

fundamentals

- 1 Bash
 - 1.1 Necessary (++) and useful (+) programs
 - 1.2 Basic Login
 - 1.3 HTTP requests
 - 1.4 GET
 - 1.5 PUT
 - 1.6 DELETE
 - 1.7 POST
- 2 Python
 - 2.1 Necessary (++) and useful (+) packages
 - 2.2 Basic Login
 - 2.3 HTTP requests
 - 2.4 GET
 - 2.5 PUT
 - 2.6 DELETE
 - 2.7 POST
- 3 R
 - 3.1 Necessary (++) and useful (+) packages
 - 3.2 Basic Login
 - 3.3 HTTP requests
 - 3.4 GET
 - 3.5 PUT
 - 3.6 DELETE
 - 3.7 POST

Bash

Necessary (++) and useful (+) programs

- [curl](#) (++)
- [jq](#) (+)

Basic Login

Using [commandline](#) requires [curl](#) to submit [http requests](#) to the [API](#).

- `-c`: writes the cookies (including the required auth-token) to a local file, in this case `cfile`
- `-s`: silent output
- `https://sandbox.sensor.awi.de/rest/sensors/contacts/login`: the API endpoint that is called
- `-X`: the type of request, in this case `POST` (see [http requests](#))
- `-o`: specifies the output (if not prompted to standard output aka the console), in this case the output is piped to `/dev/null`
- `-d`: introduces a data field that is submitted along the `POST` request

The following request carries two data fields, each contains a key value pair, together they deliver the authentication credentials.

```
curl -c cfile \  
  -s \  
  -X POST 'https://sandbox.sensor.awi.de/rest/sensors/contacts/login' \  
  -o /dev/null \  
  -d 'username=<yourUserName>' \  
  -d 'authPassword=yourSecretPassword'
```

The `cfile` looks like this:

```
# Netscape HTTP Cookie File  
# https://curl.haxx.se/docs/http-cookies.html  
# This file was generated by libcurl! Edit at your own risk.  
  
sandbox.sensor.awi.de      FALSE      /sensorManagement-web      FALSE      0      JSESSIONID  
RaCR4vWoHo7xKn92ohy2izuxjldw68olmHDjhA0.frame-test1  
.awi.de      TRUE      /      FALSE      1670407206      x-auth-token  
af9d8dce8jal78271057345p5444022
```

and can be removed **after** successful operation by

```
rm -v cfile
```

HTTP requests

GET

This gets basic information about an item.

```
curl -X GET 'https://sandbox.sensor.awi.de/rest/sensors/item/getItem/456'
```

or nicely parsed (because piped into jq):

```
curl -X GET 'https://sandbox.sensor.awi.de/rest/sensors/item/getItem/456' | jq .
```

PUT

You need proper privileges to modify items.

This puts a new item to sensor.awi.de.

- -b: a submitted cookie (remember the login cfile?)
- -X: which type of [http request](#) is performed, in this case PUT
- -H: header for the request, in this case
 - the content is json format
 - json format shall be accepted
- -d: introduces a data field that is submitted along the PUT request, this is literally the submitted json

```
curl -b cfile \  
  -X PUT 'https://sandbox.sensor.awi.de/rest/sensors/item/createItem' \  
  -H 'Content-Type: application/json' \  
  -H 'Accept: application/json' \  
  -d '{"description": "string", "shortName": "testnastring", "longName": "string", "serialNumber":  
"string", "manufacturer": "string", "parentID": 0, "applicationTypeID": 0, "model": "string",  
"inventoryNumber": "string", "itemStatusID": 2, "itemTypeID": 110, "id": 0 }'
```

- "description": *string* --> for the overview tab
- "shortName": *string* --> for the overview tab
- "longName": *string* --> for the overview tab
- "serialNumber": *string* --> for the overview tab
- "manufacturer": *string* --> for the overview tab
- "parentID": *integer* --> left blank if item should be parentless, otherwise enter numeric id of parental item (think about access rights for the parent, too)
- "applicationTypeID": *ignore*
- "model": *string* --> for the overview tab
- "inventoryNumber": *string* --> for the overview tab
- "itemStatusID": *integer* --> either know the necessary id or ask <https://sandbox.sensor.awi.de/rest/sensors/item/getAllItemStatuses>, in this case 2 refers to *construction*
- "itemTypeID": *integer* --> either know the necessary id or ask <https://sandbox.sensor.awi.de/rest/sensors/item/getAllItemCategories>, in this case 110 is a *sediment_grab*
- "id": *integer* --> set 0, since the item is supposed to be completely new

DELETE

You need proper privileges to modify items.

This deletes a certain user with a specific role from a certain item.

- -b: locally stored cookie file
- -X: which type of [http request](#) is performed, in this case DELETE
- -H: header

```
curl -b cfile \  
  -X DELETE \  
  -H 'Accept: application/json' \  
  'https://sandbox.sensor.awi.de/rest/sensors/contacts/deleteContactFromDevice/10877/80/29'
```

- 10877: the item ID
- 80: the user ID
- 29: the contact role ID

POST

Basically a POST was already done by token creation. For most purposes POST appears to be of minor importance.

Python

Necessary (++) and useful (+) packages

- [requests](#) (++)
- [json](#) (++)
- [re](#) (+)
- [pandas](#) (+)
- [datetime](#) (+)
- [itertools](#) (+)

```
pip install -r requirements.txt
```

json, re, itertools, and datetime are built-in moduls and do not need to be installed separately.

Basic Login

The carried body contains the authentication credentials as a dict element. The auth cookie is then extracted (theToken) from the response of the POST request (see [http requests](#)).

```
import requests  
import json  
  
auth = requests.post('https://sandbox.sensor.awi.de/rest/sensors/contacts/login'  
                    , data = {'username': <yourUserName>, 'authPassword': <yourSecretPassword>}  
)  
theToken = auth.cookies['x-auth-token']
```

HTTP requests

GET

This gets basic information about an item. The result is a dictionary object.

```
a = requests.get('https://sandbox.sensor.awi.de/rest/sensors/item/getItem/456')  
theItem = json.loads(a.content)
```

PUT

You need proper privileges to modify items.

This puts a new item to [sensor.awi.de](#).

```
requests.put('https://sandbox.sensor.awi.de/rest/sensors/item/createItem'
, data = json.dumps({
    "description": "string"
, "shortName": "anotherteststring"
, "longName": "string"
, "serialNumber": "string"
, "manufacturer": "string"
, "parentID": 0
, "applicationTypeID": 0
, "model": "string"
, "inventoryNumber": "string"
, "itemStatusID": 2
, "itemTypeID": 110
, "id": 0
}))
, headers = {"content-type": "application/json"}
, cookies = {'x-auth-token': theToken}
)
```

- "description": *string* --> for the overview tab
- "shortName": *string* --> for the overview tab
- "longName": *string* --> for the overview tab
- "serialNumber": *string* --> for the overview tab
- "manufacturer": *string* --> for the overview tab
- "parentID": *integer* --> left blank if item should be parentless, otherwise enter numeric id of parental item (think about access rights for the parent, too)
- "applicationTypeID": *ignore*
- "model": *string* --> for the overview tab
- "inventoryNumber": *string* --> for the overview tab
- "itemStatusID": *integer* --> either know the necessary id or ask <https://sandbox.sensor.awi.de/rest/sensors/item/getAllItemStatuses>, in this case 2 refers to *construction*
- "itemTypeID": *integer* --> either know the necessary id or ask <https://sandbox.sensor.awi.de/rest/sensors/item/getAllItemCategories>, in this case 110 is a *sediment_grab*
- "id": *integer* --> set 0, since the item is supposed to be completely new

DELETE

You need proper privileges to modify items.

This deletes a certain user with a specific role from a certain item.

```
requests.delete('https://sandbox.sensor.awi.de/rest/sensors/contacts/deleteContactFromDevice/10877/80/29'
, cookies = {'x-auth-token': theToken}
, headers = {'content-type': 'application/json'}
)
```

- 10877: the item ID
- 80: the user ID
- 29: the contact role ID

POST

Basically a POST was already done by token creation. For most purposes POST appears to be of minor importance.

R

Necessary (++) and useful (+) packages

- [httr](#) (++)
- [lubridate](#) (+)
- [jsonlite](#) (+)
- [stringr](#) (+)

```
install.packages(c('httr', 'lubridate', 'jsonlite', 'stringr'), dep = TRUE)
```

Basic Login

The carried body contains the authentication credentials as a list element. The auth cookie is then extracted (`theToken`) from the response of the `POST` request (see [http requests](#)).

```
library('httr')

x <- POST(url = 'https://sandbox.sensor.awi.de/rest/sensors/contacts/login'
, body = list("username" = '<yourUserName>', "authPassword" = 'yourSecretPassword')
, encode = "form"
)
theToken <- x$cookies$value[2]
```

HTTP requests

GET

This gets basic information about an item. The result is a list object.

```
GET(url = 'https://sandbox.sensor.awi.de/rest/sensors/item/getItem/456')
```

PUT

You need proper privileges to modify items.

This puts a new item to [sensor.awi.de](#).

- `url`: the API endpoint to call
- `add_headers`: contains the wrapped auth cookie
- `body`: a list (that is then translated to json) that holds the info about the item to be created
- `encode`: indicates how the submitted values are encoded, in this case we preferred json

```
PUT(url = 'https://sandbox.sensor.awi.de/rest/sensors/item/createItem'
, add_headers("x-auth-token" = theToken)
, body = list(description= "string"
, shortName = "testnashalalastring"
, longName = "string"
, serialNumber = "string"
, manufacturer = "string"
, parentID = 0
, applicationTypeID = 0
, model = "string"
, inventoryNumber = "string"
, itemStatusID = 2
, itemTypeID = 110
, id= 0
)
, encode = 'json'
)
```

- `"description"`: *string* --> for the overview tab
- `"shortName"`: *string* --> for the overview tab
- `"longName"`: *string* --> for the overview tab
- `"serialNumber"`: *string* --> for the overview tab
- `"manufacturer"`: *string* --> for the overview tab
- `"parentID"`: *integer* --> left blank if item should be parentless, otherwise enter numeric id of parental item (think about access rights for the parent, too)
- `"applicationTypeID"`: *ignore*
- `"model"`: *string* --> for the overview tab
- `"inventoryNumber"`: *string* --> for the overview tab
- `"itemStatusID"`: *integer* --> either know the necessary id or ask <https://sandbox.sensor.awi.de/rest/sensors/item/getAllItemStatuses>, in this case 2 refers to *construction*
- `"itemTypeID"`: *integer* --> either know the necessary id or ask <https://sandbox.sensor.awi.de/rest/sensors/item/getAllItemCategories>, in this case 110 is a *sediment_grab*
- `"id"`: *integer* --> set 0, since the item is supposed to be completely new

DELETE

You need proper privileges to modify items.

This deletes a certain user with a specific role from a certain item.

```
DELETE(url = 'https://sandbox.sensor.awi.de/rest/sensors/contacts/deleteContactFromDevice/10877/80/29'  
  , add_headers("x-auth-token" = theToken)  
  )
```

- 10877: the item ID
- 80: the user ID
- 29: the contact role ID

POST

Basically a `POST` was already done by token creation. For most purposes `POST` appears to be of minor importance.