Yuktix REST API document

Author Rajeev Jha
Version 1.0
email <rjha@yuktix.com[1]>

1. Basics of writing a REST client

To write a client for Yuktix REST API, you will need

API endpoint

The server domain name (or IP), port, the application path and the method name is required to make a complete API endpoint. For example,

```
http://api1.yuktix.com:8080/sensordb/v1/echo
```

is the echo method of sensordb application version v1, deployed on machine with name api1.yuktix.com and port 8080.

HTTP Method

The http method (verb) to use can be GET, POST, PUT etc.

Content Type

The content-type for method is required. API require different content-type headers. (application/json, text/plain etc. A wrong content-type will result in HTTP 415.

HTTP headers

we need to populate the public API key inside Authorization header. You may need to set more headers depending on the API call.

Authorization header

Authorization header should be like

```
Authorization: Signature=3884191b-3951-4576-8544-f34a1e5615e9
```

where Signature field is your public API key available from your account.

[1] mailto:rjha@yuktix.com

HTTP response codes

The API will return an HTTP code on invocation. code **200** is for success. You will get a non-200 code on error. Please see a good HTTP status code document for HTTP error codes. We try to follow the HTTP codes as closely as possible.

- 200 - Processed Ok.
- 401 - Authorization error
- 400 - Bad arguments
- 500 - Internal server error

2. Sample curL session

```
* Hostname was NOT found in DNS cache
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 9090 (#0)
> POST /sensordb/v1/module/devices HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:22.0) Gecko/20130405
 Firefox/22.0
Host: localhost:9090
Accept: */*
Authorization: Signature=b18b5fffefa240417c62c03b6a273cd9
Content-Type: application/json; charset=UTF-8
Content-Length: 24

* upload completely sent off: 24 out of 24 bytes
< HTTP/1.1 200 OK
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Headers: origin, content-type, accept,
 authorization, x-requested-with
< Access-Control-Allow-Credentials: true
< Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS, HEAD
< Access-Control-Max-Age: 1728000
< Content-Type: application/json
< Date: Sun, 07 Jun 2015 07:54:01 GMT
< Content-Length: 4330
<
```

3. API Key

To use yuktix public api, you need an api key. To Register for API key

- open www.yuktix.com/app/login.php
- click on Sign in using Google+
- create an account using Google OAuth
- After sign in, click on My Account in Toolbar.
- Note down your API key

3.1. /module/devices API

method to access all devices in a public module.The public API key should be supplied as Authorization header.

endpoint	http://api1.yuktix.com:8080/sensordb/v1/module/devices
method	POST
Content Type	application/json

request params

```
{
    "map": {
        "name" : "AWS"
    }
}
```

```
POST /sensordb/v1/module/devices HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:22.0) Gecko/20130405
 Firefox/22.0
Host: api1.yuktix.com:8080
Accept: */*
Authorization: Signature=3884191b-3951-4576-8544-f34a1e5615e9
Content-Type: application/json; charset=UTF-8
```

3.2. /module/device API

method to get details of a public device. The public API key should be supplied as Authorization header.

endpoint	http://api1.yuktix.com:8080/sensordb/v1/module/device

method	POST
Content Type	application/json

request params

```
{
    "map":
    {
        "name" : "AWS",
        "serialNumber" : "test001"
    }
}
```

3.3. /module/device/archive API

method to get archive data of a public device. The public api key should be supplied as Authorization header. This api returns data from all the channels. To examine a single channel in detail, use the /module/tsdb api.

endpoint	http://api1.yuktix.com:8080/sensordb/v1/device/archive
method	POST
Content Type	application/json

request params

> This api will fetch the datapoints between start and end timestamps. start and end parameters are unix timestamps in millis (13 digits) The interval is in seconds (900 means 15 minutes). The results are sorted on unix_timestamp in ascedning order. The API will return maximum 1000 datapoints.

Rules to determine start and end timestamps

if end is not specified it defaults to now()
if start is not specified it defaults to end - interval
if interval is not specified it defaults to 86400

interval parameter is ignored for explicit start and end

```
e.g.
 {
    "map": {
        "module" : "AWS",
        "serialNumber" : "GSN002"
    }
}

will return all datapoints for end=now() and start=end()- 86400 seconds.


{
    "map": {
        "module" : "AWS",
        "serialNumber" : "GSN002",
        "interval" : "900"
    }
}

will return all datapoints for end=now() and start=end()-900 seconds.

{
    "map": {
        "module" : "AWS",
        "serialNumber" : "GSN002",
        "interval" : "900",
        "end" : "1435472869000"
    }
}

will return all datapoints for end=1435472869000 and start=end()-900
 seconds.


{
    "map": {
        "module" : "AWS",
        "serialNumber" : "GSN002",
        "start" : "1435472869000",
        "end" : "1435472869000"
    }
}
```

```
will return all datapoints between start and end timestamps
```

The API will return maximum 1000 datapoints. The responsibility of paginating over the range lies with the client. Suppose there are more than 1000 datapoints between start and end timestamps. clients should track max(unix_timestamp) returned from API and issue fresh call with next start and end timestamps.

```
RESPONSE

Array
(
    [code] => 200
    [response] => {
  "code" : 200,
  "result" : [ {
    "id" : "1179879",
    "__rssi" : "31",
    "T" : "22.47",
    "RH" : "52.27",
    "tsUnix" : "1452834865296",
    "P" : "90708.25",
    "Rain" : "0"
  }, {
    "id" : "1179868",
    "__rssi" : "31",
    "T" : "22.02",
    "RH" : "52.99",
    "tsUnix" : "1452834595435",
    "P" : "90708.16",
    "Rain" : "0"
  }...


response.code HTTP code 200 is success
response.result is an array of objects, each having properties
    - id
    - tsUnix :13 digit unix timestamp in millis
    Rest are channels of device. (Refer to list of all channels for
 channel code/name)
    e.g.
    T 22.47
```

```
    P 90708.25 etc.
```

3.4. /module/device/archive/latest API

/device/archive/latest method complements /device/archive and is used when we are only interested in latest points to have arrived in an interval. If the device has stopped or we have no idea about the datapoint distributin in time then we can use this method to find last N points. Also, it can be used to answer questions like, has anything arrived in last 15 minutes? The method returns points between start and end timestamps.

endpoint	http://api1.yuktix.com:8080/sensordb/v1/device/archive/latest
method	POST
Content Type	application/json

start and end calculations

interval is in seconds
end timestamp is now()
start timestamp is now() - interval
interval defaults to 900
if interval cannot be parsed: 900 is used.
limit is the number of rows to return. max. number of returned rows is 1000.

if interval is not supplied then API will return last <limit> points. (no matter when they arrived) The results are sorted on unix_timestamp in Descending order.

```
{
    "map" : {
        "module": "AWS",
        "serialNumber" : "test001",
        "limit" : "1"
    }
}

return the last datapoint (no matter when it arrived)

{
```

```
    "map" : {
        "module": "AWS" ,
        "serialNumber" : "test001",
        "interval" : "900"
    }
}

will return all datapoints to have arrived in last 15 mins. If no data
 point arrived in last 15 mins.
then we get an empty resultset.

RESPONSE

@see response of /device/archive API
```

3.5. /module/device/computation API

API to return computation results for a public device.

endpoint	http://api1.yuktix.com:8080/sensordb/v1/module/device/computation
method	POST
Content Type	application/json

request params

```
{"map" : {
    "channel": "RH",
    "frequency": "__DAILY__",
    "module": "AWS",
    "package" : "AWS",
    "serialNumber": "gkvk001",
    "start": "1420945200000",
    "end": "1421031600000"
    }
}

allowed frequencies are:

__DAILY__
__HOUR__
```

```
__MINUTE__
```

start time in unix_timestamp millis (13 digit)
end time in unix_timestamp millis (13 digit)
module is the public module for grouping together the devices
package is the computation package to generate reports on module devices

3.6. /module/tsdb API

This API is used to fetch timeseries data for a channel. (as against data for all channels of a device returned by /device/archive API)

endpoint	http://api1.yuktix.com:8080/sensordb/v1/module/tsdb
method	POST
Content Type	application/json

request params

The parameters to this API is a data structure with following attributes

- serialNumber
- channel
- start
- end
- duration
- function
- bucket
- limit
- fillZero
- multipoints

The parameter is explained with examples.

The serialNumber and channel attributes are mandatory. We return last 30 data points by default.

```
limit = 30
multipoints=0
fillZero=1

{
    "channel" : "RH",
    "serialNumber" : "GSN002"
}

will return last 30 time series points for this serialNumber and RH
 channel.
```

Duration is supplied in human time. The different units of human time are

- second
- minute
- hour
- day
- week
- month

```
{
    "channel" : "RH",
    "serialNumber" : "GSN002",
    "duration" : "7day"
}
```

You can supply duration parameter as "15second" or "1week" or "7day" etc. When duration is supplied, the start and end values are

end = now()
start = end - (duration in seconds)

Explicit start and end timestamps

start and end unix timestamps are in seconds (10 digit number) and duration is supplied in human time units. if duration is supplied then end is either the supplied value or it defaults to now()

```
start = end - duration
explicit start and end timestamps will override the duration.
{
    "channel" : "RH",
    "serialNumber" : "GSN002",
    "end" : "1435476594",
    "duration" : "7day"
}

This query will return data for GSN002 serial number device between
end = 1435476594
start = end - (7 days in seconds)
```

Role of limit

limit is used much like LIMIT in SQL queries. The limit parameter ensures that only LIMIT records are returned in the resultset

Aggregate queries

Aggregate queries (like average of results) are possible with parameters

- function
- bucket

Both are required for an aggregate query. function can be

- sum
- mean
- max
- min
- count

bucket param is for time grouping. The units are

- s
- m
- h

- d
- w

So to recive sum of rain for a device since midnight

```
channel=Rain
end = now()
start = unix timestamp in seconds  for midnight
function = sum
bucket = 1d (superflous for sum queries)
```

To find max temperature since 9am in morning

```
channel=T
end=now()
start= unix timestamp in seconds for 9am
function= max
bucket=1d
```

RESPONSE

```
Array
(
    [code] => 200
    [response] => {
        "id":"T",
        "at":"1452833771841",
        "current_value":"21.91",
        "multipoints":0,
        "datapoints":[{
            "value":"16.72",
            "at":"1452825850244",
            "sequence":"126880001"},
        ...]


response.code is HTTP code. 200 is success
response.response.datapoints is an array of data points
each datapoint has 3 properties
    - value
    - at (the unix timestamp in milli seconds, 13 digit long)
    - sequence (ignore)
```

3.7. /user/auth/google API

endpoint	http://api1.yuktix.com:8080/sensordb/v1/user/auth/google
method	POST
Content Type	application/json

request params

```
{
"id" : "1",
"given_name" : "rajeev",
"family_name" : "jha",
"email" : "jha.rajeev@gmail.com",
"link" : "http://www.google.com/1",
"picture": "http://www.google.com/pic/xyz.png"
}
```

RESPONSE

```
{
  "code" : 200,
  "result" : {
    "loginId" : "11",
    "sessionKey" : "683900c3-c7d4-4fc0-830e-595a617624f9",
    "firstName" : "rajeev",
    "lastName" : "jha",
    "handle" : "1",
    "email" : "jha.rajeev2@gmail.com",
    "module" : "__NULL__",
    "accountName" : "__google_1",
    "publicAPIKey" : "85cbd4e0-c50c-42ba-b256-776ad290a0fb"
  }
}
```

3.8. /module/snapshot API

endpoint	http://api1.yuktix.com:8080/sensordb/v1/module/snapshot
method	POST

Content Type	application/json

request params

```
{
    "module" : "AWS"
}
```

RESPONSE

```
Array
(
    [code] => 200
    [response] => {
  "code" : 200,
  "result" : [ {
    "id" : "41",
    "T" : "31.5",
    "RH" : "24.2",
    "address" : "HSR Layout, Bangalore",
    "tsUnix" : "1437578705883",
    "P" : "97577.0",
    "serialNumber" : "GSN002",
    "LWS" : "153.0",
    "longitude" : "76.6333",
    "latitude" : "12.283333",
    "Rain" : "4"
  }, {
    "id" : "42",
    "T" : "33.6",
    "RH" : "27.5",
    "address" : "Kanpur",
    "tsUnix" : "1437587915717",
    "P" : "99115.0",
    "serialNumber" : "GSN007",
    "longitude" : "80.3334",
    "latitude" : "26.4607",
    "Rain" : "3"
  } ]
}
)
```

```
response.code is HTTP code. 200 means success.
response.result is an array of json objects
Each object is guaranteed to have following properties
    - id
    - address
    - serialNumber
    - latitude
    - longitute
    - tsUnix

Rest of the properties are channel name and values.
E.g. in the example above
for device with id 42,
T is channel and value is 33.6
Rain is channel and value is 3
P is channel and value is 99115
```