REST API v3 Usage Examples

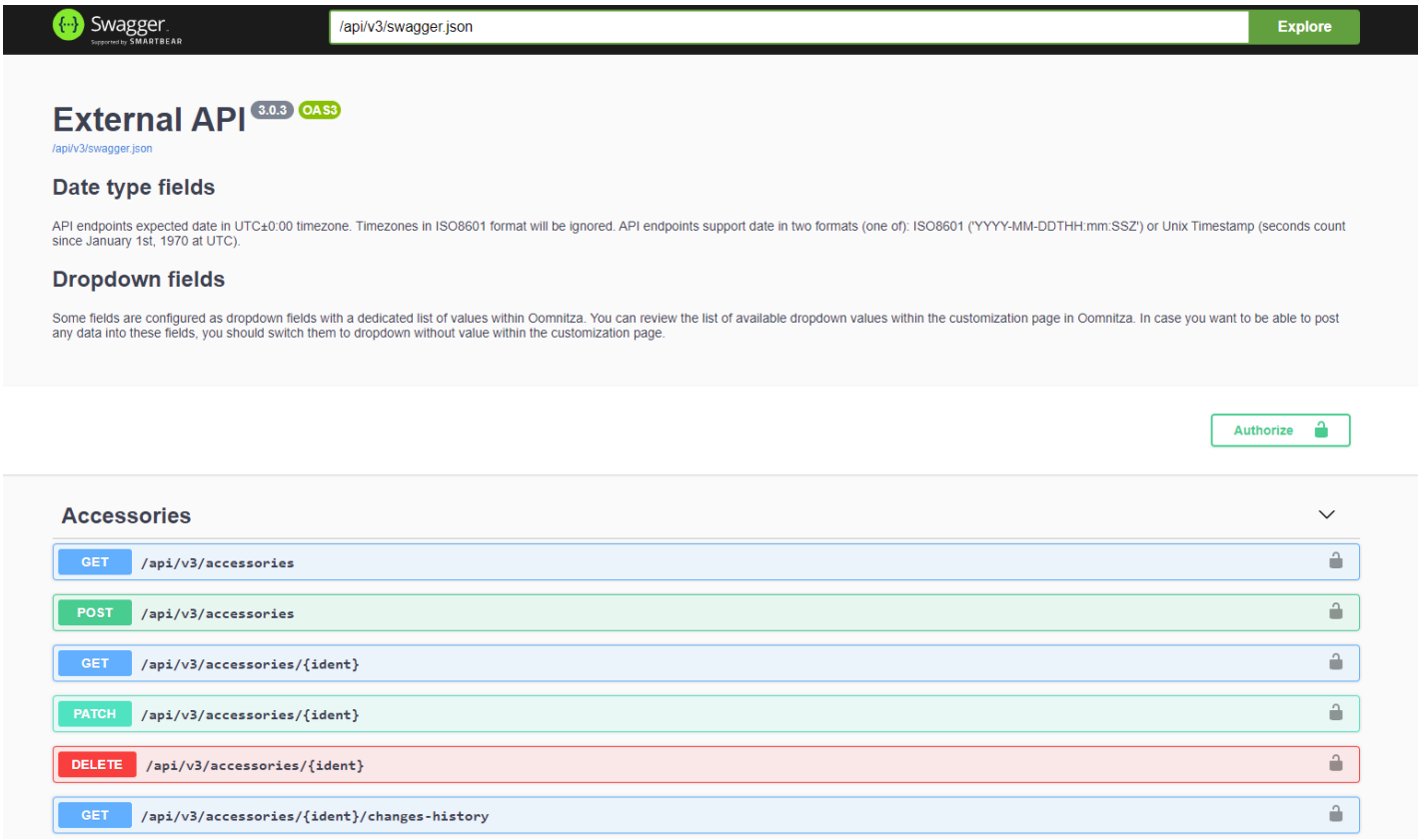
Table Of Contents

Introduction

The Oomnitza API v3 is a RESTful interface that enables interaction with resources in Oomnitza. This interface follows REST best practices, to help ensure that creating and updating objects in Oomnitza is as intuitive as possible. The purpose of this document is to provide a number of common use cases involving the API. We are constantly enhancing the platform, and the list of available resources extends beyond the scope of this document. For any questions regarding the API, please contact support@oomnitza.com.

The API supports both HTTP basic and token-based authentication, as well as session-based authentication. All data in transit is securely encrypted via SSL/TLS. The request and response Content-Type values are "`application/json`".

Additional documentation for the API v3 can be accessed within the Oomnitza WebUI. By navigating to *Configuration -> Integrations -> Rest APIs (Swagger docs)*, you will be presented with an interactive interface, with additional endpoints detailed.



Screenshot of the API documentation within the WebUI, powered by swagger.

Fetching Asset Metadata Fields

HTTPS GET
URL: /api/v3/assets/meta
HEADERS: Authorization2: {API_TOKEN_HERE}

EXAMPLE:
> REQUEST:

```
curl --request GET \
     --header 'Authorization2: da9adf3a45504b9593542d4a24d1ed5f' \
     --header "Content-Type: application/json" \
  https://feature1.oomnitza.com/api/v3/assets/meta ; echo
```

> RESPONSE:


```
{
   "machine_age_date":  {
      "default_value":"",
      "unique":"0",
      "length":11,
      "dependencies":{},
      "data_type":"DATE",
      "INTERNAL_ID":"A65A2F70849011E4ABBC066B556BA4EE",
      "searchhelp_type":null,
      "optional":"1",
      "searchhelp":[null]
   }, ...
}
```


Meta Object Detail

Key	Description
default_value	If creating a new object and this property is omitted, the *default_value* will be used.
unique	0 - Not unique; 1 - Enforce value uniqueness.
length	Maximum length of the value.
dependencies	Dictionary where custom field ids are keys, and the contents are values that must match the field value. If all match - present/display this field to the user; If one or more do not match - do not display field to user.
data_type	The data type of the field. *CHAR* | *INT* | *DOUBLE* | *LOCATION* | *BARCODE* | *USERS*
INTERNAL_ID	The unique internal id generated at time of field creation. This is the API v2 field key.
searchhelp_type	Special flags for input method. Example: *list*
optional	0 - Field is mandatory in order to create object; 1 - Field is optional (and can be left blank)
searchhelp	If *searchhelp_type* flag is set, *searchhelp* will contain the list/autocomplete settings.

Creating a New Asset Record

The API has a validation process to ensure all fields included in the payload are defined in Oomnitza. If the payload includes unknown fields, by the default the API will reject the request.

In some cases, however, the set of fields in the payload cannot be predefined to correspond to the Oomnitza instance. To ensure that the target Oomnitza instance does not reject these payloads, you can set the header: **`Oomnitza-Allow-Unknown-Meta-Fields`** equals 1. With this set, the API will ignore the unknown fields and process those that are recognized by Oomnitza.

HTTPS POST
URL: /api/v3/assets
HEADERS: Authorization2: {API_TOKEN_HERE}
BODY: { "**barcode**":"0123456789", "**status**":"Shelf Inventory", "**model**":"Nexus 7" }

EXAMPLE:
> REQUEST:

```
curl --request POST \
  --user 'api:admin123' \
  --header "Content-Type: application/json" \
  --data '{ "barcode":"0123456789",  "status":"Shelf Inventory",  "model":"Nexus 7" }' \
  https://feature1.oomnitza.com/api/v3/assets ; echo
```

> RESPONSE:
{*"**equipment_id**":"ac720fadd672405ea5141975c97800d9"*}

Viewing a Specific Asset

HTTPS GET
URL: /api/v3/assets/{**id**}
HEADERS: Authorization2: {API_TOKEN_HERE}

EXAMPLE:
> REQUEST:

```
curl --request GET \
    --user 'api:admin123' \
    --header "Content-Type: application/json" \
  https://feature1.oomnitza.com/api/v3/assets/ac720fadd672405ea5141975c97800d9 ; echo
```

> RESPONSE:


```
{
  "status": "Active",
  "total_memory_mb": "4096",
  "uid": 919,
  "warranty_end_date": "1453806523",
  "equipment_id": "ac720fadd672405ea5141975c97800d9",
  "creation_date": "1443807903",
  "last_check-in_date": "1444807542",
  "notes": "Refurbished",
  …
}
```


List Existing Asset Records

HTTPS GET
URL: /api/v3/assets
HEADERS: Authorization2: {API_TOKEN_HERE}

EXAMPLE:
> REQUEST:

```
curl --request GET \
    --user 'api:admin123' \
    --header "Content-Type: application/json" \
 https://feature1.oomnitza.com/api/v3/assets ; echo
```

> RESPONSE:


```
[ {"barcode": "012345", "equipment_id": "ddbe4bf0d8f411e482d506d4881d655b", … }, … ]
```


The response will include the header: **`X-Total-Count`**. This will be the total number of assets given the current filter. By default, 100 records will be returned. When more than 200 records are in the results, multiple calls will be needed to get all the records. This can be done using the `skip` url param. So, the call to get records 200-400 would use the following url: `/api/v3/assets/?skip=200`.

List Existing Asset Records (With a Limited Field Set).

If you are interested in a particular set of fields, you can include in the ***fields*** query-string parameter.

HTTPS GET
URL: /api/v3/assets/?fields=*{comma_delmited_list_of_fields}*

HEADERS: Authorization2: {API_TOKEN_HERE}

EXAMPLE:
> REQUEST:

```
curl --request GET \
    --user 'api:admin123' \
    --header "Content-Type: application/json" \
  https://feature1.oomnitza.com/api/v3/assets/?fields=barcode ; echo
```

> RESPONSE:

[{"**barcode**": "012345"}, {"**barcode**": "012346"}, {"**barcode**": "012347"}, {"**barcode**": "012348"}, …]

List Existing Asset Records (With Pagination).

"skip" and "limit" are used for pagination. "skip" represents the starting point which starts with index 0. "limit" means the amount of each page and the default is 200. If you specify "skip=5" and "limit=10", the request will return assets 5 through 14.

HTTPS GET
URL: /api/v3/assets/?skip={start_index}&limit={amount_each_page}
HEADERS: Authorization2: {API_TOKEN_HERE}

EXAMPLE:
> REQUEST:

```
curl --request GET \
    --user 'api:admin123' \
    --header "Content-Type: application/json" \
  https://feature1.oomnitza.com/api/v3/assets/?skip=0&limit=100 ; echo
```

> RESPONSE:


```
[
 {
   "barcode": "0123456789"
   "status": "Shelf Inventory",
   "total_memory": "4096",
   "equipment_id": "ac720fadd672405ea5141975c97800d9"
   …
  },
  ...
]
```

List Existing Asset Records (With Sorting).

This request will return the list of assets sorted by the specified field in ascending or descending order. If there is no "asc" / "desc" specified, you will see the list in ascending order as default.

HTTPS GET
URL: /api/v3/assets/?sortby={field external id} {asc | desc}
HEADERS: Authorization2: {API_TOKEN_HERE}

EXAMPLE:
> REQUEST:

```
curl --request GET \
    --user 'api:admin123' \
    --header "Content-Type: application/json" \
  https://feature1.oomnitza.com/api/v3/assets/?sortby=change_date desc ; echo
```

> RESPONSE:

```
[
 {
  "barcode": "0123456789"
  "status": "Shelf Inventory",
  "total_memory": "4096",
  "equipment_id": "ac720fadd672405ea5141975c97800d9"
  "change_date": "1555103498"
  …
 },
 {
  "barcode": "9876543210"
  "status": "Assigned",
  "total_memory": "2048",
  "equipment_id": "c36bd0e0bccc4b56b85fd66e1626f803"
  "change_date": "1553029233"
  …
 },
 ...
]
```

List Archived Asset Records

The request with "include_deleted=1" will return all assets including archived ones. Then, you need to add "filter=deleted isnt null" to filter out alive assets.

HTTPS GET
URL: /api/v3/assets/?include_deleted=1&filter=deleted isnt null
HEADERS: Authorization2: {API_TOKEN_HERE}

EXAMPLE:
> REQUEST:

```
curl --request GET \
     --user 'api:admin123' \
     --header "Content-Type: application/json" \
  https://feature1.oomnitza.com/api/v3/assets/?include_deleted=1&filter=deleted \  isnt null ;
echo
```

> RESPONSE:

```
[
 {
   "barcode": "0123456789"
   "status": "Shelf Inventory",
   "total_memory": "4096",
   "equipment_id": "ac720fadd672405ea5141975c97800d9"
   "deleted": "Y"
   …
  },
  ...
]
```

Searching For Existing Asset Records

HTTPS GET
URL: /api/v3/assets?filter={***field_id comparison value***}
HEADERS: Authorization2: {API_TOKEN_HERE}

EXAMPLE:
> REQUEST:

```
curl --request GET \
     --user 'api:admin123' \
     --header "Content-Type: application/json" \
  "https://feature1.oomnitza.com/api/v3/assets?filter=barcode eq '0123456789'" ; echo
```

> RESPONSE:

```
[
 {
   "barcode": "0123456789"
   "status": "Shelf Inventory",
   "total_memory": "4096",
   "equipment_id": "ac720fadd672405ea5141975c97800d9"
   …
  }
]
```

Editing an Asset Record

HTTPS PATCH
URL: /api/v3/assets/{**id**}
HEADERS: Authorization2: {API_TOKEN_HERE}
BODY: { "**barcode**":"0123456789", "**status**":"Loaned Out", "**assigned_to**":"trent.seed" }

EXAMPLE:
> REQUEST:

```
curl --request PATCH \
    --user 'api:admin123' \
    --header "Content-Type: application/json" \
    --data '{ "status":"Loaned Out", "assigned_to":"trent.seed" }' \
  https://feature1.oomnitza.com/api/v3/assets/ac720fadd672405ea5141975c97800d9 ; echo
```

> RESPONSE:

```
{
  "status": "Loaned Out",
  "assigned_to": "trent.seed",
  "version": "2",
  "equipment_id": "ac720fadd672405ea5141975c97800d9",
  …
}
```

Viewing Asset Record History

HTTPS GET
URL: /api/v3/assets/{**id**}/history
HEADERS: Authorization2: {API_TOKEN_HERE}

EXAMPLE:
> REQUEST:

```
curl --request GET \
    --user 'api:admin123' \
    --header "Content-Type: application/json" \
  https://feature1.oomnitza.com/api/v3/assets/ac720fadd672405ea5141975c97800d9/history ;
echo
```

> RESPONSE:

[{"**version**": "0", … }, {"**version**": "1", … }, {"**version**": "2", … }, {"**version**": "3", … }, …]

Deleting an Asset Record

HTTPS DELETE
URL: /api/v3/assets/{**id**}
HEADERS: Authorization2: {API_TOKEN_HERE}

EXAMPLE:
> REQUEST:

```
curl --request DELETE \
    --user 'api:admin123' \
    --header "Content-Type: application/json" \
  https://feature1.oomnitza.com/api/v3/assets/ac720fadd672405ea5141975c97800d9 ; echo
```

> RESPONSE:

{ "equipment_id": "string", "version": "string", "creation_date": 0, "change_date": 0, "created_by": "string", "changed_by": "string", "location": "string", "stockroom": "string….}

Viewing Asset Child Records

HTTPS GET
URL: /api/v3/assets/{**id**}/children
HEADERS: Authorization2: {API_TOKEN_HERE}

EXAMPLE:
> REQUEST:

```
curl --request GET \
    --user 'api:admin123' \
    --header "Content-Type: application/json" \
  https://feature1.oomnitza.com/api/v3/assets/ac720fadd672405ea5141975c97800d9/children ;
echo
```

> RESPONSE:

```
[
  {
    "barcode": "01234",
    "serial_number": "C0232424",
    "asset_type": "HDD",
    ...
  },
  ...
]
```

Fetching User Metadata Fields

This request will allow you to see the schema of the “users” object and identify field ids that you would like to use for creating or modifying a user record.

HTTPS GET
URL: /api/v3/users/meta
HEADERS: Authorization2: {API_TOKEN_HERE}

EXAMPLE:
> REQUEST:

```
curl --request GET \
     --header 'Authorization2: da9adf3a45504b9593542d4a24d1ed5f' \
     --header "Content-Type: application/json" \
  https://feature1.oomnitza.com/api/v3/users/meta ; echo
```

> RESPONSE:


```
{
  "username": {
        "default_value": "",
        "uid": 34,
        "INTERNAL_ID": "USER",
        "hardcode": "1",
        "display_order": 20,
        "editable": "1",
        "length": 128,
        "label": "Username",
        "data_type": "CHAR",
        "belongs_to": "USERS",
        "dependencies": {},
        "flags": {
            "text_search_enabled": "True"
        },
        "searchhelp_type": null,
        "unique": "1",
        "optional": "0",
        "searchhelp": []
    },
   ...
}
```

Creating a User Record

HTTPS POST
URL: /api/v3/users
HEADERS: Authorization2: {API_TOKEN_HERE}
BODY: {"username": "john.smith", "first_name": "John", "last_name": "Smith", "role": 1, "email": "john.smith@gmaill.com"}

EXAMPLE:
> REQUEST:

```
curl --request POST \
    --user 'api:admin123' \
    --header "Content-Type: application/json" \
    --data '{"username": "john.smith", "first_name": "John", "last_name": "Smith", "role": 1, "email": "john.smith@gmaill.com"}' \
https://feature1.oomnitza.com/api/v3/users ; echo
```

> RESPONSE:
{"**username**": "john.smith"}

Optional: Set "send_welcome_email" to "Yes" in the URL to send a welcome email to the user, /api/v3/users? send_welcome_mail=Yes

Modifying a User Record

This request will allow you to modify existing user records, including status changes which can kick-off workflows in Oomnitza.

HTTPS PATCH
URL: /api/v3/users/{username}
HEADERS: Authorization2: {API_TOKEN_HERE}
BODY: {"status": "Inactive"}

EXAMPLE:
> REQUEST:

```
curl --request PATCH \
    --user 'api:admin123' \
    --header "Content-Type: application/json" \
    --data '{"status": "Inactive"}' \
```

https://feature1.oomnitza.com/api/v3/users/john.smith ; echo

> RESPONSE:
{“status”: “Inactive”, “first_name”: “John”, “last_name”: “Smith”, ...}

Fetching a User Record

If you would like to access the detail view for a particular user id or username, the following request can be utilized.

HTTPS GET
URL: /api/v3/users/{username}
HEADERS: Authorization2: {API_TOKEN_HERE}

EXAMPLE:
> REQUEST:

```
curl --request GET \
    --user 'api:admin123' \
    --header "Content-Type: application/json" \
https://feature1.oomnitza.com/api/v3/users/john.smith ; echo
```

> RESPONSE:
{“username”: “john.smith”, “first_name”: “John”, “last_name”: “Smith”, “version”: “1”, ...}

Deleting a User Record

HTTPS DELETE
URL: /api/v3/users/{username}
HEADERS: Authorization2: {API_TOKEN_HERE}

EXAMPLE:
> REQUEST:

```
curl --request DELETE \
    --user 'api:admin123' \
    --header "Content-Type: application/json" \
https://feature1.oomnitza.com/api/v3/users/john.smith ; echo
```

> RESPONSE:
204 OK

Searching for User Records

You may search for asset records by providing a *filter* argument. See the section titled “The filter Argument” for more details.

HTTPS GET
URL: /api/v3/users?filter={FILTER}
HEADERS: Authorization2: {API_TOKEN_HERE}

EXAMPLE:
> REQUEST:

```
curl --request GET \
    --user 'api:admin123' \
    --header "Content-Type: application/json" \
https://feature1.oomnitza.com/api/v3/users?filter=status eq 'Active' ; echo
```

> RESPONSE:

```
[
   {“username”: “user_one”, “status”: “Active”, “first_name”: “John”, “last_name”: “”, …},
   {“username”: “user_two”, “status”: “Active”, “first_name”: “Bob”, “last_name”: “”, …},
   {“username”: “user_three”, “status”: “Active”, “first_name”: “Katie”, “last_name”: “”, …},
   ...
]
```

The `filter` Argument

The `filter` request argument is used to apply a filter to list operations. The format is a string of “`[field] [op] [value] ci`” conditions joined with AND and OR. Valid operators are:

API Filter Operator	Description / Purpose
eq	field Equals value
neq	field Does Not Equal value
in	field Is In [list]
like	field LIKE value (SQL LIKE)
lt	field Is Less Than value
gt	field Is Greater Than value
lte	field Is Less Than or Equal To value

gte	field Is Greater Than or Equal To value
is NULL	field Is Empty
isnt NULL	field Is Not Empty
between [date] AND [date]	date field is between the two dates. Use the format mm dd yyyy
between [number] AND [number]	numeric field is between the two dates. Use the format mm dd yyyy

Examples:

barcode IS NULL	find records with an empty barcode field
barcode LIKE “A%” OR barcode LIKE “B%”	find records with a barcode starting with an A or B
barcode IN [“123”, “ABC”, “XYZ”]	find records with a matching barcode
date_field BETWEEN 01/01/2010 AND 01/01/2020	fields records with date_field between the two dates

Please contact support@oomnitza.com with any questions.