Yeelink

API文档

设备devices

一个device表示一组传感器的集合.

创建设备

URL http://api.yeelink.net/v1.0/devices
数据格式 JSON
Method POST
返回 新设备的ID
对该URL的一个HTTP POST请求将为您创建一个新的设备.

请求实例 (运用curl):
curl --request POST --data-binary @datafile.txt --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/devices
设备信息放到datafile.txt中,需要在HTTP Header中增加API Key来授权写入操作, 具体请参照API Key.

数据例子(JSON):

```
{
  "title":"test",
  "about":"test api",
  "tags":["temperature","lab"],
  "location":{
    "local":"Qingdao",
    "latitude":0.444,
    "longitude":0.555
  }
}
```

返回(JSON)

```
{
  "device_id": 2
}
```

编辑设备

URL http://api.yeelink.net/v1.0/device/<device_id>
数据格式 JSON
Method PUT
返回 HTTP Headers only
对该URL的一个HTTP PUT请求将更新指定设备的信息,其中<device_id>为所要更新的设备的id. 若您的客户端不支持PUT请求, 请使用url方式模拟, 具体参照模拟PUT, DELETE请求.

请求实例 (运用curl):
curl --request PUT --data-binary @datafile.txt --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12
只需将要修改的设备信息放到datafile.txt中,需要在HTTP Header中增加API Key来授权写入操作, 具体请参照API Key.

数据例子(JSON):

```
{
  "title":"test",
  "about":"just a test",
  "tags":["lab","temperature"],
  "location":{
    "local":"Qingdao",
    "latitude":0.444,
    "longitude":0.555
  }
}
```

罗列设备

URL http://api.yeelink.net/v1.0/devices
数据格式 JSON
Method GET
返回 用户所有的设备
对该URL的一个HTTP GET请求将得到所有设备信息的列表.

请求实例 (运用curl):
curl --request GET --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/devices
需要在HTTP Header中增加API Key来授权GET操作, 具体请参照API Key.

返回(JSON)

```
[
  {
    "id": 2,
    "title": "test2",
    "about": "just a test"
  },
  {
    "id": 3,
    "title": "test3",
    "about": "just a test"
  }
]
```

查看设备

URL http://api.yeelink.net/v1.0/device/<device_id>

数据格式 JSON
Method GET
返回 请求的设备信息
对该URL的一个HTTP GET请求将得到所要查看设备的详细内容,其中<device_id>为所要查看的设备的id.

请求实例 (运用curl):
curl --request GET --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12
需要在HTTP Header中增加API Key来授权GET操作, 具体请参照API Key.

返回(JSON)

```
{
  "title": "test3",
  "about": "just a test",
  "tags": "lab",
  "local": "Qingdao",
  "latitude": 0.444,
  "longitude": 0.555
}
```

删除设备

URL http://api.yeelink.net/v1.0/device/<device_id>
数据格式 JSON
Method DELETE
返回 HTTP Headers only
对该URL的一个HTTP DELETE请求将将删除指定的设备,其中<device_id>为所要删除的设备的id. 若您的客户端不支持DELETE请求, 请使用url方式模拟, 具体参照模拟PUT, DELETE请求.

请求实例 (运用curl):
curl --request DELETE --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12
需要在HTTP Header中增加API Key来授权DELETE操作, 具体请参照API Key.

传感器sensors

传感器完成采集数据的功能, 一个设备支持多个传感器.

创建传感器

URL http://api.yeelink.net/v1.0/device/<device_id>/sensors
数据格式 JSON
Method POST
返回 新传感器的ID
对该URL的一个HTTP POST请求将为您指定的设备创建一个新的传感器, 其中<device_id> 为指定设备的id.

请求实例 (运用curl):
curl --request POST --data-binary @datafile.txt --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12/sensors
传感器信息放到datafile.txt中,需要在HTTP Header中增加API Key来授权写入操作, 具体请参照API Key.

数据例子(JSON):

数值型传感器格式如下:

```
{
  "type":"value",
  "title":"test",
  "about":"test api",
  "tags":["tag1","tag2"],
  "unit": {
     "name": "temperature",
     "symbol": "C"
   }
}
```

gps型传感器格式如下:

```
{
  "type":"gps",
  "title":"test",
  "about":"test api",
  "tags":["tag1","tag2"]
}
```

泛型传感器格式如下:

```
{
  "type":"gen",
  "title":"test",
  "about":"test api",
  "tags":["tag1","tag2"]
}
```

图像型传感器格式如下:

```
{
  "type":"photo",
  "title":"test",
```

```
  "about":"test api",
  "tags":["tag1","tag2"]
}
```

若不带"type", 默认为数值型传感器.

返回(JSON)

```
{
  "sensor_id": 2
}
```

编辑传感器

URL http://api.yeelink.net/v1.0/device/<device_id>/sensor/<sensor_id>
数据格式 JSON
Method PUT
返回 HTTP Headers only
对该URL的一个HTTP PUT请求将更新传感器的信息, 其中<device_id>为所要更新的传感器所属设备的id,<sensor_id>为所要更新的传感器的id. 若您的客户端不支持PUT请求, 请使用url方式模拟, 具体参照模拟PUT, DELETE请求.

请求实例 (运用curl):
curl --request PUT --data-binary @datafile.txt --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12/sensor/3
只需将要修改的传感器信息放到datafile.txt中, 需要在HTTP Header中增加API Key来授权写入操作, 具体请参照API Key.

数据例子(JSON):

数值型传感器格式如下:

```
{
  "title":"test",
  "about":"test api",
  "tags":["tag1","tag2"],
  "unit": {
     "name": "temperature",
     "symbol": "C"
   }
}
```

gps型传感器格式如下:

```
{
  "title":"test",
  "about":"test api",
  "tags":["tag1","tag2"]
}
```

泛型传感器格式如下:

```
{
  "title":"test",
  "about":"test api",
  "tags":["tag1","tag2"]
}
```

图像型传感器格式如下:

```
{
  "title":"test",
  "about":"test api",
  "tags":["tag1","tag2"]
}
```

罗列传感器

URL http://api.yeelink.net/v1.0/device/<device_id>/sensors
数据格式 JSON
Method GET
返回 该设备的所有传感器信息
对该URL的一个HTTP GET请求将得到所有属于某一设备传感器的列表, 为所要罗列的传感器所属的设备id.

请求实例 (运用curl):
curl --request GET --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12/sensors
需要在HTTP Header中增加API Key来授权GET操作, 具体请参照API Key.

返回(JSON)

```
[
  {
    "id": 2,
    "title": "test2",
    "about": "just a test"
  },
  {
    "id": 3,
    "title": "test3",
    "about": "just a test"
  }
]
```

查看传感器

URL http://api.yeelink.net/v1.0/device/<device_id>/sensor/<sensor_id>
数据格式 JSON
Method GET
返回 请求的传感器信息
对该URL的一个HTTP GET请求将得到所要查看传感器的详细内容, 其中<sensor_id>为所要查看的传感器的id,<device_id>为此传感器所属的设备的id.

请求实例 (运用curl):
curl --request GET --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12/sensor/3
需要在HTTP Header中增加API Key来授权GET操作, 具体请参照API Key.

返回(JSON)

数值型传感器格式如下:
```
{
  "title":"test",
  "about":"just a test",
  "tags":["tag1","tag2"],
  "unit_name": "temperature",
  "unit_symbol": "C"
}
```

gps型传感器格式如下:
```
{
  "title":"test",
  "about":"test api",
  "tags":["tag1","tag2"]
}
```

泛型传感器格式如下:
```
{
  "title":"test",
  "about":"test api",
  "tags":["tag1","tag2"]
}
```

图像型传感器格式如下:
```
{
  "title":"test",
  "about":"test api",
  "tags":["tag1","tag2"]
}
```

删除传感器

URL http://api.yeelink.net/v1.0/device/<device_id>/sensor/<sensor_id>
数据格式 JSON
Method DELETE
返回 HTTP Headers only
对该URL的一个HTTP DELETE请求将删除指定的传感器, 其中<sensor_id>为所要删除的传感器的id, <device_id>为该传感器所属的设备的id. 若您的客户端不支持DELETE请求, 请使用url方式模拟, 具体参照模拟PUT, DELETE请求.

请求实例 (运用curl):
curl --request DELETE --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12/sensor/3
需要在HTTP Header中增加API Key来授权DELETE操作, 具体请参照API Key.

数据点datapoints

一个datapoint是由key和value组成的键值对.

创建数据点

URL http://api.yeelink.net/v1.0/device/<device_id>/sensor/<sensor_id>/datapoints
数据格式 JSON
Method POST
返回 HTTP Headers only
对该URL的一个HTTP POST请求会为指定的传感器创建一个新的数据点, 使用此API来为传感器存储历史数据.

一个datapoint是由key和value组成的键值对.

数值型传感器 key为timestamp, value为数值;

gps型传感器 key为timestamp, value为JSON格式, 由经度、纬度、速度等GPS信息组成; 同时可以指定是否修正地图上显示的偏移.

泛型传感器 key为用户自定义字符串, 大小限定为128个字符; value为JSON格式,由用户自定义具体内容, 大小限定为1024个字符.

图像型传感器 key为timestamp, value为图像二进制信息;

微博抓取器 不支持;

Note: key为唯一索引; timestamp为ISO 8601标准时间格式(默认时区为中国标准时间CST), 例如: 2012-03-15T16:13:14.

对于数值型传感器和gps型传感器, 如果上传的数据和历史数据的timestamp相同, 则会被丢弃. 若未指定timestamp, 服务器会自动加上当前时间.

请求实例 (运用curl):
curl --request POST --data-binary @datafile.txt --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12/sensor/3/datapoints
需要在HTTP Header中增加API Key来授权写入操作, 具体请参照API Key. 支持一次传送一个数据或者批量上传.

Note: 目前限定相邻数据上传间隔须大于等于10s, 过于频繁的请求会收到406 Response.

单个上传数据例子(JSON):

数值型传感器格式如下:
```
{
  "timestamp":"2012-03-15T16:13:14",
  "value":294.34
}
```

gps型传感器格式如下:
```
{
  "timestamp":"2012-03-15T16:13:14",
  "value":{"lat":35.4567,"lng":46.1234,"speed":98.2}
}
```
gps型传感器格式(指定需要做偏移修正)如下:
```
{
  "timestamp":"2012-03-15T16:13:14",
  "value":{"lat":35.4567,"lng":46.1234,"speed":98.2,"offset":"yes"}
}
```

泛型传感器格式如下:
```
{
  "key":"e10adc3949ba59abbe56e037f20f884e",
  "value":{...}
}
```

批量上传数据例子(JSON):

数值型传感器格式如下:
```
[
  {"timestamp": "2012-06-15T14:00:00", "value":315.01},
  {"timestamp": "2012-06-15T14:00:10", "value":316.23},
  {"timestamp": "2012-06-15T14:00:20", "value":317.26},
  {"timestamp": "2012-06-15T14:00:30", "value":318},
  {"timestamp": "2012-06-15T14:00:40", "value":317}
]
```

gps型传感器格式如下:
暂时不支持.

泛型传感器格式如下:
暂时不支持

编辑数据点

URL http://api.yeelink.net/v1.0/device/<device_id>/sensor/<sensor_id>/datapoint/<key>
数据格式 JSON
Method PUT
返回 HTTP Headers only
对该URL的一个HTTP PUT请求将更新指定数据,其中<device_id>为所要更新的数据所属设备的id,<sensor_id>为所要更新的数据所属的传感器的id.若您的客户端不支持PUT请求, 请使用url方式模拟, 具体参照模拟PUT, DELETE请求.

<key>:
对于数值型传感器和gps型传感器, <key>为所要更新数据的时间戳(ISO 8601格式).
请求实例 (运用curl):
curl --request PUT --data-binary @datafile.txt --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12/sensor/3/d

对于泛型传感器, <key>为用户自定义的字符串, 在请求的URL中需要做urlencode.
请求实例 (运用curl):
curl --request PUT --data-binary @datafile.txt --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12/sensor/3/d

微博抓取器 不支持;

只需将要修改的数据值放到datafile.txt中,需要在HTTP Header中增加API Key来授权写入操作, 具体请参照API Key.

数据例子(JSON):

数值型传感器格式如下:
```
{
  "value": 39.4
}
```

gps型传感器格式如下:
```
{
```

```
  "value": {"lat":35.4321,"lng":46.3451,"speed":98.2}
}
```

泛型传感器格式如下：

```
{
  "value": {...}
}
```

查看数据点

URL http://api.yeelink.net/v1.0/device/<device_id>/sensor/<sensor_id>/datapoint/<key>
数据格式 JSON
Method GET
返回 数据点信息
对该URL的请求返回指定key的datapoint, 若未指定key, 则返回该sensor的最新数据.

<key>:
对于数值型传感器和gps型传感器, <key>为所要查看数据的时间戳(ISO 8601格式).
请求实例 (运用curl):

```
curl --request GET --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12/sensor/3/datapoint/2012-03-15T16:13:14
```

对于泛型传感器, <key>为用户自定义的字符串, 在请求的URL中需要做urlencode.
请求实例 (运用curl):

```
curl --request GET --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12/sensor/3/datapoint/e10adc3949ba59abbe5
```

微博抓取器 不支持参数key;

返回(JSON)

数值型传感器格式如下：

```
{
  "value": 39.4
}
```

gps型传感器格式如下：

```
{
  "value": {"lat":35.4321,"lng":46.3451,"speed":98.2}
}
```

泛型传感器格式如下：

```
{
  "value": {...}
}
```

微博抓取器格式如下：

```
{
  "status_cnt":0,"follower_cnt":0,"msg_cnt":0
}
```

note: status_cnt未读微博数, follower_cnt：新增好友数, msg_cnt：新消息数

若未指定key:
curl --request GET --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12/sensor/3/datapoints

返回(JSON)

数值型传感器格式如下：

```
{
  "timestamp":"2012-03-15T16:13:14",
  "value": 39.2
}
```

gps型传感器格式如下：

```
{
  "timestamp":"2012-03-15T16:13:14",
  "value": {"lat":35.4321,"lng":46.3451,"speed":98.2}
}
```

泛型传感器格式如下：

```
{
  "key":"e10adc3949ba59abbe56e037f20f884e",
  "value": {...}
}
```

需要在HTTP Header中增加API Key来授权GET操作, 具体请参照API Key.

删除数据点

URL http://api.yeelink.net/v1.0/device/<device_id>/sensor/<sensor_id>/datapoint/<key>
数据格式 JSON
Method DELETE
返回 HTTP Headers only
对该URL的一个HTTP DELETE请求将删除指定<key>的数据, 其中<sensor_id>为所要删除的数据所属传感器的id,<device_id>为所要删除的数据所属设备的id. 若您的客户端不支持DELETE请求, 请使用url方式模拟, 具体参照模拟PUT, DELETE请求.

<key>:
对于数值型传感器和gps型传感器, <key>为所要更新数据的时间戳(ISO 8601格式).
请求实例 (运用curl):
curl --request DELETE --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12/sensor/3/datapoint/2012-03-15T16:13

对于泛型传感器, <key>为用户自定义的字符串, 在请求的URL中需要做urlencode.
请求实例 (运用curl):
curl --request DELETE --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12/sensor/3/datapoint/e10adc3949ba59ab

微博抓取器 不支持

需要在HTTP Header中增加API Key来授权DELETE操作, 具体请参照API Key.

图像数据photos

图像信息是由key和value组成的键值对, 图像内容则是二进制图像文件.

上传图像

URL http://api.yeelink.net/v1.0/device/<device_id>/sensor/<sensor_id>/photos
数据格式 binary
Method POST
返回 HTTP Headers only
对该URL的一个HTTP POST请求会为指定的图像传感器上传一幅新的图像, 使用此API来为图像传感器存储图像数据, 目前只支持上传jpg, png, gif类型的图像.

图像信息是由key和value组成的键值对, 图像内容则是二进制图像文件.

图像型传感器 key为timestamp, value为图像信息(大小, 宽高, 类型);

Note: key为唯一索引; timestamp为ISO 8601标准时间格式(默认时区为中国标准时间CST), 例如: 2012-03-15T16:13:14.

图像上传成功后, 服务器会自动加上当前时间, 作为图像的key.

请求实例 (运用curl):
curl --request POST --data-binary @datafile.jpg --header "U-ApiKey: YOUR_API_KEY_HERE"
http://api.yeelink.net/v1.0/device/12/sensor/3/photos
需要在HTTP Header中增加API Key来授权写入操作, 具体请参照API Key. 支持一次传送一幅图像数据.

Note: 目前限定相邻图像数据上传间隔须大于等于10s, 过于频繁的请求会收到406 Response.

获取图像(信息)

URL http://api.yeelink.net/v1.0/device/<device_id>/sensor/<sensor_id>/photo/info/<key>
数据格式 JSON
Method GET
返回 图像信息
对该URL的请求返回指定key的图像信息, 若未指定key, 则返回该sensor的最新图像信息.

<key>为所要获取图像的时间戳(ISO 8601格式).
请求实例 (运用curl):
curl --request GET --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12/sensor/3/photo/info/2012-03-15T16:13:1

返回(JSON)

```
{
  "value":{"size": 45, "width": 240, "height": 320, "type": "jpg"}
}
```

其中size单位为kb.

若未指定key:
curl --request GET --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12/sensor/3/photo/info

返回(JSON)

```
{
  "timestamp":"2012-03-15T16:13:14",
  "value":{"size": 45, "width": 240, "height": 320, "type": "jpg"}
}
```

需要在HTTP Header中增加API Key来授权GET操作, 具体请参照API Key.

获取图像(内容)

URL http://api.yeelink.net/v1.0/device/<device_id>/sensor/<sensor_id>/photo/content/<key>

数据格式 binary
Method GET
返回 图像内容
对该URL的请求返回指定key的图像内容, 若未指定key, 则返回该sensor的最新图像内容.

```
<key>为所要获取图像的时间戳(ISO 8601格式).
请求实例 (运用curl):
curl --request GET --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12/sensor/3/photo/content/2012-03-15T16:1
```

返回(JSON)

```
<PhotoBinaryData>
```

若未指定key:
curl --request GET --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/12/sensor/3/photo/content

返回(JSON)

```
<PhotoBinaryData>
```

需要在HTTP Header中增加API Key来授权GET操作, 具体请参照API Key.

历史数据

URL http://api.yeelink.net/v1.0/device/<device_id>/sensor/<sensor_id>.json?start=<timestamp>&end=<timestamp>&interval=<interval>&page=<page>
数据格式 JSON
Method GET
返回 指定时间段的数据
对该URL的一个HTTP GET请求将返回指定时间段的数据.

start, end 规定了所请求数据的时间段, timestamp为ISO 8601格式时间(默认时区为CST),
例如: 2012-03-15T16:13:14.

page: 针对所查看的历史数据, 设定您所要查看的页码, 目前每页数据限制为200条. 默认值page=1.

interval: 规定了请求数据的间隔, 目前允许的数值如下表所示:

Interval	描述
1	每一秒取一个点
30	每30秒取一个点
60	每一分钟取一个点
300	每5分钟取一个点
900	每15分钟取一个点
1800	每30分钟取一个点
3600	每一小时取一个点
10800	每3小时取一个点
21600	每6小时取一个点
43200	每12小时取一个点
86400	每24小时取一个点

请求实例 (运用curl):
curl --request GET --header "U-ApiKey: YOUR_API_KEY_HERE" http://api.yeelink.net/v1.0/device/1/sensor/1.json?start=2012-06-02T14:01:46&end=2012-06-15T15:21:40&interval=1&page=1
需要在HTTP Header中增加API Key来授权GET操作, 具体请参照API Key.

返回(JSON)

数值型传感器 格式如下:

```
[
  {"timestamp": "2012-06-15T14:00:00", "value":315},
  {"timestamp": "2012-06-15T14:00:10", "value":316},
  {"timestamp": "2012-06-15T14:00:20", "value":317},
  {"timestamp": "2012-06-15T14:00:30", "value":317},
  {"timestamp": "2012-06-15T14:00:40", "value":317}
]
```

GPS型传感器 格式如下:

```
[
  {"timestamp": "2012-06-15T14:00:00", "value":{"lat":35.4,"lng":46.1,"speed":98.2}},
  {"timestamp": "2012-06-15T14:00:10", "value":{"lat":34.1,"lng":76.3,"speed":78.9}},
  {"timestamp": "2012-06-15T14:00:20", "value":{"lat":36.6,"lng":56.1,"speed":99.3}},
  {"timestamp": "2012-06-15T14:00:30", "value":{"lat":33.4,"lng":46.34,"speed":120}},
  {"timestamp": "2012-06-15T14:00:40", "value":{"lat":35.4,"lng":46.1,"speed":98.2}}
]
```

图像型传感器 格式如下:

```
[
  {"timestamp": "2012-03-15T16:13:14", "value":{"size":45,"width":240,"height":320,"type":"jpg"},
  {"timestamp": "2012-03-15T16:13:24", "value":{"size":180,"width":100,"height":320,"type":"png"},
  {"timestamp": "2012-03-15T16:13:34", "value":{"size":1024,"width":480,"height":360,"type":"gif"},
  {"timestamp": "2012-03-15T16:13:44", "value":{"size":2000,"width":240,"height":320,"type":"jpg"},
]
```

微博抓取器 不支持

模拟PUT, DELETE请求

若您的客户端不支持PUT, DELETE请求,您可在请求的URL中增加如下参数:

method=put (模拟PUT请求)
method=delete (模拟DELETE请求)

示例:

编辑传感器

```
URL: http://api.yeelink.net/v1.0/device/12/sensor/3?method=put
```

删除传感器

```
URL: http://api.yeelink.net/v1.0/device/12/sensor/3?method=delete
```

API Key

API Key用来授权对用户的设备, 传感器, 数据等的操作. 您可以在用户中心 -> 帐户 -> 我的帐户设置 中找到您自己的API Key.

对于需要授权访问的API, 您需要增加如下HTTP Header:

```
U-ApiKey: <your_api_key_here>
```

注册以获取 API Key

怎么,还在犹豫? 注册赶紧注册一个帐户吧.你可以使用获取的 API Key,将自己设计的硬件免费接入我们平台,以实现个性化的服务.