Curb Platform API v2.0.1

[Jun 8, 2017]

This document is intended for use by approved third party application developers.

Authors

Neil Zumwalde ***neil@energycurb.com***

Cody Rushing ***cody@energycurb.com***

Introduction

Howdy, developer! We hope that you can use this to develop magical applications in home automation and energy savings. Maybe you're going to turn off your curling iron if you left it on in the bathroom, maybe you're going to use it to monitor the energy consumed by the robot clone of a major political figure you'll inevitably end up building in your basement. That's up to you! We don't ask questions, but we will totally vote for CongressBot 5000 if it promises to lower taxes.

If you have any questions on this API or to get a Client ID and Secret, don't hesitate to contact neil@energycurb.com.

The API root for third-party apps is https://app.energycurb.com/api.

Calling the API

All endpoints authorize using a jwt access token. Headers should be formatted to have an Authorization Header with the access token after the Bearer.

```
Authorization: Bearer [access_token]
```

All PUT, PATCH, and POST requests must have

```
Content-Type: application/json
```

All of our endpoints will return JSON formatted payloads.

Authorization

In order to obtain an access token, you must be issued a Client Id and Secret from Auth0, our third party authorization service. This will be granted to you by a Curb employee, as well as any whitelisted callback URLs that are necessary for authorization.

The Auth0 API Documentation can be consulted for the various OAuth flows that are supported. Depending on your application, you may want to implement a User/Password flow, Authorization Code, or any of the other Authorization Endpoints that Auth0 Supports.

https://auth0.com/docs/api/authentication

The Audience value in the Auth flows is `app.energycurb.com/api` and the connection is `Users`

User

GET /user

This endpoint returns metadata about the user.

Example:

```
GET https://app.energycurb.com/api/user
Authorization: Bearer [access_token]

response:
   {
     "email": "cody@energycurb.com",
     "name": "Cody",
     "email_verified": true,
     "user_id": "auth0|58af3e66c7b75a07750f61cc",
     "nickname": "cody",
     "updated_at": "2017-06-08T15:49:59.430Z",
     "created_at": "2017-02-23T19:56:22.820Z",
     "logins_count": 252
   }
```

Configuration

GET /locations

This endpoint returns all locations the user has access to. The location_ids will be used to get all the sample data for that location. It should be noted that users can have access to multiple locations, and locations can be accessed by multiple users. It is used as a way to group Curb Hubs so that they can work in concert to give a granular electrical profile.

Example:

```
GET https://app.energycurb.com/api/locations
Authorization: Bearer [access_token]

response:
   [
     {
       "geocode": "30.2399664,-97.7658567",
       "id": "b07bb3af-f69c-4bf2-8bb0-674469224fab",
       "extra_data": {
         "building_type": "house"
       },
       "address": "2611 S 5th St",
       "postcode": "78704",
```

```
      "country": "USA",
      "name": "Cody's house",
      "hasProduction": false
    },
    {
      "geocode": "30.24306,-97.73605500000001",
      "id": "3a57c9dc-e732-4f71-9aff-bb7128f7583b",
      "extra_data": {},
      "address": "1524 S IH 35",
      "postcode": "78704",
      "country": "USA",
      "name": "CURB Office",
      "hasProduction": true
    }
  ]
```


GET /locations/<location_id>/hubs

This endpoint returns all hub configurations within a location that the user has access to.

Example:


```
GET https://app.energycurb.com/api/locations/b07bb3af-
f69c-4bf2-8bb0-674469224fab/hubs
Authorization: Bearer [access_token]

response:
  [
    {
      "hardware_version": "unknown",
      "serial": "cryccgmy",
      "extra_data": {
        "display_name": "Cody~39_s hub"
      },
      "software_version": "2.0.0-qa2-dev",
      "os_version": "2.0.0-qa5",
      "last_post_diagnostics_data": 1496943112,
      "plc_connection": 0.986796,
      "model_number": "00613"
    }
  ]
```


Sample Data

Historical Sample Data

All Historical data uses the following get parameters:

- location_id : Location Id is a guid that the is stored in the user_metadata object of the user ID.
- time_range : The time range is a string that is a combination of a number and a unit of time from the following (e.g. 3h, 2w)

- h (hour)
- d (day)
- w (week)
- m (month)
- y (year)

- resolution : the resolution is one of 5 base sample time resolutions
  - s (seconds)
  - m (minute)
  - 5m (five minute)
  - h (hour)
  - d (day) [it should be noted that the day resolution is not localized (UTC-only)]

GET /historical/<location_id>/<time_range>/<resolution>

The historical endpoint returns the configured circuits for a given location, for the most recent time_range and will return averaged sample wattage for the resolution.

Example:

```
      GET https://app.energycurb.com/api/historical/b07bb3af-
f69c-4bf2-8bb0-674469224fab/1d/5m
      Authorization: Bearer [access_token]

      response:
         [
           {
             "label": "",
             "main": true,
             "production": false,
             "values": [
               {
                 "t": 1490627100,
                 "w": 56
               },
               {
                 "t": 1490627400,
                 "w": 54
               }
             ]
           },
           {
             "label": "Refrigerator",
             "main": false,
             "production": false,
             "values": [
               {
                 "t": 1490627100,
                 "w": 0
               },
               {
                 "t": 1490627400,
                 "w": 24
```

```
            }
          ]
        },
        ...
      ]
```

GET /aggregate/<location_id>/<time_range>/<resolution>

The aggregate endpoint returns an array of the configured circuits' metrics about the historical data.

- sum : The sum of all the wattages that were returned from that resolution
- avg : the average of all wattages on that resolution
- min / max : the minimum and maximum wattage for that resolution
- label : the circuit label
- main : true if the circuit is a main
- production : true if the circuit is a producer of electricity

Example:

```
GET https://app.energycurb.com/api/aggregate/b07bb3af-
f69c-4bf2-8bb0-674469224fab/1d/5m
Authorization: Bearer [access_token]

response:
   [
     {
       "sum": 293,
       "avg": 24,
       "min": 15,
       "max": 39,
       "kwhr": 0.26896813872192,
       "label": "Stove",
       "main": false,
       "production": false
     },
     {
       "sum": 293,
       "avg": 24,
       "min": 15,
       "max": 39,
       "kwhr": 0.26896813872192,
       "label": "Fan",
       "main": false,
       "production": false
     },
     ...
   ]
```

Real Time Sample Data

SOCKET.IO /circuit-data

1. Your server initiates a websocket request to https://app.energycurb.com/api/circuit-data
2. Once your client successfully connects, you will emit an authentication event with the user's id token - the same id token you use to access our other API endpoints.
3. Once our app responds saying you successfully authenticated, you can then subscribe to one or more locations.
4. Data will stream into your app.

Alternatively, if you're building a client-side application, you could do all the above except the user's browser will act as the socket.io client.

Example of a socket.io flow:

```
==============================
var io = require('socket.io-client');
var connectToLiveData = function(){
  // initialize connection
  var socket = io('https://app.energycurb.com/api/circuit-data');

  socket.on('connect', function(){
    // when the client is able to successfully connect, send an 'authenticate' event with the user's id token
    socket.emit('authenticate', {
      token: USER_ID_TOKEN
    });
  });
  socket.on('authorized', function(){
    // the client has been successfully authenticated, and can now subscribe to one or more locations
    socket.emit('subscribe', LOCATION_ID);
  });

  socket.on('data', function(data){
    // the client is receiving data for a specific location
    // data is a snapshot of the current state of all the circuits in a location
    // each circuit has a UUID, label, booleans that indicate whether they are mains circuits or production circuits, and a wattage value
    // data example:
    /*
      {
        locationId: <locationId>,
        circuits: [
          {
            id: <UUID>,
            label: '',
```

```
            main: true,
            production: false,
            w: 1200
          },
          {
            id: <UUID>,
            label: 'Refrigerator',
            main: false,
            production: false,
            w: 100
          },
          {
            id: <UUID>,
            label: 'Solar',
            main: false,
            production: true,
            w: -500
          }
          ...
        ]
      }
    */
  });
  // if the connection drops, try to reconnect
  socket.on('disconnect', connectToLiveData);
  return socket;
};
connectToLiveData();
==============================
```