tag reference
- DeviceIdentifier: (string) The corresponding Device identifier
- Value : (string) Value

REQUEST BODY (JSON)

```
[
   {
      "tagReference": "reference1",
      "deviceIdentifier": "1",
      “value”: 10
   },
   {
      "tagReference": "reference2",
      "deviceIdentifier": "1",
      “value”: “stringValue”
   },
   {
      "tagReference": "reference3",
      "deviceIdentifier": "2",
      “value”: “{\”value1\”:10, \”value2\”:\”test\”}”
   }
]
```

REPONSE (OK)

Operation can take several seconds. If operation succeed method return HTTP Code 200 (OK).

3.2.10 POST multiple devices historic logs

Historics request return a set of historics for all corresponding devices identifiers and the set times (start and optionally end). As the result can contain large amount of data, the request result can be batched. The batch size is defined to 10000 logs. If the result log count exceeds 10000 logs, the parameter "Next" is added to the response. This parameter contains the URL for the next request which will return the next logs statements.

POST	~/devices/historics

Parameters:

- contractId: (string) The contract identifier

REQUEST BODY (MODEL)

DeviceLog:

- DeviceIdentifier: (string) The corresponding Device identifier
- StartTime: (string) start timestamp (Local time)
- EndTime: (string) end timestamp [Optional] (Local time)
- tagReferences: (Array<string>) the tag reference collection [Optional]

REQUEST BODY (JSON)

```
{
  "DeviceIdentifier": ["1", "2"],
  "StartTime": 577929633,
  "TagReference": ["TAG_1", " TAG_2"]
}
```

RESPONSE (JSON)

```
{
  "historics": [
    {
      "tagReference": "TAG_1",
      "deviceIdentifier": "1",
      "logs": [
        {
          "value": "1224",
          "timestamp": 577929633,
          "source": 1
        },
        ...
      ]
    },
    ...
  ]
}
```

3.2.11 POST multiple devices real-time logs

This method returns a set of current real-time logs for the corresponding device identifiers. The real-time logs values returned are the last proceeded by KSP. Values returned must not be considered as absolute real-time values as it depends on communication between the device and the platform.

POST	~/devices/realtimes

Parameters:

- contractId: (string) The contract identifier

REQUEST BODY (MODEL)

RealTimeLog:

- DeviceIdentifier: (string) The corresponding Device identifier
- TagReference: (Array) The tags reference

REQUEST BODY (JSON)

```
{
   "DeviceIdentifier" : ["1, 2"],
   "TagReference" : ["TAG_1, TAG_2"]
}
```

RESPONSE (JSON)

```
{
   "logs": [
      {
         "deviceIdentifier" :  "1",
         "tagReference": "TAG_1",
         "value": "24",
         "timestamp": 578590638,
         "isEvent": false
      },
      {
         "deviceIdentifier" : "2",
         "tagReference": "TAG_2",
         "value": "2587",
         "timestamp" : 578590638,
         "isEvent" : false
      }
   ]
}
```

3.2.12 POST multiple devices events logs

Events request return a set of events for the corresponding device id and the set times (start and optionally end). As the result can contain large amount of data, the request result can be batched. The batch size is defined to 10000 logs. If the result log count exceeds 10000 logs, the parameter "Next" is added to the response. This parameter contains the URL for the next request which will return the next logs statements.

POST	~/devices/events

Parameters:

- contractId: (string) The contract identifier

REQUEST BODY (MODEL)

EventLog:

- DeviceIdentifier: (string) The corresponding Device identifier
- StartTime: (string) start timestamp (Local time)
- EndTime: (string) end timestamp [Optional] (Local time)
- tagReferences: (Array<string>) the tag reference collection [Optional]

REQUEST BODY (JSON)

```
{
  "DeviceIdentifier": "1",
  "StartTime": 1633020442,
  "EndTime": 1633106842,
  "tagReference": ["TAG_1", " TAG_2"]
}
```

RESPONSE (JSON)

```
{
  "events": [
    {
      "tagReference": "TAG_1",
      "logs": [
        {
          "value": false,
          "timestamp": 560598370,
          "source": 2,
          "status": 0
        },
        ...
      ]
    }
  ]
}
```

3.2.13 PUT historic logs

This method allows device historic logs injection.

PUT	~/devices/historics/add

Parameters:

- contractId: (string) The contract identifier
- deviceId: (string) The device identifier

Body:

- array[HistoricLog]

REQUEST BODY (MODEL)

HistoricLog:

- TagReference: (string) The tag reference
- Logs: Array[Log]

Log:

- Value: (string) value
- Timestamp: (int64) log timestamp (Local time)

REQUEST BODY (JSON)

```
[
  {
    "tagReference": "log1",
    "logs": [
      {
        "value": "1",
        "timestamp": 646155300
      },
      {
        "value": "",
        "timestamp": 646155900
      }
    ]
  }
]
```

REPONSE (OK)

Operation can take several seconds. If operation succeed method return HTTP Code 200 (OK).

3.2.14 PUT historic logs multiple devices

This method allows historic logs injection on multiple devices at the same time.

PUT	~/devices/historics/add

Parameters:

- contractId: (string) The contract identifier

Body:

- array[HistoricLog]

REQUEST BODY (MODEL)

HistoricLog:

- TagReference: (string) The tag reference
- DeviceIdentifier: (string) The corresponding Device identifier
- Logs: Array[Log]

Log:

- Value: (string) value
- Timestamp: (int64) log timestamp (Local time)

REQUEST BODY (JSON)

```
[
  {
    "tagReference": "log1",
    "deviceIdentifier": "150",
    "logs": [
      {
        "value": "1",
        "timestamp": 646155300
      },
      {
        "value": "15",
        "timestamp": 646155900
      }
    ]
  }
```

REPONSE (OK)

Operation can take several seconds. If operation succeed method return HTTP Code 200 (OK).

3.2.15 PUT event logs

This method allows device event logs injection.

PUT	~/devices/events/add

Parameters:

- contractId: (string) The contract identifier
- deviceId: (string) The device identifier

Body:

- array[EventLog]

REQUEST BODY (MODEL)

EventLog:

- TagReference: (string) The tag reference
- Logs: Array[Log]

Log:

- Value: (bool) value (true = active, false = inactive)
- Timestamp: (int64) log timestamp (Local time)

REQUEST BODY (JSON)

```
[
  {
    "tagReference": "event1",
    "logs": [
      {
        "value": true,
        "timestamp": 646155300
      },
      {
        "value": false,
        "timestamp": 646155900
      }
    ]
  }
]
```

REPONSE (OK)

Operation can take several seconds. If operation succeed method return HTTP Code 200 (OK).

3.2.16 DELETE historic logs

This method allows device historic logs deletion.

DELETE	~/devices/historics/delete

Parameters:

- contractId: (string) The contract identifier
- deviceId: (string) The device identifier
- tagReferences: (Array<string>) the tag reference collection
- startTime: (string) start timestamp (Local time)
- endTime: (string) end timestamp (Local time)

REPONSE (OK)
Operation can take several seconds. If operation succeed method return HTTP Code 200 (OK).

3.2.17 DELETE historic logs multiple devices

This method allows historic logs deletion on multiple devices at the same time.

DELETE	~/devices/historics/delete

Parameters:

- contractId: (string) The contract identifier
- deviceIdentifiers: (Array<string>) The devices identifier collection
- tagReferences: (Array<string>) the tag reference collection
- startTime: (string) start timestamp (Local time)
- endTime: (string) end timestamp (Local time)

REPONSE (OK)
Operation can take several seconds. If operation succeed method return HTTP Code 200 (OK).

3.3 Groups

3.3.1 GET groups

GET	~/groups

Parameters:

- contractId: (string) The contract identifier

REPONSE (MODEL)

Groups: Array[Group]

Group:

- Id: (string) Group unique identifier
- Name: (string) Group name
- Details: (string) Group details
- ParentId: (string) Group parent identifier

RESPONSE (JSON)

```
{
   "groups": [
      {
         "id": "45874",
         "name": "Group 1 ",
         "details": "The first group",
         "parentId": ""
      },
      …
   ]
}
```

3.4.1 GET group

GET	~/group

Parameters:

- contractId: (string) the contract identifier
- groupId: (string) the group identifier

REPONSE (MODEL)

Group: (Group)

Group:

- Id: (string) Group unique identifier
- Name: (string) Group name
- Details: (string) Group details
- ParentId: (string) Group parent identifier
- Children: (array<Group>) The children groups
- Devices: (array<Device>) The group devices *(see device section for description)*

RESPONSE (JSON)

```
{
  "group": {
    "id": "436988",
    "name": "Sondes d'ambiance",
    "details": "Températures",
    "parentId": null,
    "children": [
      {
        "id": "1",
        "name": "Sondes R-1",
        "details": "Sondes R-1",
        "parentId": "45",
        "children": [],
        "devices": [
          {
            "id": "7",
            "name": "Bureau 005",
            "details": "70B3D54750100327",
            "status": 0,
            "timezone": "(UTC+01:00) Bruxelles, Copenhague, Madrid, Paris"
          },
          ...
        ]
      }
    ],
    "devices": []
  }
}
```