PSWebApp Documentation

Release 1.0

Malcolm Mackay

Aug 27, 2020

CONTENTS

See also cuplcodec and cuplfrontend.

CHAPTER

ONE

API SPECIFICATIONS

1.1 Admin API Specification

1.1.1 Tag

GET /tag
get a tag

Get a tag from an id.

Query Parameters

- **id** (*integer*) – Tag id

Example request:

```
GET /tag HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A tag object

  Example response:

  ```
  HTTP/1.1 200 OK
  Content-Type: application/json

  {
      "id": 1,
      "serial": "string",
      "secretKey": "string",
      "timeregistered": "2020-08-27T10:57:40.207501"
  }
  ```

- 400 Bad Request – bad input parameter

  Example response:

  ```
  HTTP/1.1 400 Bad Request
  Content-Type: application/json

  {}
  ```

DELETE /tag

delete a tag

delete a tag

Query Parameters

- **id** (*integer*) – Tag id

Status Codes

- 204 No Content – Tag has been deleted

  Example response:

```
HTTP/1.1 204 No Content
Content-Type: application/json

{}
```

- 400 Bad Request – bad input parameter

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 404 Not Found – No tag found

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

GET /tags

Get a list of tags ordered by ID.

Get a list of tags

Query Parameters

- **offset** (*integer*) – Return samples starting from this index.
- **limit** (*integer*) – Limit the number of samples returned.

Example request:

```
GET /tags HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – List of tags

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "id": 1,
        "serial": "string",
        "secretKey": "string",
        "timeregistered": "2020-08-27T10:57:40.207501"
    }
]
```

- 400 Bad Request – bad input parameter

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

POST /tags

create a tag

Create a tag

Example request:

```
POST /tags HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "id": 1
}
```

Status Codes

- 201 Created – Tag created

  Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "id": 1,
    "serial": "string",
    "secretKey": "string",
    "timeregistered": "2020-08-27T10:57:40.207501"
}
```

- 400 Bad Request – invalid input, object invalid

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
```

(continues on next page)

(continued from previous page)

```
{}
```

- 409 Conflict – a user with the same oauth_id already exists

  Example response:

```
HTTP/1.1 409 Conflict
Content-Type: application/json

{}
```

GET /tagviews
Get a list of TagViews.

Get a list of TagViews.

Query Parameters

- **distinctOnTag** (*boolean*) – Return only the latest TagView for each scanned tag.
- **tag_id** (*integer*) – Filter TagViews by tag_id.

Example request:

```
GET /tagviews HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A list of tagview objects ordered from newest to oldest.

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "id": 1,
        "tagserial": "string",
        "timestamp": "2020-08-27T10:57:40.207501"
    }
]
```

GET /tagviews/{id}
Get a tagview

Get a tagview

Parameters

- **id** (*integer*) – Tag view ID

Example request:

```
GET /tagviews/{id} HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A tagview object

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": 1,
    "tagserial": "string",
    "timestamp": "2020-08-27T10:57:40.207501"
}
```

- 404 Not Found – TagView not found.

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

DELETE /tagviews/{id}

Delete a tag view

Parameters

- **id** (*integer*) – Tag view ID

Status Codes

- 204 No Content – TagView deleted

  Example response:

```
HTTP/1.1 204 No Content
Content-Type: application/json

{}
```

- 400 Bad Request – Bad input.

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 404 Not Found – TagView not found

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

Simulate

GET /tag/{id}/simulate
Simulate URL.

Obtain a simulated URL from the tag. This will contain mock sensor data.

Parameters

- **id** (*integer*) – Tag id to simulate

Query Parameters

- **frontendurl** (*string*) – URL of the consumer frontend application that will decode wscodec URLs. (Required)
- **nsamples** (*integer*) – Number of temperature and humidity samples to include in the wscodec URL. 0 samples is valid and should raise an error.
- **smplintervalmins** (*integer*) – Time interval between samples in minutes.
- **format** (*integer*) – Format code e.g. 1 for temperature and humidity and 2 for temperarture samples only.
- **usehmac** (*boolean*) – Use HMAC rather than MD5.
- **batvoltagemv** (*integer*) – Battery voltage in mV.
- **bor** (*boolean*) – Reset caused by Brownout.
- **svsh** (*boolean*) – Reset caused by Supply Voltage Supervisor (High Side).
- **wdt** (*boolean*) – Reset caused by watchdog timer.
- **misc** (*boolean*) – Miscellaneous error flag on cupl Tag.
- **lpm5wu** (*boolean*) – Low Power Mode x.5 wakeup flag.
- **clockfail** (*boolean*) – Clock failure error flag.
- **tagerror** (*boolean*) – Initialise encoder with the error flag set. The circular buffer will be empty.

Example request:

```
GET /tag/{id}/simulate?frontendurl=string HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – URL containing sensor data.

  Example response:

  ```
  HTTP/1.1 200 OK
  Content-Type: application/json

  string
  ```

- 400 Bad Request – bad input parameter

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

1.1.2 Capture

GET /capture
get a capture.

Get a capture by its ID.

Query Parameters

- **id** (*integer*) – Capture id

Example request:

```
GET /capture HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A capture object

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": 1,
    "serial": "string",
    "statusb64": "string",
    "timeintb64": "string",
    "circbufb64": "string",
    "hmac": "string",
    "start-timestamp": "2020-08-27T10:57:40.207501",
    "end-timestamp": "2020-08-27T10:57:40.207501"
}
```

- 400 Bad Request – bad input parameter

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

DELETE /capture
delete a capture

delete a capture

Query Parameters

- **id** (*integer*) – Capture id

Status Codes

- 204 No Content – Capture has been deleted

  Example response:

```
HTTP/1.1 204 No Content
Content-Type: application/json

{}
```

- 400 Bad Request – bad input parameter

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 404 Not Found – No capture found

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

GET /captures

Get a list of captures ordered by ID.

Get a list of captures

Query Parameters

- **offset** (*integer*) – Return captures starting from this index.
- **limit** (*integer*) – Limit the number of captures returned.
- **tag_id** (*integer*) – Only returns captures from tag_id.

Example request:

```
GET /captures HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – List of captures

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "id": 1,
        "serial": "string",
        "statusb64": "string",
```

(continues on next page)

(continued from previous page)

```
        "timeintb64": "string",
        "circbufb64": "string",
        "hmac": "string",
        "start-timestamp": "2020-08-27T10:57:40.207501",
        "end-timestamp": "2020-08-27T10:57:40.207501"
    }
]
```

- 400 Bad Request – bad input parameter

Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

POST /captures

create a capture

Create a capture

Example request:

```
POST /captures HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "id": 1,
    "serial": "string",
    "statusb64": "string",
    "timeintb64": "string",
    "circbufb64": "string",
    "hmac": "string",
    "start-timestamp": "2020-08-27T10:57:40.207501",
    "end-timestamp": "2020-08-27T10:57:40.207501"
}
```

Status Codes

- 201 Created – Capture created

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "id": 1,
    "serial": "string",
    "statusb64": "string",
    "timeintb64": "string",
    "circbufb64": "string",
    "hmac": "string",
    "start-timestamp": "2020-08-27T10:57:40.207501",
    "end-timestamp": "2020-08-27T10:57:40.207501"
}
```

- 400 Bad Request – invalid input, object invalid

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 409 Conflict – a capture with the same id already exists

  Example response:

```
HTTP/1.1 409 Conflict
Content-Type: application/json

{}
```

GET /capture
get a capture.

Get a capture by its ID.

Query Parameters

- **id** (*integer*) – Capture id

Example request:

```
GET /capture HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A capture object

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": 1,
    "serial": "string",
    "statusb64": "string",
    "timeintb64": "string",
    "circbufb64": "string",
    "hmac": "string",
    "start-timestamp": "2020-08-27T10:57:40.207501",
    "end-timestamp": "2020-08-27T10:57:40.207501"
}
```

- 400 Bad Request – bad input parameter

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

DELETE /capture

delete a capture

delete a capture

Query Parameters

- **id** (*integer*) – Capture id

Status Codes

- 204 No Content – Capture has been deleted

  Example response:

```
HTTP/1.1 204 No Content
Content-Type: application/json

{}
```

- 400 Bad Request – bad input parameter

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 404 Not Found – No capture found

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

GET /captures

Get a list of captures ordered by ID.

Get a list of captures

Query Parameters

- **offset** (*integer*) – Return captures starting from this index.
- **limit** (*integer*) – Limit the number of captures returned.
- **tag_id** (*integer*) – Only returns captures from tag_id.

Example request:

```
GET /captures HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – List of captures

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "id": 1,
        "serial": "string",
        "statusb64": "string",
        "timeintb64": "string",
        "circbufb64": "string",
        "hmac": "string",
        "start-timestamp": "2020-08-27T10:57:40.207501",
        "end-timestamp": "2020-08-27T10:57:40.207501"
    }
]
```

- 400 Bad Request – bad input parameter

Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

POST /captures

create a capture

Create a capture

Example request:

```
POST /captures HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "id": 1,
    "serial": "string",
    "statusb64": "string",
    "timeintb64": "string",
    "circbufb64": "string",
    "hmac": "string",
    "start-timestamp": "2020-08-27T10:57:40.207501",
    "end-timestamp": "2020-08-27T10:57:40.207501"
}
```

Status Codes

- 201 Created – Capture created

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "id": 1,
```

(continues on next page)

(continued from previous page)

```
    "serial": "string",
    "statusb64": "string",
    "timeintb64": "string",
    "circbufb64": "string",
    "hmac": "string",
    "start-timestamp": "2020-08-27T10:57:40.207501",
    "end-timestamp": "2020-08-27T10:57:40.207501"
}
```

- 400 Bad Request – invalid input, object invalid

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 409 Conflict – a capture with the same id already exists

  Example response:

```
HTTP/1.1 409 Conflict
Content-Type: application/json

{}
```

1.1.3 TagView

GET /tagviews

Get a list of TagViews.

Get a list of TagViews.

Query Parameters

- **distinctOnTag** (*boolean*) – Return only the latest TagView for each scanned tag.
- **tag_id** (*integer*) – Filter TagViews by tag_id.

Example request:

```
GET /tagviews HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A list of tagview objects ordered from newest to oldest.

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "id": 1,
```

(continues on next page)

(continued from previous page)

```
        "tagserial": "string",
        "timestamp": "2020-08-27T10:57:40.207501"
    }
]
```

GET /tagviews/{id}

Get a tagview

Get a tagview

Parameters

- **id** (*integer*) – Tag view ID

Example request:

```
GET /tagviews/{id} HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A tagview object

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": 1,
    "tagserial": "string",
    "timestamp": "2020-08-27T10:57:40.207501"
}
```

- 404 Not Found – TagView not found.

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

DELETE /tagviews/{id}

Delete a tag view

Parameters

- **id** (*integer*) – Tag view ID

Status Codes

- 204 No Content – TagView deleted

  Example response:

```
HTTP/1.1 204 No Content
Content-Type: application/json
```

(continues on next page)

(continued from previous page)

```
{}
```

- 400 Bad Request – Bad input.

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 404 Not Found – TagView not found

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

GET /tagviews
Get a list of TagViews.

Get a list of TagViews.

Query Parameters

- **distinctOnTag** (*boolean*) – Return only the latest TagView for each scanned tag.
- **tag_id** (*integer*) – Filter TagViews by tag_id.

Example request:

```
GET /tagviews HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A list of tagview objects ordered from newest to oldest.

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "id": 1,
        "tagserial": "string",
        "timestamp": "2020-08-27T10:57:40.207501"
    }
]
```

GET /tagviews/{id}
Get a tagview

Get a tagview

Parameters

- **id** (*integer*) – Tag view ID

Example request:

```
GET /tagviews/{id} HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A tagview object

  Example response:

  ```
  HTTP/1.1 200 OK
  Content-Type: application/json

  {
      "id": 1,
      "tagserial": "string",
      "timestamp": "2020-08-27T10:57:40.207501"
  }
  ```

- 404 Not Found – TagView not found.

  Example response:

  ```
  HTTP/1.1 404 Not Found
  Content-Type: application/json

  {}
  ```

DELETE /tagviews/{id}

Delete a tag view

Parameters

- **id** (*integer*) – Tag view ID

Status Codes

- 204 No Content – TagView deleted

  Example response:

  ```
  HTTP/1.1 204 No Content
  Content-Type: application/json

  {}
  ```

- 400 Bad Request – Bad input.

  Example response:

  ```
  HTTP/1.1 400 Bad Request
  Content-Type: application/json

  {}
  ```

- 404 Not Found – TagView not found

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

1.1.4 User

1.1.5 Token

POST /token

Obtain a bearer token.

Obtain a JWT for interacting with this API.

Example request:

```
POST /token HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "client_id": "string",
    "client_secret": "string"
}
```

Status Codes

- 200 OK – A bearer token that can be used to make calls to other endpoints in this API.

  Example response:

  ```
  HTTP/1.1 200 OK
  Content-Type: application/json

  {
      "token": "string",
      "token_type": "string"
  }
  ```

- 400 Bad Request – No credentials supplied

  Example response:

  ```
  HTTP/1.1 400 Bad Request
  Content-Type: application/json

  {}
  ```

- 401 Unauthorized – Bad credentials

  Example response:

  ```
  HTTP/1.1 401 Unauthorized
  Content-Type: application/json

  {}
  ```

1.2 Consumer API Specification

1.2.1 Tag

GET /tag/{serial}
Get a tag by its serial.

Query Parameters

- **serial** (*string*) – Tag serial

Example request:

```
GET /tag/{serial} HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A tag object

  Example response:

  ```
  HTTP/1.1 200 OK
  Content-Type: application/json

  {
      "serial": "string",
      "timeregistered": "2020-08-27T10:57:40.207501"
  }
  ```

- 400 Bad Request – Bad input parameter.

  Example response:

  ```
  HTTP/1.1 400 Bad Request
  Content-Type: application/json

  {}
  ```

- 404 Not Found – Tag not found.

Scanned

Determines if a tag has been scanned by the current user.

GET /tag/{serial}/scanned
Has a tag with a given serial been scanned by the current user?

Query Parameters

- **serial** (*string*) – Tag serial (Required)

Example request:

```
GET /tag/{serial}/scanned?serial=string HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – True if the tag has a capture taken by the current user.
- 400 Bad Request – Bad input parameter

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 404 Not Found – Tag or user not found.

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

1.2.2 Captures

GET /captures

Get a list of captures for a tag

Query Parameters

- **serial** (*string*) – Tag serial (Required)
- **offset** (*integer*) – Return samples starting from this index.
- **limit** (*integer*) – Limit the number of samples returned.

Example request:

```
GET /captures?serial=string HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A capture object

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "batvoltagemv": 1,
    "tagserial": "string",
    "cursorpos": 1,
    "id": 1,
    "loopcount": 1,
    "md5": "string",
    "status": {
        "brownout": true,
        "clockfail": true,
        "id": 1,
        "lpm5wakeup": true,
```

(continues on next page)

(continued from previous page)


```
        "misc": true,
        "parent_capture": 1,
        "resetsalltime": 1,
        "supervisor": true,
        "watchdog": true
    },
    "timeintmins": 1,
    "timestamp": "2020-08-27T10:57:40.207501",
    "version": 1
}
```


- 400 Bad Request – invalid input, object invalid

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 404 Not Found – Tag with serial not found.

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

POST /captures

Create a capture

Status Codes

- 200 OK – A capture object

  Example response:


```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "batvoltagemv": 1,
    "tagserial": "string",
    "cursorpos": 1,
    "id": 1,
    "loopcount": 1,
    "md5": "string",
    "status": {
        "brownout": true,
        "clockfail": true,
        "id": 1,
        "lpm5wakeup": true,
        "misc": true,
        "parent_capture": 1,
        "resetsalltime": 1,
        "supervisor": true,
        "watchdog": true
```


(continues on next page)

(continued from previous page)

```
    },
    "timeintmins": 1,
    "timestamp": "2020-08-27T10:57:40.207501",
    "version": 1
}
```

- 400 Bad Request – invalid input, object invalid

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 401 Unauthorized – Not authorised. HMAC does not correspond to input data.

  Example response:

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json

{}
```

- 404 Not Found – Tag not found

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

- 409 Conflict – Conflict. A capture with this HMAC already exists. Dead battery or replay attack.

  Example response:

```
HTTP/1.1 409 Conflict
Content-Type: application/json

{}
```

GET /captures/{id}
Get a capture by ID

Example request:

```
GET /captures/{id} HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A capture object

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "batvoltagemv": 1,
    "tagserial": "string",
    "cursorpos": 1,
    "id": 1,
    "loopcount": 1,
    "md5": "string",
    "status": {
        "brownout": true,
        "clockfail": true,
        "id": 1,
        "lpm5wakeup": true,
        "misc": true,
        "parent_capture": 1,
        "resetsalltime": 1,
        "supervisor": true,
        "watchdog": true
    },
    "timeintmins": 1,
    "timestamp": "2020-08-27T10:57:40.207501",
    "version": 1
}
```

- 400 Bad Request – invalid input, object invalid

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 404 Not Found – Capture not found.

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

Samples

Returns samples for a given capture.

GET /captures/{id}/samples
Get samples for a capture.

Query Parameters

- **offset** (*integer*) – Return samples starting from this index.
- **limit** (*integer*) – Limit the number of samples returned.

Example request:

```
GET /captures/{id}/samples HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A list of sample objects

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "capture_id": 1,
        "id": 1,
        "location": {
            "capturesample_id": 1,
            "description": "string",
            "id": 1,
            "timestamp": "2020-08-27T10:57:40.207501"
        },
        "rh": 1.0,
        "temp": 1.0,
        "timestamp": "2020-08-27T10:57:40.207501"
    }
]
```

- 400 Bad Request – bad input parameter

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

1.2.3 Samples

GET /samples

Get unique samples for a tag in a given time range

Query Parameters

- **serial** (*string*) – Tag serial (Required)
- **starttime** (*string*) – start timestamp as an ISO-8601 string. (Required)
- **endtime** (*string*) – end timestamp as an ISO-8601 string. (Required)
- **offset** (*integer*) – Return samples starting from this index.
- **limit** (*integer*) – Limit the number of samples returned.

Example request:

```
GET /samples?serial=string&starttime=string&endtime=string HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A list of samples from newest to oldest

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "capture_id": 1,
        "id": 1,
        "location": {
            "capturesample_id": 1,
            "description": "string",
            "id": 1,
            "timestamp": "2020-08-27T10:57:40.207501"
        },
        "rh": 1.0,
        "temp": 1.0,
        "timestamp": "2020-08-27T10:57:40.207501"
    }
]
```

- 400 Bad Request – bad input parameter

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

1.2.4 Users

POST /users

Create a new user from the Auth0 access token.

Status Codes

- 201 Created – User created
- 400 Bad Request – Invalid input, object invalid

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 403 Forbidden – Invalid JWT

  Example response:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{}
```

- 409 Conflict – Conflict. User already exists.

  Example response:

```
HTTP/1.1 409 Conflict
Content-Type: application/json

{}
```

1.2.5 Me

GET /me

Get current user from the Auth0 access token.

Auth0 ID of the user.

Example request:

```
GET /me HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A user object

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": 1,
    "oauth_id": "string",
    "roles": "string",
    "userinfo": {
        "family_name": "string",
        "given_name": "string",
        "locale": "string",
        "name": "string",
        "nickname": "string",
        "picture": "string",
        "sub": "string",
        "updated_at": "2020-08-27T10:57:40.207501"
    }
}
```

- 403 Forbidden – invalid JWT

  Example response:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{}
```

DELETE /me

Delete current user.

Status Codes

- 204 No Content – User deleted

  Example response:

```
HTTP/1.1 204 No Content
Content-Type: application/json

{}
```

- 400 Bad Request – Bad input parameter

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 404 Not Found – User not found

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

Captures

GET /me/captures

Get a list of captures taken by the current user ordered by most recent first.

Query Parameters

- **distinctOnTag** (*boolean*) – Return only the latest capture for each scanned tag.

Example request:

```
GET /me/captures HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A list of capture objects ordered from newest to oldest

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "batvoltagemv": 1,
        "tagserial": "string",
        "cursorpos": 1,
        "id": 1,
        "loopcount": 1,
```

(continues on next page)

(continued from previous page)


```
        "md5": "string",
        "status": {
            "brownout": true,
            "clockfail": true,
            "id": 1,
            "lpm5wakeup": true,
            "misc": true,
            "parent_capture": 1,
            "resetsalltime": 1,
            "supervisor": true,
            "watchdog": true
        },
        "timeintmins": 1,
        "timestamp": "2020-08-27T10:57:40.207501",
        "version": 1
    }
]
```


- 400 Bad Request – invalid input, object invalid

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 401 Unauthorized – Not authorised. HMAC does not correspond to input data or invalid JWT.

  Example response:

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json

{}
```

- 404 Not Found – Tag not found.

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

POST /me/captures

Create a capture for a user

Status Codes

- 200 OK – A capture object

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

(continues on next page)

(continued from previous page)


```
{
    "batvoltagemv": 1,
    "tagserial": "string",
    "cursorpos": 1,
    "id": 1,
    "loopcount": 1,
    "md5": "string",
    "status": {
        "brownout": true,
        "clockfail": true,
        "id": 1,
        "lpm5wakeup": true,
        "misc": true,
        "parent_capture": 1,
        "resetsalltime": 1,
        "supervisor": true,
        "watchdog": true
    },
    "timeintmins": 1,
    "timestamp": "2020-08-27T10:57:40.207501",
    "version": 1
}
```


- 400 Bad Request – Invalid input, object invalid

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 401 Unauthorized – Not authorised. HMAC does not correspond to input data.

  Example response:

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json

{}
```

- 403 Forbidden – Not authorised. Invalid JWT.

  Example response:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{}
```

- 404 Not Found – Parent tag or user not found.

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

- 409 Conflict – Conflict. A capture with this HMAC already exists. Dead battery or replay attack.

  Example response:

```
HTTP/1.1 409 Conflict
Content-Type: application/json

{}
```

TagViews

GET /me/tagviews
Get a list of TagViews for the current user.

Auth0 ID of the user.

Query Parameters

- **distinctOnTag** (*boolean*) – Return only the latest TagView for each scanned tag.

Example request:

```
GET /me/tagviews HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A list of tagview objects ordered from newest to oldest.

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "tagserial": "string",
        "id": 1,
        "timestamp": "2020-08-27T10:57:40.207501"
    }
]
```

- 403 Forbidden – Invalid JWT

  Example response:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{}
```

POST /me/tagviews
Post a tag view

Status Codes

- 201 Created – TagView created

  Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "tagserial": "string",
    "id": 1,
    "timestamp": "2020-08-27T10:57:40.207501"
}
```

- 400 Bad Request – Invalid input, object invalid

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 403 Forbidden – Invalid JWT

  Example response:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{}
```

- 404 Not Found – Parent resource not found.

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

GET /me/tagviews/{id}
Get a tagview for the current user

Auth0 ID of the user.

Parameters

- **id** (*integer*) – Tag view ID

Example request:

```
GET /me/tagviews/{id} HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A tagview object

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
```

(continues on next page)

(continued from previous page)

```
    "tagserial": "string",
    "id": 1,
    "timestamp": "2020-08-27T10:57:40.207501"
}
```

- 403 Forbidden – Invalid JWT

  Example response:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{}
```

- 404 Not Found – TagView not found.

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

DELETE /me/tagviews/{id}

Delete tag view from the current user.

Status Codes

- 204 No Content – TagView deleted

  Example response:

```
HTTP/1.1 204 No Content
Content-Type: application/json

{}
```

- 400 Bad Request – Bad input.

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 404 Not Found – TagView not found

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

1.2.6 Locations

GET /locations

Get a list of locations for a tag ordered by most recent

Query Parameters

- **serial** (*string*) – Tag serial (Required)
- **starttime** (*string*) – start timestamp as an ISO-8601 string.
- **endtime** (*string*) – end timestamp as an ISO-8601 string.

Example request:

```
GET /locations?serial=string HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A list of locations ordered from newest to oldest.

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "capturesample_id": 1,
        "description": "string",
        "id": 1,
        "timestamp": "2020-08-27T10:57:40.207501"
    }
]
```

- 400 Bad Request – invalid input, object invalid

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 404 Not Found – Location not found.

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

POST /locations

Add location information to a tag

Status Codes

- 200 OK – A capture object

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "capturesample_id": 1,
    "description": "string",
    "id": 1,
    "timestamp": "2020-08-27T10:57:40.207501"
}
```

- 400 Bad Request – invalid input, object invalid

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 401 Unauthorized – Not authorised. The user has no scanned this tag.

  Example response:

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json

{}
```

- 403 Forbidden – Not authorised. Invalid JWT.

  Example response:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{}
```

- 404 Not Found – Parent resource not found.

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

GET /locations/{id}
Get a list of locations for a tag ordered by most recent

Parameters

- **id** (*integer*) – Location ID

Example request:

```
GET /locations/{id} HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – A list of locations ordered from newest to oldest.

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "capturesample_id": 1,
    "description": "string",
    "id": 1,
    "timestamp": "2020-08-27T10:57:40.207501"
}
```

- 400 Bad Request – invalid input, object invalid

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 404 Not Found – Location not found.

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

DELETE /locations/{id}

Delete a location

Status Codes

- 204 No Content – Location deleted

  Example response:

```
HTTP/1.1 204 No Content
Content-Type: application/json

{}
```

- 400 Bad Request – Bad input parameter

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 401 Unauthorized – Not authorised. The user has no scanned this tag.

  Example response:

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json

{}
```

- 403 Forbidden – Not authorised. Invalid JWT.

  Example response:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{}
```

- 404 Not Found – Location not found.

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

PATCH /locations/{id}

Edit location information for a tag

Status Codes

- 200 OK – A capture object

  Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "capturesample_id": 1,
    "description": "string",
    "id": 1,
    "timestamp": "2020-08-27T10:57:40.207501"
}
```

- 400 Bad Request – invalid input, object invalid

  Example response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{}
```

- 401 Unauthorized – Not authorised. The user has no scanned this tag.

  Example response:

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json

{}
```

- 403 Forbidden – Not authorised. Invalid JWT.

  Example response:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{}
```

- 404 Not Found – Parent resource not found.

  Example response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{}
```

CHAPTER

TWO

DEVELOP LOCALLY

wsbackend has external dependencies such as a database and an identity provider. It is designed to be deployed to a cloud provider. Nonetheless it is easy to test this application locally without an internet connection.

2.1 Add / Edit an API endpoint

2.1.1 Update Specification

The API specification should be updated first. These are contained in files named `api.yaml`, which conform to the Swagger OpenAPI standard. It is sometimes easier to edit these files by copying them into and out of the Swagger web application.

2.1.2 Write a Test that Fails

Tavern is used for API testing. Simple tests are defined within yaml files that reside in the tests directory.

The script conftest.py contains pytest fixtures. These obtain access tokens or read environment variables on behalf of test scripts.

Executing `py.test` will run all tests against a live wsbackend server. Its address is set by environment variables:

- WSB_PROTOCOL
- WSB_HOST
- WSB_PORT

The newly developed application must be installed and made to serve pages at this address first! To automate this process I have created a Docker Compose file `docker-compose.test.yml` for quickly running tests locally.

First build all images with:

```
docker-compose -f docker-compose.test.yml build
```

This will take a few minutes the first time, but afterwards the docker layers will be cached so it should copy your new files in within seconds.

Next run tests with:

```
docker-compose -f docker-compose.test.yml up
```

2.1.3 Implement feature. Test until success

Push to GitHub. Tests will run nodejs.yml. A docker image is tested and built on every pull request. Readthedocs will execute on webookhook. If tagging the build then package is deployed to pypi. Build docker image.

2.2 Write a Frontend Application

When writing frontend applications you may want to deploy wsbackend locally. To do this I have created `docker-compose.yml`.

CHAPTER

THREE

CONSUMER API AUTHORIZATION

Some API endpoints require authorization. Only users with a third party account (e.g. Google or Facebook) are granted access. Such accounts cannot be set up without some human interaction. The requirement for user authentication guards against bots filling the wsbackend database with rubbish.

The websensor Web Application uses the Open ID Connect (OIDC) protocol to communicate with a third-party identity provider (IdP). When a user authenticates, the IdP produces an access token. This token is unique to a user and decodes to a number of claims. These include:

- User name.
- Audience (URL of the websensor API for which access has been granted).
- Issuer (URL of the identity provider).
- Issued timestamp.
- Expiry timestamp.
- Scope (granular API permissions).

Tokens are not encrypted. They **must** always be transmitted through a secure channel (HTTPS). The value of tokens is they include a digital signature. If signature verification is successful then the claims can be trusted. In this way access tokens are used to access protected API resources.

The access token also grants the Web Application permission to read some **personal data** about a user (e.g. name and profile picture). Crucially these data are not stored in the Web Application itself. They are requested by making an API call to `/userinfo` API endpoint of the IdP. It can be assumed that the IdP handles these data in a secure and GDPR compliant way.

3.1 Production

In a production environment, the IdP is Auth0.com. Others can be used if they adhere to the OIDC protocol.

Auth0.com acts as an intermediary. It allows users to authenticate with a large number of OAuth2 providers such as Google, GitHub and Facebook.

3.1.1 Obtain an API Access Token

wsfrontend obtains an access token fom the identify provider using the Authorization Code Grant Flow.

3.1.2 Protected API Resource is Called

wsfrontend calls wsbackend endpoints with the access token:

```
curl -X GET "{WSB_HOST}:{WSB_PORT}/api/consumer/v1/me" -H "accept:␣
↪application/json" -H "Authorization: Bearer eyJhbGciOiJS... ZOA4t7Q"
```

3.1.3 Access Token is Validated

The access token signature is generated asymetrically (RS-256). A private key (on Auth0.com) generates the signature. A public key (hosted by Auth0.com) is used for validation.

wsbackend downloads the `public_key` (JWKs) from Auth0.com:

```
GET {IDP_ORIGIN}{IDP_JWKS} => GET https://plotsensor.eu.auth0.com/
.well\unhbox\voidb@x\kern\z@\char`\protect\discretionary{\char\
defaulthyphenchar}{}{}known/jwks.json
```

Signature verification and decoding are performed using PyJWT:

```
decoded = jwt.decode(
            token,
            public_key,
            algorithms=self.algorithms,
            audience={API_AUDIENCE},
            issuer={IDP_ORIGIN}
            )
```

If validation fails, an exception is raised. The token is rejected and the API responds with an error `403: Forbidden`.

3.1.4 Protected Resource Content are Served

If validation succeeds, wsbackend transmits a `200 OK` response to the wsfrontend, along with the requested resource data.

3.2 Testing

For test, the OIDC provider is substituted with a mock https://www.npmjs.com/package/oauth2-mock-server. The test workflow sets this up first on `http://127.0.0.1:3000`.

3.2.1 Obtain an API Access Token

Access tokens are obtained from this using the client-creditials OAuth2 flow.

3.2.2 Test Scripts Call Protected API Resources

wsbackend endpoints are called with the access token in the HTTP header.

3.2.3 Access Token Validated

wsbackend downloads a `public_key` from the mock provider JWKs endpoint. from the mock provider on port 3000:

`GET {IDP_ORIGIN}{IDP_JWKS} => GET http://127.0.0.1:3000/jwks`

Userinfo can also be mocked up.

CHAPTER

FOUR

WSAPIWRAPPER

4.1 Installation

Install from pip.

4.2 Reference

4.2.1 Admin API Wrapper

Tag

class wsapiwrapper.admin.tag.**TagFormat**(*value*)

An enumeration.

FORMAT_HDC2021_TEMPONLY = 2

FORMAT_HDC2021_TRH = 1

class wsapiwrapper.admin.tag.**TagWrapper**(*baseurl: str*, *adminapi_token: str*)

Wraps calls to tag endpoints on the Admin API.

__init__(*baseurl: str*, *adminapi_token: str*)

Constructor for TagWrapper

Parameters

- **baseurl** (*str*) – Websensor backend URL.
- **adminapi_token** (*str*) – Token for accessing the admin API.
- **endpoint_one** (*str*) – Endpoint for returning one resource instance.
- **endpoint_many** (*str*) – Endpoint for returning a list of resource instances.

post(*serial: str = None*, *secretkey: str = None*, *fwversion: str = None*, *hwversion: str = None*, *description: str = None*) → dict

Make a POST request to the *Tag* endpoint.

Parameters

- **serial** – Tag serial string (8 characters).
- **secretkey** – Secret key (16 characters).
- **fwversion** – Tag firmware version.
- **hwversion** – Tag hardware version.

- **description** – Tag description.

Returns dict: A dictionary representing the new tag object.

simulate(*tagid: int*, *frontendurl: str*, *nsamples: int = 100*, *smplintervalmins: int = 100*, *tagformat: wsapiwrapper.admin.tag.TagFormat = <TagFormat.FORMAT_HDC2021_TRH: 1>*, *usehmac: bool = False*, *batvoltagemv: int = 3000*, *bor: bool = False*, *svsh: bool = False*, *wdt: bool = False*, *misc: bool = False*, *lpm5wu: bool = False*, *clockfail: bool = False*, *tagerror: bool = False*) → str

Make a GET request to the *Tag* simulate endpoint.

Parameters

- **tagid** – Database ID of the tag to simulate.
- **frontendurl** – URL of a cuplfrontend instance that will be contained in the simulated tag URL.
- **nsamples** – Number of samples to put in the simulated tag URL.
- **smplintervalmins** – Time interval between samples in minutes.
- **tagformat** – Indicates the datatype for each sample.
- **usehmac** – Specifies whether the hash is HMAC-MD5 or just MD5.
- **batvoltagemv** – Battery voltage of the simulated tag in mV.
- **bor** – Brown-out-Reset flag.
- **svsh** – Supply Voltage Supervisor (high-side) reset flag.
- **wdt** – Watchdog reset flag.
- **misc** – Miscellaneous reset flag.
- **lpm5wu** – Low power mode wake-up flag.
- **clockfail** – Clock failure reset flag.
- **tagerror** – Specify a tag error URL where the circular buffer is omitted.

Returns A string containing a simulated tag URL

Return type str

Capture

class wsapiwrapper.admin.capture.**CaptureWrapper**(*baseurl: str*, *adminapi_token: str*)

Wraps calls to capture endpoints on the Admin API.

__init__(*baseurl: str*, *adminapi_token: str*)

Constructor for CaptureWrapper

Parameters

- **baseurl** (*str*) – Websensor backend URL.
- **adminapi_token** (*str*) – Token for accessing the admin API.
- **endpoint_one** (*str*) – Endpoint for returning one resource instance.
- **endpoint_many** (*str*) – Endpoint for returning a list of resource instances.

get_many(*offset: int = 0*, *limit: int = None*, *tag_id: int = None*) → list

Makes a GET request to endpoint_many.

Parameters

- **offset** (*int*) – Return captures with an ID greater than this number.
- **limit** (*int*) – Number of captures to return.
- **tag_id** (*int*) – Filter by tag_id (optional)

Returns A list of capture dictionaries

Return type list

post (*capturepayload: dict*)
Create a capture in the database directly.

The is made from a dictionary including a list of samples, a tag_id and a user_id. The endpoint is used for testing. It removes the need to encode test vectors with wscodec (introduces a second point of failure in an external library).

Captures can be created with a known list of samples. Then samples are collected from the tag using the consumer API. These are compared with a vector of expected samples. If captures overlap slightly in time it is expected that duplicate samples are removed.

Parameters **capturepayload** (*dict*) – Dictionary following the *capture schema*.

Returns: HTTP response from wsbackend.

class wsapiwrapper.admin.**AdminApiWrapper** (*baseurl: str*, *tokenstr: str*, *endpoint_one: str*, *endpoint_many: str*)
Wraps calls to the wsbackend Admin API

The Admin API is intended for administrators.

__init__ (*baseurl: str*, *tokenstr: str*, *endpoint_one: str*, *endpoint_many: str*)
Constructor for AdminApiWrapper

Parameters

- **baseurl** (*str*) – Websensor backend URL.
- **adminapi_client_id** (*str*) – Client ID API access credential. A long base64 string e.g. SVpP...kO8
- **adminapi_client_secret** (*str*) – Client Secret API access credential. A long base64 string e.g. CM300...1aVB
- **endpoint_one** (*str*) – Endpoint for returning one resource instance.
- **endpoint_many** (*str*) – Endpoint for returning a list of resource instances.

delete (*id: int*)
Make a DELETE request to the *Tag* endpoint.

Parameters **id** (*int*) – ID of the resource to delete

get (*id: int*) → dict
Make a GET request to endpoint_single.

Returns A dictionary representing a tag object.

Return type dict

get_many (*offset: int = 0*, *limit: int = None*, ***kwargs*) → list
Makes a GET request to endpoint_many.

Parameters

- **offset** (*int*) – Return captures with an ID greater than this number.

- **limit** (*int*) – Number of captures to return.

Returns A list of resource dictionaries

Return type list

wsapiwrapper.admin.**api_url**(*baseurl: str*) → str

wsapiwrapper.admin.**request_admin_token**(*baseurl: str*, *adminapi_client_id: str*, *adminapi_client_secret: str*) → str

Request a token from the token endpoint.

A client_id and client_secret are exchanged for a token. This uses the OAuth Client Credentials flow:

1. A POST request is made to the token endpoint of wsbackend.
2. Client ID and Client Secret are validated.
3. Access token is returned.

Returns token received from wsbackend.

Return type str

4.2.2 Consumer API Wrapper

Me

exception wsapiwrapper.consumer.user.**UserAlreadyExistsException**

__init__()
Initialize self. See help(type(self)) for accurate signature.

exception wsapiwrapper.consumer.user.**UserNotFoundException**

__init__()
Initialize self. See help(type(self)) for accurate signature.

class wsapiwrapper.consumer.user.**UserWrapper**(*baseurl: str*, *tokenstr: str = None*)
Wraps users and me endpoints of the Consumer API. Consumers can retrieve more information about themselves (me) than they can about other users.

delete() → None
Delete current user.

Consumer API permits users to delete their own database entry.

Makes a DELETE request to the *Me* endpoint.

Current user is identified by an access token passed to the *constructor*.

Returns None

get() → dict
Get current user.

Retrieves database record for the current user.

Makes a GET request to the *Me* endpoint.

Current user is identified by an access token passed to the *constructor*.

Returns API representation of the current user.

Return type dict

post() → dict
Create new user from an access token.

The sub claim in the decoded token uniquely identifies a user.

Makes a POST request to the *Users* endpoint.

Returns Dictionary representing a user.

static process_status(*status_code: int*, *desc: str = None*) → None
Raise class-specific exceptions corresponding to API errors.

TagView

class wsapiwrapper.consumer.tagview.**TagViewWrapper**(*baseurl: str*, *tokenstr: str*)

__init__(*baseurl: str*, *tokenstr: str*)
Constructor for ConsumerApiWrapper

Parameters

- **baseurl** (*str*) – Websensor backend base URL.
- **tokenstr** (*str*) – OAuth access token.

get(*distinct: bool = False*) → list
Get a list of tag views by the current user.

Current user is identified by an API access token passed to the *constructor*.

Parameters **distinct** (*bool*) – When true only the most recent TagView for each tag will be returned.

Returns A list of timestamped tag views.

Return type list

post(*tagserial: str*) → dict
Record that the current user has viewed a tag.

Makes a POST request to the *TagView* Consumer API endpoint.

Current user is identified by the API access token passed to the *constructor*.

Parameters **tagserial** (*str*) – Base64 serial that uniquely identifies a tag (hardware module).

Returns API representation of the newly created tag view.

Return type dict

Captures

class wsapiwrapper.consumer.mecapture.**MeCaptureWrapper**(*baseurl: str*, *tokenstr: str = None*)

Create and retrieve captures linked to the current user.

get(*distinct: bool = False*)

Get a list of captures made by the current user.

Makes a GET request to the *MeCaptures* endpoint.

Current user is identified by an access token passed to the *constructor*.

Parameters **distinct** (*bool*) – When true, only the most recent capture is returned for each tag.

Returns A list of capture dictionaries.

Return type list

post(*circbufb64: str*, *serial: str*, *statusb64: str*, *timeintb64: str*, *versionStr: str*) → dict

Create a new capture. Record that it was made by the current user.

See *wsapiwrapper.consumer.capture.CaptureWrapper.post()*.

Current user is identified by an access token passed to the *constructor*.

Tag

class wsapiwrapper.consumer.tag.**TagWrapper**(*baseurl: str*, *tokenstr: str = None*)

Wraps calls to tag endpoints on the Consumer API

get(*tagserial: str*) → dict

Parameters **tagserial** (*str*) – Base64 serial that uniquely identifies a tag (hardware module).

Returns: Tag dictionary as described *here*.

Scanned

class wsapiwrapper.consumer.tagscanned.**TagScannedWrapper**(*baseurl: str*, *tokenstr: str = None*)

Wraps a Consumer API endpoint for determining if a tag has been scanned by a user.

get(*tagserial: str*) → bool

Makes a GET request to the *TagScanned* Consumer API endpoint.

Current user is identified by an access token passed to the *constructor*.

Parameters **tagserial** (*str*) – Base64 serial that uniquely identifies a tag (hardware module).

Returns True if the tag has been scanned a user identified in the token header. False otherwise.

Return type bool

Capture

class wsapiwrapper.consumer.capture.**CaptureWrapper**(*baseurl: str*, *tokenstr: str = None*)
Wraps capture endpoints of the Consumer API.

get(*capture_id: int*) → dict
Get one capture by its ID.

Makes a GET request to the Capture endpoint.

Parameters **capture_id** (*int*) – Capture ID to retrieve.

Returns Capture dictionary returned by the API.

Return type dict

get_list(*serial: str*, *offset: int = 0*, *limit: int = None*) → list
Get a list of captures for a tag ordered newest first.

The list can be paginated so that each call returns 'limit' captures.

Parameters

- **serial** (*str*) – Base64 serial that identifies the tag to read captures from.
- **offset** (*int*) – Start list at the nth capture for pagination.
- **limit** (*int*) – Limit the list length for pagination.

Returns A list of capture dictionaries.

Return type list

get_samples(*capture_id: int*, *offset: int = 0*, *limit: int = None*) → list
Get a list of samples from a capture.

Makes a GET request to the *CaptureSamples* endpoint.

The list can be paginated so that each call returns 'limit' samples at an 'offset' from the first.

Parameters

- **capture_id** (*int*) – Capture ID to retrieve.
- **offset** (*int*) – Start list at the nth sample for pagination.
- **limit** (*int*) – Limit the list length for pagination.

Returns A list of samples.

Return type list

post(*circbufb64: str*, *serial: str*, *statusb64: str*, *timeintb64: str*, *vfmtb64: str*) → dict
Create a new capture from parameters encoded by wscodec.

These data are included in URL parameters passed to wsfrontend when a tag is scanned.

Makes a POST request to the *Capture* endpoint, which unwraps the circular buffer and decodes samples.

Parameters

- **circbufb64** (*str*) – Circular buffer containing base64 encoded samples. Ouptut by wscodec.
- **serial** (*str*) – Base64 serial identifying the tag the capture has originated from.
- **statusb64** (*str*) – Base64 encoded status information (e.g. battery level). Output by wscodec.

- **timeintb64** (*str*) – Base64 encoded time interval between samples. Output by wscodec.
- **vfmtb64** (*str*) – Tag version string. See wscodec.

Returns Capture dictionary.

Return type dict

Sample

class wsapiwrapper.consumer.sample.**SampleWrapper**(*baseurl: str*, *tokenstr: str = None*)

Wraps samples endpoint of the Consumer API.

get_samples(*serial: str*, *starttime: str*, *endtime: str*, *offset: int = 0*, *limit: int = None*) → list

Get a list of temperature/humidity samples collected by a tag.

A time window can be specified so that only samples that fall between starttime and endtime are returned.

Makes a GET request to the *Samples* endpoint.

Parameters

- **serial** (*str*) – Base64 string. Identifies a tag to return samples from.
- **starttime** (*str*) – Start datetime as an ISO-8601 string.
- **endtime** (*str*) – End datetime as an ISO-8601 string.
- **offset** (*int*) – Start list at the nth sample for pagination.
- **limit** (*int*) – Limit the list length for pagination.

Returns A list of samples. Each is a dictionary.

Return type list

Location

class wsapiwrapper.consumer.location.**LocationWrapper**(*baseurl: str*, *tokenstr: str*)

Wraps location endpoints of the Consumer API.

__init__(*baseurl: str*, *tokenstr: str*)

Constructor for ConsumerApiWrapper

Parameters

- **baseurl** (*str*) – Websensor backend base URL.
- **tokenstr** (*str*) – OAuth access token.

delete(*location_id: int*) → int

Delete one location by ID.

Makes a DELETE request to the *Location* endpoint. To delete locations the current user must have scanned the tag.

Parameters **location_id** (*int*) – Location ID to delete.

Returns HTTP status code.

Return type int

get (*location_id: int*) → dict
Get one location by its ID.

Makes a GET request to the *Location* endpoint.

Parameters **location_id** (*int*) – Location ID to retrieve.

Returns Location dictionary returned by the API.

Return type dict

get_list (*tagserial: str*, *starttime: str = None*, *endtime: str = None*) → list
Get a list of locations for a tag.

Optionally a time window can be specified, so that only location timestamps within that window will be returned.

Makes a GET request to the *Locations* endpoint.

Parameters

- **tagserial** – Base64 serial identifying a tag.
- **starttime** – Start datetime as an ISO-8601 string.
- **endtime** – End datetime as an ISO-8601 string.

Returns A list of Location dictionaries.

Return type list

patch (*location_id: int*, *description: str*) → dict
Change description of an existing location.

The current user must have scanned the tag.

Parameters

- **location_id** (*int*) – Location ID to modify.
- **description** – New tag location (e.g. above the fireplace).

Returns Location dictionary returned by the API.

Return type dict

post (*capturesample_id: int*, *description: str*) → dict
Annotate a sample with location.

The timestamp of a location corresponds to that of its parent sample. All samples can be traced back to the tag that created them. This must have been scanned by the current user.

Parameters

- **capturesample_id** (*int*) – Capturesample ID to annotate.
- **description** (*str*) – The tag location e.g. under the stairs.

Returns Location dictionary returned by the API and converted from JSON.

Return type dict

static process_status (*status_code: int*, *desc=''*) → None
Raise an exception in response to an HTTP error code.

Parameters **status_code** – HTTP status code.

exception wsapiwrapper.consumer.location.**NoScanOnTagException**
Location data cannot be modified on tags that have not been scanned.

__init__()
Initialize self. See help(type(self)) for accurate signature.

exception wsapiwrapper.consumer.**ConsumerAPIException**(*message*)

__init__(*message*)
Initialize self. See help(type(self)) for accurate signature.

class wsapiwrapper.consumer.**ConsumerApiWrapper**(*baseurl: str*, *tokenstr: str = None*)

__init__(*baseurl: str*, *tokenstr: str = None*)
Constructor for ConsumerApiWrapper

Parameters

- **baseurl** (*str*) – Websensor backend base URL.
- **tokenstr** (*str*) – OAuth access token.

static process_status(*status_code: int*, *desc: str = None*)
Raise exception if an HTTP error occurs.

Parameters

- **status_code** (*int*) – HTTP status code.
- **desc** (*str*) – Description of error.

Returns None

exception wsapiwrapper.consumer.**Exception400**(*message='400 Bad input'*)

__init__(*message='400 Bad input'*)
Initialize self. See help(type(self)) for accurate signature.

exception wsapiwrapper.consumer.**Exception401**(*message='Not authorised to access this resource. Invalid JWT or bad HMAC.'*)

__init__(*message='Not authorised to access this resource. Invalid JWT or bad HMAC.'*)
Initialize self. See help(type(self)) for accurate signature.

exception wsapiwrapper.consumer.**Exception403**(*message='Forbidden.'*)

__init__(*message='Forbidden.'*)
Initialize self. See help(type(self)) for accurate signature.

exception wsapiwrapper.consumer.**Exception404**(*message='404 Resource not found'*)

__init__(*message='404 Resource not found'*)
Initialize self. See help(type(self)) for accurate signature.

exception wsapiwrapper.consumer.**Exception409**(*message='409 Resource already exists'*)

__init__(*message='409 Resource already exists'*)
Initialize self. See help(type(self)) for accurate signature.

class wsapiwrapper.**ApiWrapper**(*baseurl: str*, *tokenstr: str*)
Wraps calls to wsbackend APIs.

Uses the Requests HTTP library to call wsbackend web APIs.

__init__ (*baseurl: str*, *tokenstr: str*)
Constructor for ApiWrapper.

Parameters **baseurl** (*str*) – Websensor backend base URL.

auth_header (*tokenstr*)
Get a dictionary of headers. One contains an API access token. This is needed for some API requests to be authorized.

Parameters **tokenstr** (*str*) – API access token

Returns a dictionary containing two HTTP headers.

Return type dict

CHAPTER

FIVE

WSBACKEND

5.1 Installation

This tutorial assumes you have Git and Python 3 installed.

5.1.1 Install Prerequisites

One of the dependencies (Postgres library Psycopg2) should be built from source.

GCC Compiler

To install GCC on Debian, open a terminal window and type:

```
$ sudo apt install build-essential
```

To check gcc is installed, type:

```
$ gcc --version
```

Python Header Files

To install the Python 3 header files type:

```
$ sudo apt-get install python3-dev
```

Libpq-dev

This To install libbq-dev type:

```
$ sudo apt-get install libpq-dev
```

5.1.2 Clone the Repo

First you will need to download wsbackend by cloning its Git repository. Open a terminal window, browse to a folder of your choice and type:

```
$ git clone https://github.com/websensor/wsbackend.git
```

The GitHub repository will be cloned to a folder named `wsbackend`. Open this by typing:

```
$ cd wsbackend
```

5.1.3 Create a Virtual Environment

We will use virtualbox to create a virtual environment. This will isolate our wsbackend installation from the rest of the system.

First install virtualenv:

```
$ sudo pip3 install virtualenv
```

Next create a virtual environment named `wsbackend_env`:

```
$ virtualenv wsbackend_env
```

5.1.4 Activate the Virtual Environment

In order for `pip` to install dependencies into the virtual enviornment, you must activate it first with:

```
$ source wsbackend_env/bin/activate
```

5.1.5 Install Dependencies

wsbackend is dependent on a number of Python packages including flask. These are listed in requirements.txt.

To install all requirements, type:

```
$ pip3 install -r requirements.txt
```

5.1.6 Define Environment Variables

Auth0

5.1.7 Install Gunicorn

There are many options for serving this Flask application. In this tutorial we will use Gunicorn. It is the easiest to install:

```
$ pip3 install gunicorn
```

Next run the application with:

```
$ gunicorn main:app
```

5.2 Environment Variables

5.2.1 Host address

WSB_PROTOCOL

Default: `http://`

Protocol of wsbackend.

WSB_HOST

Default: `localhost`

Hostname of wsbackend.

WSB_PORT

Default: `5000`

Port of wsbackend.

5.2.2 Identity Provider

IDP_PROTOCOL

Default: `http://`

Protocol of the OpenID Connect Identity Provider.

Defaults to the protocol of the mock IdP.

IDP_HOST

Default: `localhost`

Hostname of the OpenID Connect Identity Provider.

IDP_PORT

Default: `3000`

Port of the OpenID Connect Identity Provider.

IDP_JWKS

Default: `/jwks`

Endpoint on the IdP from which the public JSON Web Key set can be downloaded.

Defaults to the mock IdP JWKs endpoint.

5.2.3 Database

DB_HOST

Default: `localhost`

Host where the database is running.

DB_PORT

Default: `5432`

Port number to connect to the database.

DB_USER

Default: `postgres`

Database user name.

DB_PASS

Database password.

No default for security reasons. This variable must be set before runtime.

5.2.4 Consumer API

API_ISSUER

Default: `http://localhost:3000`

API issuer claim. Must correspond to the issuer of the API access token.

API_AUDIENCE

Default: `mock_api_audience`

Access tokens signed by the IdP must contain this audience claim. Without it, the consumer API will reject the token and deny access to an endpoint that requires authorization.

Defaults to a value used by the mock IdP.

5.2.5 Admin API

ADMINAPI_AUDIENCE

Default: `default_adminapi_audience`

Access tokens issued using the client credentials flow will contain this audience claim.

ADMINAPI_CLIENTID

Default: `default_adminapi_clientid`

A token request using the client credentials flow must contain this `client_id`. This is a shared secret between the sender and `wsbackend`.

ADMINAPI_CLIENTSECRET

A token request using the client credentials flow must contain this `client_secret`. This is a shared secret between the sender and `wsbackend`.

No default for security reasons. This variable must be set before runtime.

5.3 Reference

5.3.1 API Layer

5.3.2 Service Layer

5.3.3 Model Layer

class wsapiwrapper.**ApiWrapper**(*baseurl: str*, *tokenstr: str*)

Wraps calls to wsbackend APIs.

Uses the Requests HTTP library to call wsbackend web APIs.

__init__(*baseurl: str*, *tokenstr: str*)

Constructor for ApiWrapper.

Parameters **baseurl** (*str*) – Websensor backend base URL.

auth_header(*tokenstr*)

Get a dictionary of headers. One contains an API access token. This is needed for some API requests to be authorized.

Parameters **tokenstr** (*str*) – API access token

Returns a dictionary containing two HTTP headers.

Return type dict

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

W

HTTP ROUTING TABLE

INDEX

Symbols

A

C

D

E

F

G

L

M

N

P

R

S

T

U

W