

---

# **AnonAPI Documentation**

***Release 1.7.0***

**Sjoerd Kerkstra**

**Feb 16, 2023**



**CONTENTS:**

<b>1</b>	<b>Getting started</b>	<b>1</b>
<b>2</b>	<b>Tutorials</b>	<b>3</b>
<b>3</b>	<b>Usage</b>	<b>5</b>
<b>4</b>	<b>Command reference</b>	<b>7</b>
<b>5</b>	<b>For developers</b>	<b>9</b>
5.1	Quick start . . . . .	9
5.2	Getting started . . . . .	9
5.3	Usage . . . . .	10
5.4	Tutorials . . . . .	16
5.5	Command reference . . . . .	20
5.6	Concepts . . . . .	28
5.7	Troubleshooting . . . . .	33
5.8	Python Usage . . . . .	36
5.9	Modules . . . . .	42
5.10	Release Notes . . . . .	45
<b>6</b>	<b>Indices and tables</b>	<b>47</b>
	<b>Python Module Index</b>	<b>49</b>
	<b>Index</b>	<b>51</b>



## GETTING STARTED

If you have not installed anonapi before, try *Getting started* or *Quick start*



## TUTORIALS

Step-by-step instruction on how to anonymize data can be found in the *Tutorials* section





## USAGE

More detailed than tutorials, The *Usage* section contains details on things like canceling jobs, working with multiple jobs at once, etc.



## COMMAND REFERENCE

The *command reference section* Lists all available anonapi commands and their options



## FOR DEVELOPERS

- Full API reference can be found in *Modules*
- Code examples and pointers for testing are in the *python usage* section
- For code standard, PR guide and tips on updating the docs see *<no title>*

### 5.1 Quick start

- Get the url for an anonymization api server
- Install Python 3.6 or higher (see *Installation*)
- Open a command line terminal and type the following:

```
$ pip install anonapi
$ anon settings user set-username z1234      # replace z1234 with your username
$ anon server add server1 https://anonapi    # replace https://anonapi with actual url
$ anon server activate server1
$ anon server jobs                          # get some info on the jobs running on server1
```

More commands can be found in *Usage* and in the *Command reference*

For more details on installation and configuration see *gettings started*.

### 5.2 Getting started

#### 5.2.1 Installation

To install AnonAPI, run this command in a *command prompt*:

```
$ pip install anonapi
```

This is the preferred method to install AnonAPI, as it will always install the most recent stable release.

If you don't have *pip* installed, this *Python installation guide* can guide you through the process.

If you installed anonapi before and want to upgrade to the latest version:

```
$ pip install --upgrade anonapi
```

## 5.2.2 Configuration

The Command Line Interface (CLI) needs to know a few things before it can be used.

### Add a server to the CLI

The CLI communicates with one or more API servers. At least one of these should be added using the following steps: First Find the url of an IDIS anonymization web API. An overview of servers within the radboudumc can be found [here](#).

Lets say the server address is <https://anonapi.org/server1>. you can now add this server as 'server1':

```
$ anon server add server1 https://anonapi.org/server1
```

You can now see the new server in the server list:

```
$ anon server list
```

Activate server1. All subsequent commands will use this server.

```
$ anon server activate server1
```

### Configure credentials

To make calls to any IDIS web API, the CLI needs to know which credentials to use. Do the following:

Set your username. For radboudumc this is your z-number. To set z1234567 as your username:

```
$ anon settings user set-username z1234567
```

Obtain an API token. This might require your z-number password)

```
$ anon settings user get-token
```

## 5.3 Usage

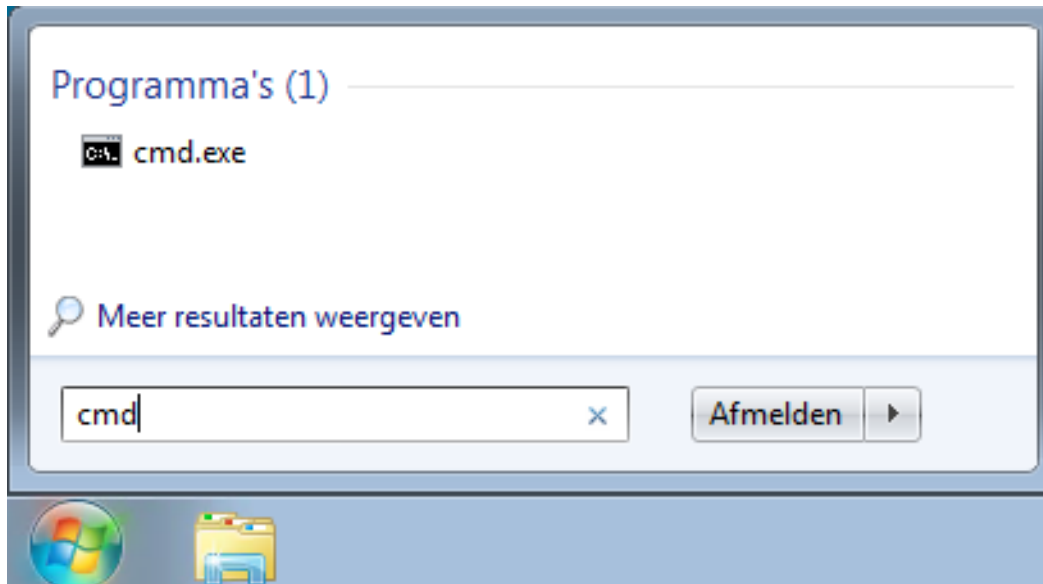
Information on how to combine one or more anonapi *commands* to do things like resetting jobs, working with multiple jobs. For more detailed step-by-step instructions on performing a specific task, see *Tutorials*

### 5.3.1 Starting a command prompt

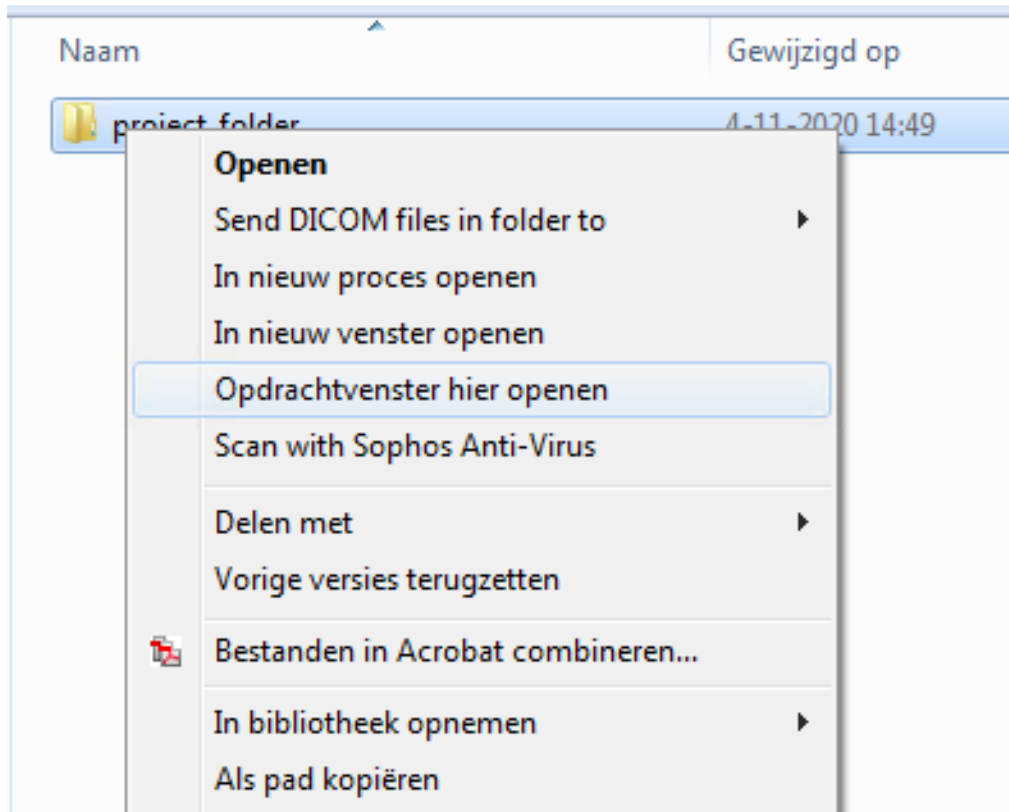
Anonapi commands are given from a command line or command prompt:

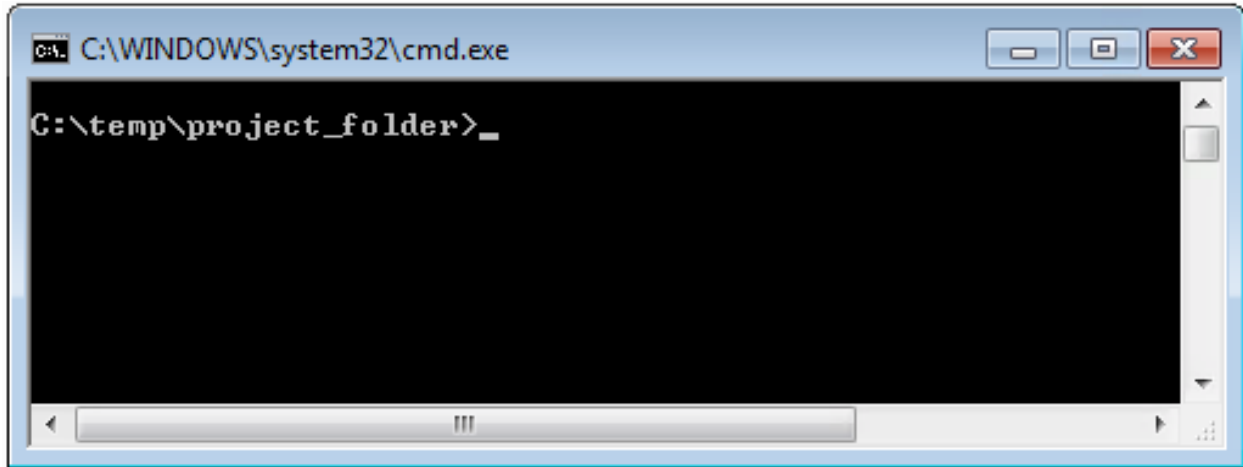
## Windows

To start a command prompt: press the windows start menu -> type cmd press enter



To start a command prompt in a specific folder: find the folder in windows explorer, then shift + right-click on that folder -> Open command prompt here (Dutch: *Opdrachtvenster hier openen*).





**Linux** This varies between distributions. Common ways are pressing Meta+t, Konsole on ubuntu, or Terminal. For more info see [here](#)

### 5.3.2 Getting info on commands

All CLI commands have the following form:

```
$ anon <command-group> <command> <arguments>    # General form
$ anon job info 1234                             # Example: get info for job '1234'
$ anon server activate server2                   # Example: Activate the server named
→ 'server2'
```

A list of commands for any command group can be printed this way:

```
$ anon job          # Shows all commands for command group 'job'
$ anon settings     # Shows all commands for command group 'settings'
$ anon <command-group> # General form. Shows all commands for any command group
```

Detailed help on commands is often available directly from the command line by adding `--help` to any command. For example:

```
$ anon job info --help    # Shows detailed help for job info
$ anon server activate --help # Shows detailed help for server activate
```

### 5.3.3 Information about jobs

```
$ anon server jobs    # shows last 100 jobs on server
$ anon job info 123   # shows details for job with id 123
```

---

**Tip:** see *Job status codes* for more information on job status

---



### 5.3.4 Cancel or restart jobs

```
$ anon job reset 123    # reset job with id 123
$ anon job cancel 123   # cancel job with id 123
```

### 5.3.5 Multiple jobs at once (batch)

More information on job batches: *batch*

```
$ cd C:/myfolder          # any folder you want. One folder can only contain one batch.
$ anon batch init          # initialises an empty batch
$ anon batch add 10 11 13  # add three job ids to this batch
$ anon batch add 20-35     # add fifteen job ids: 20 through to 35
$ anon batch status        # print info for all jobs in batch
$ anon batch               # see other commands including reset and cancel all
```

### 5.3.6 Creating jobs

The general procedure for creating a jobs is as follows:

1. *open a terminal*
2. create a *mapping* using the *map init* command
3. edit the mapping to suit your needs. Most commands for this are in the *map* command group
4. based on the mapping, run the *create from-mapping* command
5. monitor your jobs progress with the *batch status* command

Two specific cases are shown below:

### 5.3.7 Anonymize files from PACS

In this example we want to retrieve and anonymize studies from PACS

#### Quick example

- Create a folder for your project (will hold a record of jobs created)
- Open a *command prompt* in this folder
- Then type the following:

```
$ anon map init          # create a mapping at the source of the data
$ anon map edit          # set correct paths, add studyUIDs or accession numbers
$ anon create from-mapping # create jobs on anonymization server
$ anon batch status       # monitor the progress of your jobs
```

## Detailed example

For this example we want to retrieve and anonymize the following studies from PACS:

- A study with AccessionNumber 123456.1234567
- A study with AccessionNumber 123456.2234568
- A study with StudyInstanceUID 123.1232.23.24

To do this, follow these steps:

```
$ anon map init
> Initialised example mapping in anon_mapping.csv

$ anon map edit    # opens mapping for editing
```

Now edit the mapping until it looks like this:

```
## Description ##
Mapping created February 12 2020

## Options ##
project,          Wetenschap-Algemeen
destination_path, \\server\share\myoutput

## Mapping ##
source,           patient_id, patient_name, description
accession_number:123456.1234567, 001,      Patient2,      Test PACS project
accession_number:123456.2234568, 002,      Patient2,      Test PACS project
study_instance_uid:123.1232.23.24, 003,      Patient3,      Test PACS project
```

Now close the editor and run *anon create from-mapping*:

```
$ anon create from-mapping
> This will create 3 jobs on p01, for projects '['Wetenschap-Algemeen']' etc..
> Done
```

To monitor the status of your created jobs, use *anon batch status*:

```
$ anon batch status
```

## 5.3.8 Anonymize files from a share

In this example we will anonymize data from three folders on a share

## Quick example

- Create a folder for your project (will hold your mapping and record of jobs created)
- Open a *command prompt* in this folder
- Then type the following

```
$ anon map init          # create a new mapping, make it active
$ anon map edit          # set correct paths, remove example rows

# now add three folders to the mapping
$ anon map add-study-folders patient1/study patient2/study patient3/study_fixed

$ anon map edit          # now set the anonymized names for the added studies
$ anon create from-mapping # create jobs on anonymization server

$ anon batch status      # monitor the progress of your jobs
```

**Tip:** If you already have a *csv or excel file* containing paths, you can use the `-file` option on the *add-study-folders* command to add them all in one command

## Detailed example

In this example we will anonymize three studies that are on a share `\\server1\share`. The data folder looks like this:

```
\\server1\share\data
|--patient1
|  |--raw
|  |  |--raw1.dcm
|  |  |--raw2.dcm
|  |--study1      <- this should become 'anon1'
|  |--file1
|  |--file2
|--patient2
|  |--raw
|  |  |--raw1.dcm
|  |  |--raw2.dcm
|  |--study1      <- this should become 'anon2'
|  |--file1
|  |--file2
|  |--notes.txt
|--patient3
|  |--study1
|  |  |--file1
|  |  |--file2
|  |--study1_fixed <- this should become 'anon3'
|  |--file1
|  |--file2
```

For each patient, we want to to anonymize the data from the *study* folder. Except for *patient3*, where we want to get the data from the *study1\_fixed* folder. To do this take the following steps:

```
$ cd \\server\share\data    # Or use a drive letter or mount. Will be made UNC later
$ anon map init             # create a mapping at the source of the data
$ anon map edit             # opens mapping for editing
```

The mapping needs to be edited in two ways:

- the *root\_source\_path* parameter needs to be changed into a *UNC path* for the anonymization server to be able to find the data.

---

**Tip:** To find out the UNC path for a windows drive letter or a linux mount, see [Finding a UNC path](#)

---

- initially the mapping contains several rows with example data. These can be removed
- The *destination\_path* parameter will probably need to be changed

After making these changes, the mapping file should look like this:

```
## Description ##
Mapping created February 12 2020

## Options ##
root_source_path  \\server\share\data          <= changed
project,          Wetenschap-Algemeen
destination_path, \\server\share\myoutput      <= changed

## Mapping ##
source,          patient_id, patient_name, description
```

Now we will add each of the studies we want to anonymize. Make sure you close the editor before doing this:

```
$ anon map add-study-folders patient1/study patient2/study patient3/study_fixed
```

All DICOM files in these folders have now been selected and added as rows in the mapping. Now edit the rows to suit your needs, setting the patient ID and name you want.

```
$ anon map edit          # edit patientID, name etc. Save
$ anon create from-mapping # create anonymization jobs
```

## 5.4 Tutorials

Step-by-step instructions on how to perform particular tasks with anonapi.

---

**Note:** These instructions use windows paths and screenshots, but all commands work the same way for linux. In addition, *input files* are referred to as excel files, but csv format is also accepted

---

### 5.4.1 Anonymize from PACS

How to anonymize data when you have a list of accession numbers

#### Requirements

For this tutorial you need the following:

- C:\project\_folder - A folder that will hold records of the jobs created. Anonymized data does not have to go into this folder.
- C:\project\_folder\accession\_numbers.xlsx - An excel or csv *input file* containing a list of accession numbers that you want to anonymize. This file could be anywhere, but for convenience it is inside the project folder in this tutorial.

The path and file above are just examples. They can be anything you like.

#### Step 1: prepare input file

To start, open `accession_numbers.xlsx` and check the following:

- The file *must* contain a column with header `accession_number`
- The file *may* contain a column with header `pseudonym`. These pseudonyms will be used if found.
- The file *may* contain any other columns and text. These will all be ignored for example, The following is a valid input file:

Some text here does not matter  
The column 'project' below will also just be ignored

accession_number	pseudonym	project
1234567.12345678	studyA	some_projectA
2234567.12345678	studyB	some_projectB
3234567.12345678	studyC	some_projectB

#### Step 2: add input to mapping

- Find your *project folder* C:\project\_folder in windows explorer, shift + right-click->Open command prompt (see *opening a command prompt*)
- In the command line, run the following:

```
$ anon map init
$ anon map add-accession-numbers --input-file accession_numbers.xlsx
```

- Now open the mapping for editing:

```
anon map edit
```

**Tip:** If values are not sorted into columns properly in excel, do the following:

- select column A -> click menu Data -> click Text to Columns
- In the menu choose delimited -> click Next

- Under ‘delimiters’ check either Comma or Semicolon -> click Next

make the following edits (as shown in the image below):

- project should have the correct anonymization project (usually ‘Wetenschap-Algemeen’)
- destination\_path should be an be a *UNC path* (like `\\server\share`) where you want your anonymized data to be written.

	A	B	C	D	E	F	G
1	## Description ##						
2	Mapping created november 05 2020 by 2428172						
3	## Options ##						
4	root_source_path	C:\project_folder					
5	project	NOT SET					
6	destination_path	.					
7							
8	## Mapping ##						
9	source	pseudo_name	description				

### Step 3: create jobs

Make sure you have a *command prompt* in your *project folder* `C:\project_folder`. Then run the following:

```
$ anon create from-mapping
```

This will create all jobs defined in your mapping file and save a reference to them as a *batch*.

### Step 4: monitor job batch

To see how your jobs are doing use

```
$ anon batch status
```

To print error messages for all failed jobs use

```
$ anon batch show-error
```

Other batch commands can be found [here](#).

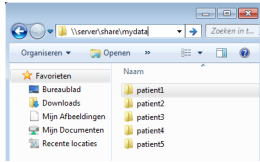
## 5.4.2 Anonymize from a share

How to anonymize data when you have data for a large number of patients on a share and want to anonymize several, but not all of these.

### Requirements

For this tutorial you need the following:

- `C:\project_folder` - A folder that will hold records of the jobs created. Anonymized data does not have to go into this folder.
- `\\server\share\mydata` - The data to anonymize. This share should contain a folder for each patient or each study that you wish to anonymize. The folders *may* contain subfolders and/or non-DICOM files. Files in subfolders will be included. Non-DICOM files will be ignored. For this tutorial, we will assume the following simple structure with 5 patients:



- C:\project\_folder\folders.xlsx - An excel or csv *input file* containing a list of folder names that you want to anonymize. This file could be anywhere, but for convenience it is inside the project folder in this tutorial.

The paths above are just examples. They can be anything you like.

## Step 1: prepare input file

To start, open the *input file* folders.xlsx and check the following:

- The file *must* contain a column with header `folder`, containing the names of the folders you want to anonymize from \\server\share\mydata
- The file *may* contain a column with header `pseudonym`. These pseudonyms will be used if found.

For example:

folder	pseudonym
patient1	anon_patientA
patient3	anon_patientB
patient5	anon_patientC

## Step 2: add input to mapping

- Find your *project folder* C:\project\_folder in windows explorer, shift + right-click-> Open command prompt (see *opening a command prompt*)
- In the command prompt, type `anon map init` and press enter. Then close the prompt again.
- Now find \\server\share\mydata in windows explorer and shift + right-click -> Open command prompt there.

**Note:** windows command prompt will show this location as a mapped drive letter like H:\ or X:\. This is not a problem here. When setting source and destination locations later on in this tutorial, be sure to always use the \\server\share form, never the drive letter.

- In the command prompt, type the following:

```
$ anon map add-study-folders --input-file "C:\project_folder\folders.xlsx"
```

This command might take some time to complete as it will scan for all DICOM files in each of the folders.

- Now open the mapping for editing:

```
anon map edit
```

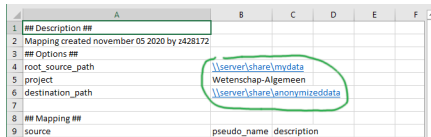
**Tip:** If values are not sorted into columns properly in excel do the following:

- select column A -> click menu Data -> click Text to Columns
- In the menu choose delimited -> click Next

- Under ‘delimiters’ check either Comma or Semicolon -> click Next
- 

make the following edits (as shown in the image below):

- `root_source_path` should be `\\server\share\mydata`, the root folder that contains each of your patient or study folders
- `project` should have the correct anonymization project (usually ‘Wetenschap-Algemeen’)
- `destination_path` should be an be a *UNC path* (like `\\server\share`) where you want your anonymized data to be written.



	A	B	C	D	E	F
1	## Description ##					
2	Mapping created november 05 2020 by z428172					
3	## Options ##					
4	root_source_path	\\server\share\mydata				
5	project	Wetenschap-Algemeen				
6	destination_path	\\server\share\anonymizeddata				
7						
8	## Mapping ##					
9	source	pseudo_name	description			

### Step 3: create jobs

Make sure you have a *command prompt* in your *project folder* `C:\project_folder`. Then run the following:

```
$ anon create from-mapping
```

This will create all jobs defined in your mapping file and save a reference to them as a *batch*.

### Step 4: monitor job batch

To see how your jobs are doing use

```
$ anon batch status
```

To print error messages for all failed jobs use

```
$ anon batch show-error
```

Other batch commands can be found [here](#).

## 5.5 Command reference

Information on specific anonapi CLI functions. For more how-to's on achieving specific goals, see [Usage](#)

---

**Tip:** Detailed help on commands is often available directly from the command prompt by adding `--help` to any command. See [Getting info on commands](#)

---



### 5.5.1 status

Display information on the command line tool itself. Which API servers it knows about, current active server

### 5.5.2 server

Work with Anonymization server API servers. Add, remove servers, set active server

Overview of server functions:

Command	Description
activate	Commands will use given server by default
add	Add a server to the list of servers in settings
jobs	List latest 100 jobs for active server
list	Show all servers in settings
remove	Remove a server from list in settings
status	Check whether active server is online and responding

### 5.5.3 job

Work with single jobs. Get extended info, reset, restart a job

Overview of job functions:

Command	Description
cancel	Set job status to inactive
info	Print full info for one or more jobs
list	List info for multiple jobs
reset	Reset job, process again
set-opt-out-ignore	Set opt-out ignore with given reason for job_id

### 5.5.4 settings

View and edit local settings for this anonapi instance. Credentials that are used to communicate with the API. See *Configure credentials*.

Command	Description
get-token	Obtain a security token
info	Show current credentials
set-username	Set the given username in settings

Settings are stored in the users home directory in a file called *AnonWebAPIClientSettings.yml*

### 5.5.5 batch

Work with lists of jobs on a certain server. Anonymization jobs often occur in sets. With batches you can group jobs together and do batch processing on them. A batch lives in a single folder. To work with a batch you have to be in that folder. For example:

```
$ cd /tmp/my_folder
$ anon batch info # Will not work because there is no batch defined in this folder yet
> Error: No batch defined in current folder

$ anon batch init # Create empty batch for the currently active server
> Initialised batch for p01: https://apiservers/p01 in current dir

$ anon batch info # Now there is an empty list
> job_ids: []
  server:
    name: p01
    url: https://apiservers/p01

$ anon batch add 1 2 3 # Now you can add job ids, separated by spaces
$ anon batch info # Now there is an empty list
> job_ids: ['1', '2', '3']
  server:
    name: p01
    url: https://apiservers/p01

$ anon batch status # Now you can print status for all ids in this batch
> Job info for 3 jobs on p01: https://umcradanonp11.umcn.nl/p01:
  id  date                status    down    proc    user
  ----
  1    2016-08-26T15:04:44  INACTIVE  0        0      z123456
  2    2016-08-26T15:04:44  ERROR     503      100     z123456
  3    2016-08-26T15:04:44  DONE      1155     1155    z123456
```

batch command overview:

Command	Description
add	Add ids to current batch. Space-separated (1 2 3) or range (1-40)
cancel	Cancel every job in the current batch
cancel-active	Cancel unprocessed (active) jobs, leave done and error
delete	Delete batch in current folder
info	Show batch in current directory
init	Save an empty batch in the current folder, for current server
remove	Remove ids from current batch. Space-separated (1 2 3) or range (1-40)
reset	Reset every job in the current batch
reset-error	Reset all jobs with error status in the current batch
show-error	Show full error message for all error jobs in batch
status	Print status overview for all jobs in batch

For convenience, it is possible to pass job ids for batch add and batch remove as ranges:

```
$ anon batch add 5-12 # Add range
$ anon batch info # ranges include both start and end number
```

(continues on next page)

(continued from previous page)

```
> job_ids: ['5', '6', '7', '8', '9', '10', '11', '12']
  server:
    name: p01
    url: https://apiservers/p01

$ anon batch remove 8-11 # Remove range
$ anon batch info # ranges include both start and end number
> job_ids: ['5', '6', '7', '12']
  server:
    name: p01
    url: https://apiservers/p01
```

## status

Show a table with status for all jobs in the batch in current directory.

```
$ anon batch status # Now you can print status for all ids in this batch
> Job info for 3 jobs on p01:
```

id	date	status	down	proc	user
1	2016-08-26T15:04:44	INACTIVE	0	0	z123456
2	2016-08-26T15:04:44	ERROR	503	100	z123456
3	2016-08-26T15:04:44	DONE	1155	1155	z123456

Modifiers:

### --patient-name

With this modifier a column *anon\_name* is added, which shows the anonymized name used in this job:

```
$ anon batch status --patient-name
> Job info for 3 jobs on p01: https://umcradanonp11.umcn.nl/
p01:
```

id	date	status	down	proc	user	anon_name
1	2016-08-26T15:04:44	INACTIVE	0	0		
z123456	patient34					
2	2016-08-26T15:04:44	ERROR	503	100		
z123456	patient40					
3	2016-08-26T15:04:44	DONE	1155	1155		
z123456	patient41					

### 5.5.6 map

Create a mapping between data and anonymization parameters. This mapping contains everything needed to create anonymization jobs

Overview of map functions:

Command	Description
activate	All subsequent mapping actions will target this folder
add-accession-numbers	Add accession numbers to an existing mapping
add-selection	Add selection file to mapping
add-study-folders	Add all dicom files in given folders to mapping
delete	Delete current active mapping
edit	Edit the active mapping in OS default editor
init	Save a default mapping in a default location in the current
status	Show mapping in current directory

#### add-study-folders

Add the given folders to *mapping*. This is done by finding all dicom files in the folder and any folders below it, adding those to a *file selection*, and then adding the file selection to the mapping. You can add multiple folders at once by using an *Input file*.

Options:

**-f, --input-file** add all study folders in this xlsx or csv file to mapping. Looks for column 'folder' in file. If a column 'pseudoID' is present, adds these instead of auto-generating

**-check-dicom/ --no-check-dicom** -check-dicom: Open each file to check whether it is valid DICOM. --no-check-dicom: Add all files that look like DICOM (exclude files with known file extensions like .txt or .xml). on by default

Example:

```
$ anon map add-study-folders folder1/
> Adding 'folder1' to mapping
> Finding all files in folder1
> 1it [12:01, 145.41it/s]
> Found 1512 files. Finding out which ones are DICOM
> 100%|| 1420/1512 [00:00<00:00, 10.51it/s]
> Found 1420 DICOM files
```

To find out which files are DICOM, each file is opened as DICOM. If this succeeds the file is added. This makes sure that only valid DICOM is sent to the anonymization server.

Running the command `anon map add-study-folders <folder>` is equivalent to running `anon select add <folder>` and then `anon map add-selection-file <folder>/fileselection.txt`

## Wildcards

Folder paths can contain asterisk `*` characters as wildcards. For example:

Command	matches_header paths (examples)
<code>add-study-folders folder*</code>	folderA, folderB
<code>add-study-folders folder*1</code>	folderA1, folderB1
<code>add-study-folders *</code>	<all folders>
<code>add-study-folders folder* extra</code>	folderA, folderB, extra
<code>add-study-folders **/raw</code>	A/B/raw, raw, C/raw

**Note:** For folders with many files, on a slow shared folder, `add-study-folders --check-dicom` might take several minutes to complete.

## add-accession-numbers

Add the given accession numbers to the active *mapping*. You can add multiple accession numbers at once by passing multiple values separated by spaces:

```
$ anon map add-accession-numbers 1234567.12345673 1111112.12345673 2222222.12345673
```

Alternatively you can use the `--input-file` flag with an *Input file* to add multiple accession numbers and pseudonyms at once. For example:

```
$ anon map add-accession-numbers --input-file myfile.csv
```

## add-selection-file

Add the given *file selection* file to *mapping*. This will create a new row in the mapping

## edit

Open the *mapping* file in current dir in the default editor for csv files. On windows this is usually excel.

**Warning:** Always close the editor before running anon commands that modify the mapping like *add-selection-file*. Many editors lock the file while open, making it impossible to change it by other means.

Some editors will ask you whether you want to save the mapping file in their own file format like `xlsx`. Never do this as this will make the mapping unreadable for anonapi.

## init

Create a *mapping* in the current folder containing some default content. *destination\_path* and *project* are based on the defaults set with the *create set-defaults* command

### 5.5.7 select

select files for a single anonymization job. The selection is saved in a *file selection* file.

Overview of select functions:

Command	Description
add	Add all files matching pattern to selection in the current directory.
delete	Remove selection file in current directory
edit	Open selection file in default editor
status	Show selection in current directory

## add

Add all files matching pattern paths to a *file selection* in the current folder. Pattern can use \* to match any part of a name. Excludes files called *fileselection.txt*

There are several modifiers available:

**--recurse/ --no-recurse** Search for files to add in subfolders as well. On by default

**--check-dicom/ --no-check-dicom** Only add files that are valid DICOM file. For many files, this might take some time. On by default.

**--exclude-pattern, -e** Exclude any file matching the given pattern. The pattern can use \* to match any part of a name. --exclude-pattern can be used multiple times, to exclude multiple patterns

Examples of different selections. Given the following folder structure:

```
patient1
|--study1
|   |--file1.dcm          (valid DICOM file)
|   |--bigfile.raw        (valid DICOM file)
|--study2
|   |--123.1224.5354.543.4 (valid DICOM file)
|   |--123.1224.2534.34.2 (valid DICOM file)
|--fileselection.txt
|--screenshots
|   |--shot1.jpg
```

You can select files like this:

```
$ anon select add *          # adds all files in the folder except 'fileselection.
↪txt'
$ anon select add --check-dicom * # adds both files in study1 and both in study2
$ anon select add study2/*      # adds both files in study2
$ anon select add *.dcm        # adds only study1/file1.dcm
```

(continues on next page)

(continued from previous page)

```
$ anon select add * --exclude-pattern *.raw # all DICOM except study1/bigfile.raw
$ anon select add * --exclude-pattern *.raw --exclude-pattern *.dcm # only files in_
↪ study2
```

## 5.5.8 create

create jobs on server

Overview of create functions:

Command	Description
from-mapping	Create jobs from mapping in current folder
set-defaults	Set project name used when creating jobs
show-defaults	Show project name used when creating jobs

### from-mapping

Create a job for each row in the *Mapping* in the current directory. This will do some validation and ask for confirmation:

```
$ anon create from-mapping
> This will create 3 jobs on p01, for projects '['Wetenschap-Algemeen']',
> writing data to '['\\\\server\\share\\folder']'. Are you sure? [y/N]:
$ Y
> Created job with id 1
> Created job with id 2
> Created job with id 3
> created 3 jobs: [1, 2, 3]
> Saving job ids in batch in current folder
> Done
```

The command will create a *Batch* in the current folder containing each created job. This means you can use all *batch* commands on your created jobs:

```
$ anon batch info
> job_ids:
> - '1'
> - '2'
> - '3'
> server:
>   name: p01
>   url: https://anonserver_p01/api
```

## 5.6 Concepts

Information on some key concepts in the anonapi CLI

### 5.6.1 Batch

A file holding one or more job ids. This makes it possible to easily query or modify several jobs at once. See the [batch command](#).

### 5.6.2 Mapping

A file that contains everything that is needed to create one or more anonymization jobs.

A typical mapping file will look like this:

```
## Description ##
Mapping created February 12 2020

## Options ##
root_source_path, \\server\share2\data
project,          Wetenschap-Algemeen
destination_path, \\server\share\folder

## Mapping ##
source,           patient_id, patient_name, description
folder:example/folder1, 001,      Patient1,      All files from folder1
study_instance_uid:123.12178, 002,      Patient2,      A StudyInstanceUID from PACS
accession_number:12345678.1234567, 003,      Patient3,      An AccessionNumber from PACS
fileselection:a/fileselection.txt, 004,      Patient4,      A selection of files in a
```

This is a CSV (comma separated values) file that can be edited by any editor. The most convenient way to edit is probably the [edit](#) command.

A mapping consists of three sections:

**Description** This can contain any text. A description of what this mapping is for

**Options** Parameters that are the same for each job. The following parameters can be set:

Parameter	Description
destination_path	Write data to this UNC path after anonymization
pims_key	Use this PIMS project to pseudonymize
project	Anonymize according to this project
root_source_path	Path sources are all relative to this UNC path

---

**Note:** Any paths defined in this section have to be *UNC paths*. No windows drive letters like H:\ or linux mounts such as /mnt/data allowed

---

**Mapping** Parameters that are different for each job. The following parameters can be set:



Parameter	Description
description	Job description, free text
pseudo_id	Pseudonym for Patient ID to set in anonymized data
pseudo_name	Pseudonym for Patient name to set in anonymized data
source	Data to anonymize comes from this source

The value of the *source* parameter is a *source identifier*. The different types of identifiers are listed below.

For an overview of map functions, see *map*.

### 5.6.3 Input file

A csv or excel file that contains one or more columns with folders, pseudonyms or accession numbers. A file like this can be used as an input for *map* functions such as *add-study-folders* to add multiple values at once.

Example input file containing folders and pseudonyms:

folder	pseudonym
folder1	studyA
folder2/st1	studyB
folder2/st2	studyC

The column headers ('folder' and 'pseudonym' above) are used to identify type of data and to find where the columns are in the file. The following column types are currently supported:

Parameter	Allowed column names
accession_number	accession number, acc nr
path	folder, map, path
pseudo_name	pseudoID, pseudonym, name

Finding column headers ignores case and space characters. For example, the following are all valid column headers for accession number: *accession number*, *Accession Number*, *accession\_number*, *accession-number*, *AccessionNumber*

Information that is not recognized as valid is ignored. For example, the following input file is valid and contains the same information as the example given above:

Some descriptive text that will just be ignored when parsing this as an input file.			
Columns with headers that are not recognized are ignored as well. Below, 'folder' and 'pseudonym' will be recognized, others ignored			
folder	value	pseudonym	comment
folder1	A	studyA	
folder2/st1	A	studyB	this column
folder2/st2	B	studyC	will be ignored

### 5.6.4 Source Identifier

Used in *mapping* to indicate where the data for a job is coming from. Always of the form `<identifier_type>:<value>`. Types of identifiers:

**Folder** Example: `folder:mydata/experiment1`

Refers to all files in the given folder, relative to the source root path.

---

**Note:** If the folder contains any files that are not valid DICOM, the job will fail. Only use this identifier if you want to anonymize all files in a folder, and the folder contains only valid DICOM

---

**File selection** Example: `fileselection:mydata/patient1/fileselection.txt`

Refers to all the paths listed in the *fileselection file*. Contrary to the Folder identifier, file selection can be used in a folder where there are non-DICOM files or where only part of the files should be anonymized. When creating a fileselection with *add-study-folders* or *add*, non-DICOM files can be excluded automatically

**Study instance UID** Example: `study_instance_uid:123.1217.23234.2323`

Refers to a single study. The anonymization server will retrieve this study from PACS by matching the DICOM tag StudyInstanceUID.

**Accession number** Example: `accession_number:12345678.1234567`

Refers to a single study. The anonymization server will retrieve this study from PACS by matching the DICOM tag AccessionNumber.

### 5.6.5 Job

The basic unit of information on an anonymization server. A job specifies three things. Where the data is, how to anonymize it and where it should go. For working with jobs see *job*.

### 5.6.6 File Selection

A file typically called `fileselection.txt` that contains a list of paths. A selection can be a data source for a job. It makes it possible to specify which files should be sent for anonymization and which should not. Methods like *add-study-folders* and *add* only include valid DICOM files in a selection.

The contents of a typical file selection that contains 4 file paths:

```
description: a typical file selection
id: bfc33f5e-d1cc-472e-aa05-31a5979d52be
selected_paths:
- folder1/1.dcm
- folder1/2.dcm
- folder2/1.dcm
- folder4/raw/raw1.dcm
```

A selection file can be edited by any text editor. See *select*.

---

**Note:** Selected paths are always relative to the location of `fileselection.txt`. Selected paths are always in a path on or below the selection file.

---

### 5.6.7 Server

An anonymization server fetches, anonymizes and delivers your data according to the *jobs* it has in its database. Servers can retrieve data from PACS or from network shares. The anonapi CLI can work with multiple servers. See *Server commands*.

### 5.6.8 UNC paths

Any path sent to the anonymization server should be a UNC path. A UNC path is any path starting with:

```
\\<server_name>\<share_name>
```

For example:

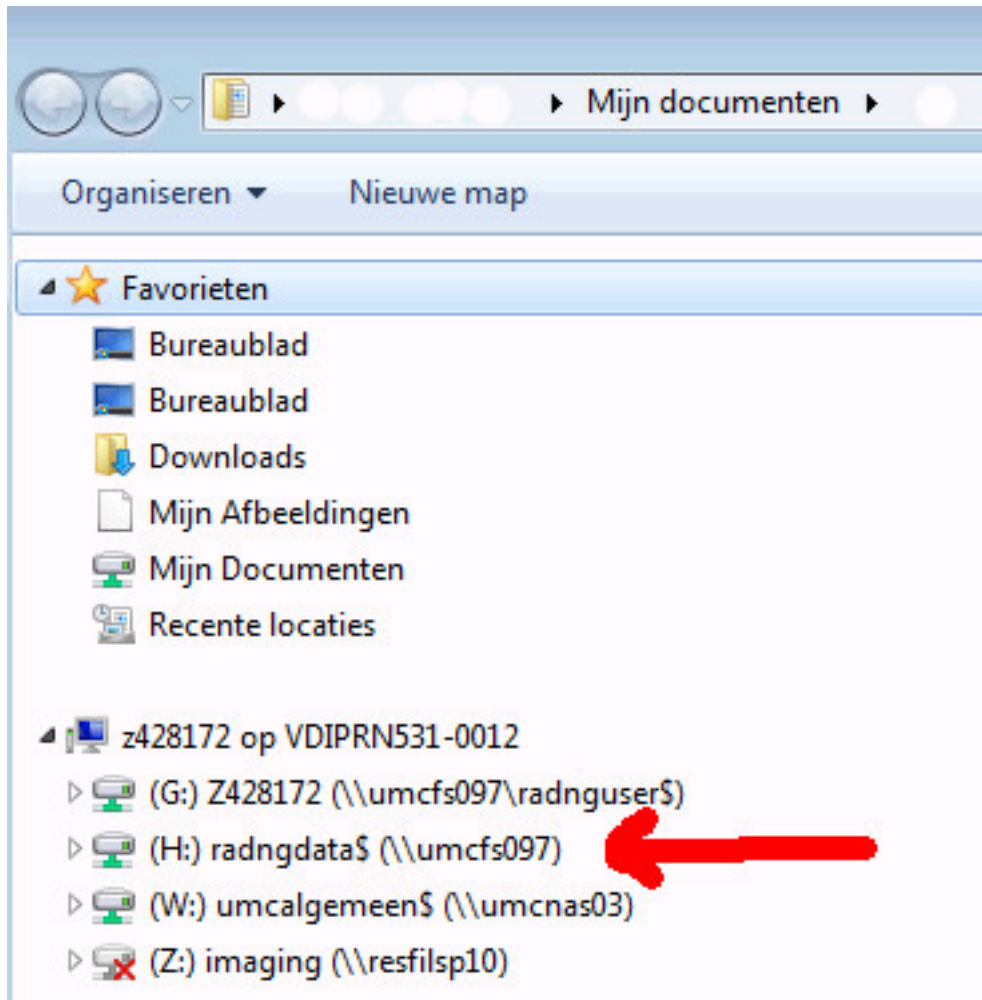
```
\\umcfiles01\research\folder1\file.dcm  
\\server1\share2\myfolder\
```

UNC paths are mandatory for creating *anonymization jobs* because they are well supported in most operating systems and unambiguous. In contrast, windows drive letters such as C:\, mapped network drives such as X:\ and linux mounts like /mnt/share1 can refer to different locations on different computers.

You can find more [unc\\_path\\_info](#) online.

#### Finding a UNC path

**Windows** In windows shares are often *mapped* to a drive letter such as H:\ or X:\. To find the UNC path for these drive letters, open windows explorer (start menu -> explorer) and expand the computer icon in the lower left side:



In this example (H:) radngdata\$ (\\umcfs097) corresponds to the UNC path \\umcfs097\radngdata\$ note the path in this case includes the final \$

**Linux** In linux UNC paths are mounted in fstab. Use:

```
$ less /etc/fstab
```

To find out which UNC path is mapped to which mount point.

## 5.7 Troubleshooting

Examples of commonly encountered errors when using the anonymization api CLI

### 5.7.1 Common anonapi errors

#### Error Max retries exceeded

When getting information from server, something like this

```
Error getting jobs from p01: https://umcradanonp11.umcn.nl/p01:
HTTPSConnectionPool(host='umcradanonp11.umcn.nl', port=443): Max retries exceeded
with url: /p01/get_jobs (Caused by NewConnectionError('<urllib3.connection.
VerifiedHTTPSConnection object at 0x7f5580875198>: Failed to establish a
new connection: [Errno 111] Connection refused'))
```

This means the anonymization api server is not responding. Check the end of the message. If it says 'Connection refused', the server is online but not responding. Inform a server admin.

If it says 'Name or service not known', the url you have entered for the server might be wrong. Recheck [Add a server to the CLI](#)

#### Unexpected response code 500

Something like

```
Error getting job info from t01: https://umcradanonp11.umcn.nl/t01:
Unexpected response from WebAPIClient for z428172@https://umcradanonp11.umcn.nl/t01:
code '500', reason 'Internal Server Error'
```

There is something wrong with the job itself. Inform a server admin.

### 5.7.2 Common job errors

Error messages you might encounter when getting [Information about jobs](#)

#### Job stuck on status UPLOADED

Example

```
job 50881 on t01:

('job_id', 50881)
('date', '2019-04-29T12:13:55')
('user_name', 'z123456')
('status', 'UPLOADED')          <== Status does not change
('error', None)                 <== No error message
('description', 'Ultrasound test')
('project_name', 'Wetenschap-Algemeen')
('priority', 30)
```

(continues on next page)

(continued from previous page)

```

('files_downloaded', 1683)
('files_processed', 1600)          <== Not all downloaded files have been processed

```

Stuck on **UPLOADED** usually means that part or all of the collected data is being quarantined. This regularly happens with data with burned in annotations such as ultrasound. The system will have to be explicitly shown which portions of images contain patient information.

Solution: record job id and send to anonapi admin

## HTTP 404 not found

Example

```

job 52019 on p03:

('job_id', 112323)
('date', '2019-06-14T12:14:04')
('user_name', 'z123456')
('status', 'ERROR')
('error', "Error while pre-processing job 52019: WadoWrapperException: Got HTTP error_
↳response
    from server when requesting 'http://umcidcsasp04:8080/wado/?
↳studyInstanceUID=1234
    &contentType=application/dicom&requestType=WADO&transferSyntax=1.2.840.10008.
↳1.2.1'
    Original error:'HTTP Error 404: Not Found'")
('description', Test)
('project_name', 'Wetenschap-Algemeen')

```

This often means data could not be retrieved from the hospital image server. This might be due to a temporary glitch in that server.

Solution: retry job after at least 30 minutes. See *Cancel or restart jobs*

## HTTP 400 server error

Example

```

job 52863 on p03:

('job_id', 12345)
('date', '2019-07-22T10:46:48')
('user_name', 'Z123456')
('status', 'ERROR')
('error', "Error while pre-processing job 52863: WadoWrapperException: Got HTTP error_
↳response
    from server when requesting 'http://umcidcsasp04:8080/wado/?
↳studyInstanceUID=1234"
    "&contentType=application/dicom&requestType=WADO&transferSyntax=1.2.840.10008.
↳1.2.1'"
    "Original error:'HTTP Error 400: Bad Request'")

```

(continues on next page)

(continued from previous page)

```
('description', A test project)
('project_name', 'Wetenschap-Algemeen')
```

The end of the error message is important. There was an internal error in the hospital image server when retrieving the data for this job. This sometimes happens for incorrectly imported data or additional findings that have been incorrectly pushed to the images server.

Solution: First retry after a few hours, a day, possibly days. These errors sometimes solve themselves. If this does not work, Record the accession number for the data in this job and ask beeld en zorg to check for errors in that data.

### Could not move file

Example

```
job 52132 on p03:

('job_id', 52132)
('date', '2019-06-14T12:14:12')
('user_name', 'z1234567')
('status', 'ERROR')
('error', 'Could not move file D:\\CTP\\roots\\Post-anonimizationDSS\\52132\\test\\'
          'to \\\\umcsanfsclp01\\radng_trialbureau\\52132\\test\\,'
          'original error:[Errno 28] No space left on device')
('description', None)
('project_name', 'Wetenschap-Algemeen')
```

The share that the data is going to is full. Make sure there is enough space and retry

### Patient has opted out

Example

```
job 51815 on p03:

('job_id', 51815)
('date', '2019-06-14T12:13:49')
('user_name', 'z123456')
('status', 'ERROR')
('error', 'Error while pre-processing job 51815: Patient "123456" does not want'
          'his or her data to be used for research')
('description', None)
('project_name', 'Wetenschap-Algemeen')
```

The patient associated with this data has indicated he or she does not want his or her data to be used for research purposes. You cannot use this data. If the patient has signed a specific declaration of consent for your specific research, contact the trial bureau.

### 5.7.3 Job status codes

These are part of the info you get when getting *Information about jobs*. For example

```
job 51815 on p03:

('job_id', 51815)
('date', '2019-06-14T12:13:49')
('user_name', 'z123456')
('status', 'DONE')      <== that one
```

Job status codes and their meaning:

**ACTIVE** This job is waiting to be processed by the server

**UPLOADED** Uploaded to anonymization core. If a job has this status for longer than 30 minutes, refer to *Job stuck on status UPLOADED*

**DONE** Data has been processed and copied to final destination. Some quarantined files such as embedded pdfs might still be held back.

**ERROR** Something went wrong. Refer to *Common job errors* for more information.

## 5.8 Python Usage

How to use anonapi from within a python script

`anonapi.client.WebAPIClient` is the main class to use when interacting with an IDIS server web API from python. To use it in a python file:

```
from anonapi.client import WebAPIClient

client = WebAPIClient(
    hostname="https://umcradanonp11.umcn.nl/sandbox",
    username="z123sandbox",
    token="token",
)

# Get some information on first few jobs
jobs_info = client.get("get_jobs")
```

### 5.8.1 Testing

The `anonapi.testresources` module can be used to generate mock responses without the need for a live server:

```
from anonapi.testresources import (
    MockAnonClientTool,
    JobInfoFactory,
    RemoteAnonServerFactory,
    JobStatus,
)

# Create a mock client tool that returns information about jobs
```

(continues on next page)



(continued from previous page)

```

tool = MockAnonClientTool()

# Client tool methods need a server to query. This is mocked too here
mock_server = RemoteAnonServerFactory()
tool.get_job_info(server=mock_server, job_id=1)    # Returns realistic JobInfo response

# You can set the responses that mock client cycles through:
some_responses = [
    JobInfoFactory(status=JobStatus.DONE),
    JobInfoFactory(status=JobStatus.ERROR),
    JobInfoFactory(status=JobStatus.INACTIVE),
]
tool = MockAnonClientTool(responses=some_responses)
tool.get_job_info(mock_server, job_id=1).status # returns JobStatus.DONE
tool.get_job_info(mock_server, job_id=1).status # returns JobStatus.ERROR
tool.get_job_info(mock_server, job_id=1).status # returns JobStatus.INACTIVE
tool.get_job_info(mock_server, job_id=1).status # returns JobStatus.DONE again

# You can set any of the JobInfo fields on the JobInfo instances:
JobInfoFactory(project_name='project1',
                destination_path='\\server\\share\\folder',
                priority=50)

```

## 5.8.2 Exceptions

All exceptions raised in anonapi derive from `anonapi.exceptions.AnonAPIError`. Catching that will allow you to handle them:

```

from anonapi.exceptions import AnonAPIError
from anonapi.client import AnonClientTool

tool = AnonClientTool('user', 'token')
server = RemoveAnonServer('a_server', 'https://aserver')
try:
    tool.get_server_status(server)

except AnonAPIError as e:
    print(f"Something went wrong but its anonapi's fault. Look: {e}")

```

### 5.8.3 Examples

These examples are taken from the code in the *Examples* directory

#### Anonymize from IDC

IDC is the hospital medical image server

```
from anonapi.client import WebAPIClient

def anonymize_files_from_idc():
    """Create an IDIS job that pulls files from the hospital information system"""

    # Create a client that will talk to the web API
    client = WebAPIClient(
        hostname="https://umcradanonp11.umcn.nl/sandbox",
        username="z123sandbox",
        token="token",
    )

    # Create a job that takes data from the IDC (formally IMPAX) directly
    anon_name = "TEST_NAME_02"
    anon_id = "02"
    sid = "123.12335.3353.36464.343435677" # study UID
    destination_path = r"\\umcsanfscpl01\radng_imaging\temptest_output"
    idc_job_info = client.post(
        "create_job",
        source_type="WADO",
        source_name="IDC_WADO",
        source_instance_id=sid,
        anonymizedpatientname=anon_name,
        anonymizedpatientid=anon_id,
        destination_type="PATH",
        project_name="Wetenschap-Algemeen",
        destination_path=destination_path,
        description="A test idc job",
    )
    print(
        f"Successfully created a job in {client}, job_id={idc_job_info['job_id']}"
    )

if __name__ == "__main__":
    anonymize_files_from_idc()
```

## Anonymize from network share

```

from anonapi.client import WebAPIClient

def anonymize_files_from_share():
    """Create an IDIS job that pulls files from a network share"""

    # Create a client that will talk to the web API
    client = WebAPIClient(
        hostname="https://umcradanonp11.umcn.nl/sandbox",
        username="z123sandbox",
        token="token",
    )

    # Create a job that takes data from a source on a network root_path, and writes
    # the anonymized data to a destination that is also a network root_path. Note
    # that the network root_path should be accessible for the anonymization server
    # for this to work.
    anon_name = "TEST_NAME_01"
    anon_id = "01"
    source_path = r"\\umcsanfscpl01\radng_imaging\temp\test"
    destination_path = r"\\umcsanfscpl01\radng_imaging\temptest_output"
    network_job_info = client.post(
        "create_job",
        source_type="PATH",
        source_path=source_path,
        destination_type="PATH",
        project_name="Wetenschap-Algemeen",
        destination_path=destination_path,
        anonymizedpatientname=anon_name,
        anonymizedpatientid=anon_id,
        description="A test root_path job",
    )
    # new_job_info response contains extended info on the new job that has
    # just been created
    print(
        f"Successfully created a job in {client}, job_id={network_job_info['job_id']}"
    )

if __name__ == "__main__":
    anonymize_files_from_share()

```

## Filter on SOPClass

SOPClassUID is the DICOM 'image type'

```
from anonapi.client import WebAPIClient

def anonymize_files_sop_class_filter():
    """Create an IDIS job that pulls files from the hospital information system"""

    # Create a client that will talk to the web API
    client = WebAPIClient(
        hostname="https://umcradanonp11.umcn.nl/sandbox",
        username="z123sandbox",
        token="token",
    )

    # Create a job that takes data from the IDC (formally IMPAX) directly
    # and allow only files that match the given SOPClassUIDs. For a full list of
    # possible SOPClassUIDs see https://www.dicomlibrary.com/dicom/sop/
    anon_name = "TEST_NAME_03"
    anon_id = "03"
    sid = "123.12335.3353.36464.343435677" # study UID
    destination_path = r"\\umcsanfsclp01\radng_imaging\temptest_output"
    idc_job_info = client.post(
        "create_job",
        source_type="WADO",
        source_name="IDC_WADO",
        source_instance_id=sid,
        source_sop_class_filter_list="1.2.840.10008.5.1.4.1.1.88.67, "
        "1.2.840.10008.5.1.4.1.1.7",
        anonymizedpatientname=anon_name,
        anonymizedpatientid=anon_id,
        destination_type="PATH",
        project_name="Wetenschap-Algemeen",
        destination_path=destination_path,
        description="A test idc job",
    )
    print(
        f"Successfully created a job in {client}, job_id={idc_job_info['job_id']}"
    )

if __name__ == "__main__":
    anonymize_files_sop_class_filter()
```

## Cancel job

```
from anonapi.client import WebAPIClient

def cancel_job():
    # Create a client that will talk to the web API
    client = WebAPIClient(
        hostname="https://umcradanonp11.umcn.nl/sandbox",
        username="z123sandbox",
        token="token",
    )

    # cancel job 100
    client.post("cancel_job", job_id=100)

if __name__ == "__main__":
    cancel_job()
```

## Get job status

```
from anonapi.client import WebAPIClient

def get_job_status():
    """Get information about a number of jobs"""

    # Create a client that will talk to the web API
    client = WebAPIClient(
        hostname="https://umcradanonp11.umcn.nl/p01",
        username="z123sandbox",
        token="token",
    )

    # get the status for 3 specific jobs
    job_status_list = client.get(
        "get_jobs_list_extended", job_ids=[53769, 53770, 53771]
    )
    print(f"found status for {len(job_status_list)} jobs in list:")
    print(job_status_list)

if __name__ == "__main__":
    get_job_status()
```

## Modify jobs

```
from anonapi.client import WebAPIClient

def modify_jobs():
    """Modify several jobs

    Notes
    ----
    For a full list of the job fields that can be modified see example
    'get_api_definition'
    """

    # Create a client that will talk to the web API
    client = WebAPIClient(
        hostname="https://umcradanonp11.umcn.nl/sandbox",
        username="z123sandbox",
        token="token",
    )

    job_ids = [1, 2, 3]
    for job_id in job_ids:
        client.post(
            "modify_job",
            job_id=job_id,
            source_path=r"\\umcsanfsclp01\radng_imaging\temp\modified\test",
        )

if __name__ == "__main__":
    modify_jobs()
```

## 5.9 Modules

### Table of Contents

- *Modules*
  - *anonapi package*
    - \* *anonapi.batch module*
    - \* *anonapi.client module*
    - \* *anonapi.context module*
    - \* *anonapi.decorators module*
    - \* *anonapi.exceptions module*
    - \* *anonapi.mapper module*
    - \* *anonapi.objects module*

```

* anonapi.parameters module
* anonapi.responses module
* anonapi.selection module
* anonapi.settings module
* anonapi.testresources module
* Module contents
* Examples

```

### 5.9.1 anonapi package

#### anonapi.batch module

Work with batches of jobs. Batches are modeled on git repos; state is maintained via hidden file in current folder.

**class** `anonapi.batch.BatchFolder(path)`

Bases: `object`

A folder in which a batch might be defined

**BATCH\_FILE\_NAME** = `'anonbatch'`

**\_\_init\_\_**(path)

**Parameters** `path` (*Pathlike*) – root\_path to this folder

**property** `batch_file_path`

**delete\_batch**()

Delete the batch file in this folder

**Raises** `BatchFolderError`: – if remove does not work for some reason

**has\_batch**()

**load**()

Load batch from the current folder

**Return type** *JobBatch*

**save**(batch)

Save the given batch to this folder

**Parameters** `batch` (*JobBatch*) – job batch to save in this folder

**exception** `anonapi.batch.BatchFolderError`

Bases: `anonapi.exceptions.AnonAPIError`

**class** `anonapi.batch.JobBatch(job_ids, server)`

Bases: `anonapi.persistence.YAMLSerializable`

A collection of anonymisation jobs

**\_\_init\_\_**(job\_ids, server)

**Parameters**

- **job\_ids** (*List(str)*) – All job ids in this batch

- **server** ([RemoteAnonServer](#)) – Server these jobs were created in

**classmethod** `from_dict(dict_in)`

Create object from dict. Basis for json serialization. Overwrite this in child classes to yield an instance of the child class

**Raises** **ValueError** – If an object cannot be created from dict\_in

**to\_dict()**

**Return type** str

**to\_string()**

Batch as input

**Returns** String with newlines representing this batch

**Return type** str

**exception** `anonapi.batch.NoBatchDefinedError`

Bases: [anonapi.exceptions.AnonAPIError](#)

**anonapi.client module**

**anonapi.context module**

**anonapi.decorators module**

**anonapi.exceptions module**

**exception** `anonapi.exceptions.AnonAPIError`

Bases: Exception

Base exception for everything raised in anonapi

**anonapi.mapper module**

**anonapi.objects module**

classes and methods shared by anonapi modules

**class** `anonapi.objects.RemoteAnonServer(name, url)`

Bases: object

An anonymization server that can be talked to via the API

**\_\_init\_\_**(name, url)

Create a Remote anon server entry

**Parameters**

- **name** (str) – short keyword to identify this server
- **url** (str) – full url to a valid Anonymization server web API

**classmethod** `from_dict(dict_in)`

Load instance from output of to\_dict

**Return type** [RemoteAnonServer](#)



**to\_dict()**

Dictionary representation of this server

**Return type** Dict

**anonapi.parameters module**

**anonapi.responses module**

**anonapi.selection module**

**anonapi.settings module**

**anonapi.testresources module**

### **Module contents**

Top-level package for AnonAPI.

### **Examples**

To see the source code for these examples, press the [source] link to the right of each title

## **5.10 Release Notes**

Details of each release are available on [gihub](#).



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### a

`anonapi`, [45](#)

`anonapi.batch`, [43](#)

`anonapi.exceptions`, [44](#)

`anonapi.objects`, [44](#)



## Symbols

`__init__()` (*anonapi.batch.BatchFolder* method), 43  
`__init__()` (*anonapi.batch.JobBatch* method), 43  
`__init__()` (*anonapi.objects.RemoteAnonServer* method), 44

## A

*anonapi*  
     *module*, 45  
*anonapi.batch*  
     *module*, 43  
*anonapi.exceptions*  
     *module*, 44  
*anonapi.objects*  
     *module*, 44  
*AnonAPIError*, 44

## B

*BATCH\_FILE\_NAME* (*anonapi.batch.BatchFolder* attribute), 43  
*batch\_file\_path* (*anonapi.batch.BatchFolder* property), 43  
*BatchFolder* (class in *anonapi.batch*), 43  
*BatchFolderError*, 43

## D

*delete\_batch()* (*anonapi.batch.BatchFolder* method), 43

## F

*from\_dict()* (*anonapi.batch.JobBatch* class method), 44  
*from\_dict()* (*anonapi.objects.RemoteAnonServer* class method), 44

## H

*has\_batch()* (*anonapi.batch.BatchFolder* method), 43

## J

*JobBatch* (class in *anonapi.batch*), 43

## L

*load()* (*anonapi.batch.BatchFolder* method), 43

## M

*module*  
     *anonapi*, 45  
     *anonapi.batch*, 43  
     *anonapi.exceptions*, 44  
     *anonapi.objects*, 44

## N

*NoBatchDefinedError*, 44

## R

*RemoteAnonServer* (class in *anonapi.objects*), 44

## S

*save()* (*anonapi.batch.BatchFolder* method), 43

## T

*to\_dict()* (*anonapi.batch.JobBatch* method), 44  
*to\_dict()* (*anonapi.objects.RemoteAnonServer* method), 44  
*to\_string()* (*anonapi.batch.JobBatch* method), 44