OpenTHC API

Table of Contents

The OpenTHC API Specification is not an API for a specific system, rather it is an approach towards a **common** API for the cannabis industry. This document, and it's contents should be viewed as proposed guidelines.

Chapter 1. Overview

There are currently 100s of new software vendors in the cannabis technology space, some with APIs, some without. Each of these systems, as well as the government software provided by BioTrack, METRC, LeafData, etc, has a unique approach, with unique terminology to the same core data. This makes interoperability difficult, or at least tedious.

These proposals include a common data model, with provided JSON schema and samples as well as a REST (or JSON-RPC) style API. These models and interface are hopefully useful for others constructing tools in this space.

We want these standard base data models for objects in the Cannabis Industry to represent the common data all of our software shares with a common language and provide a basis for data increased interoperability.

With this foundation maybe we can all move a little faster.

Chapter 2. General

Unless stated otherwise, these things generally hold true:

2.1. Dates and Times

Date and Time values should be expressed using RFC 3339 formats which are an extension of ISO-8601

2.2. Units of Measure

Weights and Volume values should be expressed using the International System of Units (https://en.wikipedia.org/wiki/International_System_of_Units)

The system store all values internally in grams or liters, accurate to four decimal places. That is, accurate to 0.1 milligram/milliliter; expressed internally as grams or liters.

Weights can be input in any of the following values:

- Grams
- Milligrams
- Kilograms
- US Pounds
- US Ounces

Volume can be input in any of the following values

- Liters
- Milliliters
- US Ounces

Chapter 3. Authentication

Authentication to OpenTHC can occur through different methods with a preference for oAuth2 When OpenTHC is connecting through to a back-end system some of those parameters may need to be passed as well.

3.1. Open Connecxion

Authentication occurs through a request that looks something like this:

```
# With shell, you can just pass the correct header with each request
curl "$BASE/auth/open" \
    --data "rce=wa" \
    --data "license=ABC" \
    --data "client-key=ZYX"
```

OpenTHC will will respond to both set a cookie your client libraries can use to retain the session. Or, this token can be included in a request header as a **Bearer** token.

3.1.1. Variations

No all API compatible systems will use the same authentication methods. For example

- the OpenTHC Core system uses credential pairs and HMACs;
- the OpenTHC Pipe service uses mapped-pass-through credentials dependent on the backend.
- the OpenTHC P2P system uses pre-shared keys for authentication and signing

Refer to the implementation specific documentation for authentication methods

3.2. Ping

The `/auth/ping` endpoint provides a method for a client to check the status of their connexion. It should respond with some type of JSON, which may be dependent on which upstream system is in play.

```
curl "$BASE/auth/ping" \
    --header "Authorization: Bearer $HASH"

{
    "status": "success"
    "detail": "All Systems are Go"
}
```

3.3. Shut

Any request to `/auth/shut` with a session or access token will terminate/revoke this session or token. This request should always respond with an HTTP Staus of 206 for success or an appropriate HTTP Status on error.

```
curl "$BASE/auth/shut" \
    --header "Authorization: Bearer $HASH"
```

Chapter 4. Base Data

4.1. Company

A Company is a container for one or more Contact objects. A Company will have one or more Contacts and one or more Licenses. A **Company** is a container object, which will contain one or more **License** objects, and one or more **Contact** objects.

[diagram company] | *diagram-company.png*

4.2. License

A **License** is a container for the tracked materials, the license will have a LicenseType and is generally tied to a specific physical address. === Contact

A **Contact** is a human, as a member of a Company. A Contact may be a User or and Employee or simply a record for a visitor to a Company/License location. A Contact may authenticate to the system.

4.2.1. Contact User Accounts

Each user is identified by their email address, which must be unique across the system. A standard contact object contains name, phone and email address.

4.2.2. User Account Security

User access to the system is logged. Group ownership is checked on all object access. Each user has an access control list expressed for them.

Chapter 5. Crops & Plants

Plants are growing items in the System

5.1. Plants :: Create

Creating new Plants from either Clones or Seeds or other allowed Inventory Lot types.

5.1.1. Parameters

- Source: the Source Inventory Identifier
- Strain: Strain Name
- Batch: Some systems operate with a Batch concept, which could be provided here
- Stage: Descriptive text of the Stage of the plant
- Planted: Date Planted, ISO Date format
- Zone: The Room/Zone the Plant is located in

```
curl -X POST $API_BASE/plant

{
    license: {
        guid: "ABC123"
    },
    source: {
        guid: "I1A",
    },
    strain: {
        guid: "S2B"
    },
}

{
    guid: "P3C"
    strain: {
        name: "Alpha"
        guid: "S2B"
    }
}
```

5.2. Plants :: Modify

Change either the Batch, Strain, Stage, Zone, Planting Date, Mother Designation or other regulatory system defined attributes. When present, attributes will follow the OpenTHC JSON Schema. An implementation is free to extend these attributes.

```
curl -X POST $API_BASE/plant/$GUID

{
    "strain": { "guid": "S3C" }
    "zone": { "guid": "Z4D" }
}
```

5.3. Plants :: Delete

Marking a Plant as Deleted is the method to mark or confirm destruction of plant material.

If the compliance engine requires confirmation then a DELETE method is sent once to mark as scheduled for removal, and a second DELETE request to confirm.

```
curl -X DELETE $API_BASE/plant/$GUID

{
    "status": "success"
    "detail": "Requires Confirmation"
}
```

And then send the second request

```
curl -X DELETE $API_BASE/plant/$GUID

{
    "status": "success"
}
```

5.4. Growth

Add documentation about feed, fertilizer, nutrients, pesticides and other things added to or on the plants.

5.5. Plants :: Grow

Additives record the application of some material to the plants, including pesticies and nutrients.

5.5.1. Apply Grow Materials

5.5.2. Attach Grow Notes to Multiple Plants

5.6. Plants :: Notes

Notes on the plants record the application of nutrieinets, pesticides and other matter. Farmers may also use the Note field to attach comments or photos to the records.

5.6.1. Create a Plant Note

5.6.2. Attach Note to multiple Plants

5.7. Collecting Materials

A Wet Collection, sometimes called a Harvest or Manicure, is the process of taking raw materials from the crop.

A Dry Collection, sometimes called a Cure, is the process of collecting refined Wet Materials from Plants

5.8. Plants :: Wet Collect

Wet materials collection is also known as *Harvesting* or *Manicuring*. Generally the phrase harvesting means collection from the entire plant, while manicuring implies that collections will be made a bit at a time.

From Plants one or more **Wet Collections** can be made. A Wet collection is to enter materials that have been directly collected from the plants.

A **Plant Batch** is created to bundle these

Then Plants are added to this **Plant Batch**, including the weight in grams.

Once the Harvest is complete, as determined by farmer, this harvest bundle is closed

5.9. Plants :: Dry Collect

Dry materials collection is also known as *Curing*.

These dry materials are pulled from plant batches and classified by the farmer.

The reply includes the ID of the inventory items that were created from this process and includes and advisory value of the new weight of the affected batch.

This call can be repeated for each kind of material collected.

Chapter 6. Inventory

Lots, sometimes called Inventory, represent all Source, Product, Processing and Retail materials

[diagram lot] | *diagram-lot.png*

6.1. Listing Inventory

```
curl $API_BASE/inventory
```

6.2. Inventory / Adjust

A regulatory system specific type of adjustment to the inventory, generally requires a note.

```
curl -X POST $API_BASE/inventory/{OID}/adjust

{
    quantity: 55,
    code: 'audit',
    note: 'mis-count in processing'
}
```

6.3. Inventory / Modify

Only Permitted Modifications will be Allowed For Modification of Weight or Volume requires docuemntation

6.3.1. Inventory / Move

For Moving Inventory to a New Zone (aka Area or Room)

6.4. Inventory / Convert

The process of taking one or more Source lots and converting into one, or more, Output lots.

```
curl -X POST $API_BASE/inventory/convert --data-binary <-
{
    source: [
        {
            "guid": $GUID_A,
            "remove": 900,
        },
        {
            "guid": $GUID_B
            "remove": 100,
        }
    ],
    output: {
        product: {
            guid: $PRODUCT_GUID
        }
        qty: 1000
    }
}
```

This will record the removal from each of the indicated source items and record the linkage to the single output item.

6.5. Inventory / Split

Slice off a portion of an existing inventory, also known as Sub-Lotting.

```
curl -X POST $API_BASE/inventory/{$GUID}/split --data-binary <-
{
    qty: 1000
}
```

Chapter 7. Quality Assurance

Take a Sample from a product, provide Sample Results

7.1. Sample :: Create

From an Inventory Lot create a QA Sample Lot, which is a special type of sub-lot from the primary Inventory Lot. This Sample item will have a unique identifier and a child relationship to the source.

```
curl -X POST $API_BASE/lot/{OID}/sample

{
    type: "QA",
    quantity: "5"
}
```

7.2. Sample :: Detail

Return the details of the QA Sample, including which tests are required/requested. Similar to **requiredlabtestbatches** API call in METRC.

```
curl $API_BASE/qa/{OID}

{
}
```

7.3. Sample :: Destroy

A Sample is Destroyed by the Laboratory when they have finished sampling the materials. Or, in the case where a supplier no longer wants the test, the material should be destroyed. If the material is being returned to the supplier, one should use Void

```
curl $API_BASE/qa/{OID}

{
}
```

7.4. Sample :: Void

If the sample is no longer valid and the material is being returned to the supplier, use Void.

7.5. Result :: Create

Generally the Labratory (or sometimes the Licensed Operator) will update the QA results in the system. Either through the WebUI or via API.

```
curl -X POST $API_BASE/qa/{OID}

{
    "status": "failed",
    "metric": {
        "general": { ... },
        "potency": { ... },
        "microbe": { ... },
    }
}
```

7.6. Result :: Void

Generally the Labratory (or sometimes the Licensed Operator) will update the QA results in the system. Either through the WebUI or via API.

```
curl -X DELETE $API_BASE/qa/{OID}

{
    "status": "warning",
    "detail": "Call Delete again to confirm"
}
```

Chapter 8. Transfers

Chapter 9. Transfer

A Transfer manifest is prepared to indicate the Transfer of materials from one license holder to another.

9.1. Outgoing

Export is the process of selecting one or more Inventory items and preparing them for delivery. Items and Quantities (or Sub-Lots) are placed on a Transfer Manifest. And this Manifest is filed with the regulatory system.

9.1.1. Outgoing / Create

9.1.2. Outgoing / Commit

9.1.3. Outgoing / Update

9.2. Incoming

Incoming Transfers is the process of receiving a request, processing the materials into the target License inventory.

9.2.1. Incoming / Accept

Chapter 10. Retail Sales

Chapter 11. Retail Sales

A transaction selling one or more items to a retail customer. This customer may or may not be tied to a specific Contact (or a Generic Contact such as "walk-in")

11.1. Retail Sales Item

Each line-item, and it’s tied back to the Retail Sale as well as the specific Inventory Lot.

[diagram retail sale] | *diagram-retail-sale.png*

Chapter 12. Adminstration

12.1. Configuration Accounts

The OpenTHC API also supports interactions to keep track of the companies, licensees and locations.

12.2. Company

A Company is a container for one or more Contact objects. A Company will have one or more Contacts and one or more Licenses. A **Company** is a container object, which will contain one or more **License** objects, and one or more **Contact** objects.

[diagram company] | *diagram-company.png*

12.3. Contact

A **Contact** is a human, as a member of a Company. A Contact may be a User or and Employee or simply a record for a visitor to a Company/License location. A Contact may authenticate to the system.

12.3.1. Contact User Accounts

Each user is identified by their email address, which must be unique across the system. A standard contact object contains name, phone and email address.

12.3.2. User Account Security

User access to the system is logged. Group ownership is checked on all object access. Each user has an access control list expressed for them.

12.4. Permissions

12.5. Grow Materials

Inputs for grow supplies; adding a bulk item, with cost and the removing portions.

Inputs for grow journals; adding a note and a metric to a plant (or group of plants, but tracked per-plant)

Chapter 13. Customization / Extending

If it's a really good idea please consider a pull request.

Additionally, the JSON can be extended, without affecting the base. The addition of an **x-[vendor]** attribute to a JSON model should suffice. This is shown in some of the examples.