Introduction

Tank Utility and SkyBitz have proposed a partnership in which SkyBitz can sell Tank Utility's equipment under their own brand. This document serves as a reference for the APIs necessary for configuring devices and exchanging readings data between systems.

These APIs explicitly omit customer-relevant data for the purposes of keeping SkyBitz and Tank Utility proprietary knowledge separate. Each party shall be responsible for maintaining their own customer data.

Authentication

Tank Utility will support an OAuth2.0 bearer token scheme. A refresh token will be traded for an access token. Access token will be valid for one hour.

Access tokens will be passed can be pass in the following ways:

- Header
    - 'Authorization: Bearer'
    - 'Authorization: OAuth
- URL parameter
    - 'access_token'
    - 'token'

Relevant Oauth endpoints

- auth endpoint: https://auth.tankutility.com/dialog/authorize
- token endpoint: https://auth.tankutility.com/oauth/token

Once the work has been started a client_id and client_secret will be shared.

For more information, please see: https://tools.ietf.org/html/rfc6750

Tank Utility Endpoints

GET /devices

Return Tank level information.

Return (JSON)

devices: Array of device IDs

Example

GET /devices

```
{
   "devices": [
      "00220027373234330347 3931",
      "00230016343438331547 3931",
      "00290022343438340247 3931",
      "002b002a3434383402473931",
      "002c00373434383402473931",
      "004a00133434383402473931"
   ]
}
```

GET /devices/{tank_id}

Return Tank level information. Tank_id can either be the long or the short ID.

Return (JSON)

device_id: string *(long device id)*
short_device_id: string *(short device id)*
status: string
capacity: number *(gallons)*
orientation: string
consumption_types: string
battery_warn: boolean
battery_critical: boolean
reading_interval: number *(seconds)*
transmission_interval: number *(seconds)*
threshold_1: number *(tank level in percent)*
threshold_2: number *(tank level in percent)*
lastReading: {
    tank: number *(tank %, from 0 - 100)*
    temperature: number
    time: number (*epoch*)
    time_iso: ISO date
}
telemetry: JSON (*diagnostic info, structure subject to change without notice*)

Example

GET /tank/00230016343438331547 3931


```
{
   "device": {
      "device_id": "00230016343438331547 3931",
      "short_device_id": "GAREVKMJ",
      "name": "MH - GAREVKMJ",
      "address": "Chicago, IL, USA",
      "account_id": "",
      "fuel_type": "propane",
      "status": "deployed",
      "capacity": 1000,
      "orientation": "horizontal",
      "consumption_types": "",
      "battery_warn": false,
      "battery_crit": false,
      "reading_interval": 21600,
      "transmission_interval": 86400,
      "threshold_1": 30,
      "threshold_2": -1,
      "lastReading": {
         "tank": 51.5972,
         "temperature": 76.585,
         "time": 1597114209344,
         "time_iso": "2020-08-11T02:50:09.344Z"
      },
      "telemetry": {...}

      }
}
```


PATCH /devices/{tank_id}

Update Tank configuration.

Body (JSON)

capacity: number **[1 - 65,535]**	(water capacity in gallons)
orientation: string **[horizontal/vertical]**	
name: string	
reading_interval: number **[3600 - 86400]**	(seconds; 21600 = default)
transmission_interval: number **[3600 - 86400]**	(seconds; 86400 = default)
fixed_time_transmission: number **[0 - 86400]**	*(seconds from midnight, UTC)*
threshold_1: number **[0-100]**	*(tank level in percent)*

threshold_2: number **[0-100]** *(tank level in percent)*

Example

```
PATCH /tank/101
{
        capacity: 500,
        orientation: 'vertical',
        name: 'Vacation House!',
        reading_interval: 21600,
        transmission_interval: 86400,
        fixed_time_transmission: 25200,
        threshold_1: 30,
        threshold_2: 15
}
```

GET /readings/{tank_id}

Return reading data for the last 7 days. Tank_id can either be the long or the short ID.

Return (Array of JSON)

tank: number *(tank %, from 0 - 100)*
temperature: number
time: number (*epoch*)
time_iso: ISO date,
telemetry: JSON (*diagnostic info, structure subject to change without notice*)

Example

GET /readings/00230016343483315473931


```
[
  {
    "tank": 51.674,
    "temperature": 81.095,
    "time": 1597630764000,
    "time_iso": "2020-08-17T02:19:24.000Z"
  },
  {
    "tank": 51.6548,
    "temperature": 77.534,
    "time": 1597652364000,
    "time_iso": "2020-08-17T08:19:24.000Z"
  },
  {
```

```
    "tank": 51.6548,
    "temperature": 76.11,
    "time": 1597673959000,
    "time_iso": "2020-08-17T14:19:19.000Z",
    "telemetry": [{..}]
  },
  ….
]
```


SkyBitz Endpoints

POST /tank/{tank_id}/reading

Create a new reading entry for the given Tank. When a new reading arrives at Tank Utility, we will pass the data to SkyBitz

Body (JSON)

time: timestamp *(UTC)*
percent: number
estimated_fill_date: timestamp *(UTC)*
event_code: string

Example

POST /tank/101/reading

{
time: '2020-07-01T20:46:02Z',
percent: 22.75,
estimated_fill_date: '2020-07-15T03:44:02Z',
event_code: 'thresh_1_trip'
}

POST /tank/{tank_id}

Update Tank configuration. When configuration information is changed in Tank Utility, we will pass the data to SkyBitz

Body

capacity: number

orientation: string
description: string
reading_interval: number (seconds)
transmission_interval: number (seconds)
fixed_time_transmission: number *(seconds from midnight)*
threshold_1: number *(tank level in percent)*
threshold_2: number *(tank level in percent)*

Example

POST /tank/101

```
{
    capacity: 500,
    orientation: 'vertical',
    description: 'Vacation House!',
    reading_interval: 21600,
    transmission_interval: 86400,
    fixed_time_transmission: 25200,
    threshold_1: 30,
    threshold_2: 15
}
```