DICOMweb Client Documentation

Release 0.13.0

Markus D. Herrmann

May 06, 2019

Contents:

CHAPTER 1

Introduction

The `dicomweb-client` build distribution provides client intefaces for DICOMweb RESTful services QIDO-RS, WADO-RS and STOW-RS to search, retrieve and store DICOM objects over the web, respectively. For more information about DICOMweb please refer to the documentation of the DICOM standard, in particular PS3.18.

The `dicomweb_client` Python package exposes

- Application Programming Interface (API) (see *`api`* module)
- Command Line Interface (CLI) (see *`cli`* module)

CHAPTER 2

Installation guide

2.1 Requirements

- Python (version 2.7 or higher)
- Python package manager pip

For support of image formats:

- JPEG (libjpeg or libjpeg-turbo)
- JPEG2000 (openjpeg)
- PNG (libpng)

2.2 Installation

Pre-build package available at PyPi:

```
pip install dicomweb-client
```

Source code available at Github:

```
git clone https://github.com/clindatsci/dicomweb-client ~/dicomweb-client
pip install ~/dicomweb-client
```

CHAPTER 3

User guide

The client can be used with any DICOMweb server, such as dcm4che, orthanc or DICOMcloud.

3.1 Application Programming Interface (API)

To interact with a publicly accessible server, you only need to provide the `url` for the server address.

```
from dicomweb_client.api import DICOMwebClient

client = DICOMwebClient("https://mydicomwebserver.com")
```

Some servers expose the different DICOMweb RESTful services using different path prefixes. For example, the publicly accessible DICOMcloud server uses the prefixes `"qidors"`, `"wadors"`, and `"stowrs"` for QIDO-RS, WADO-RS, and STOW-RS, respectively. You can specify these prefixes using `qido_url_prefix`, `wado_url_prefix`, and `stow_url_prefix`.

```
from dicomweb_client.api import DICOMwebClient

client = DICOMwebClient(
    url="https://dicomcloud.azurewebsites.net",
    qido_url_prefix="qidors",
    wado_url_prefix="wadors",
    stow_url_prefix="stowrs"
)
```

To interact with servers requiring authentication, `DICOMwebClient` accepts arbitrary authentication handlers derived from `requests.auth.AuthBase` (see here for details).

```
from requests.auth import HTTPBasicAuth
from dicomweb_client.api import DICOMwebClient

client = DICOMwebClient(
    url="https://mydicomwebserver.com",
```

(continues on next page)

(continued from previous page)

```
    auth=HTTPBasicAuth('myusername', 'mypassword')
)
```

To simplify usage for `HTTPBasicAuth`, you may also directly provide a username and password using the corresponding arguments.

```
from dicomweb_client.api import DICOMwebClient

client = DICOMwebClient(
    url="https://mydicomwebserver.com",
    username="myusername",
    password="mypassword"
)
```

To interact with server requiring certificate-based authentication, you can provide the CA bundle and client certificate using the `ca_bundle` and `cert` arguments, respectively.

```
from dicomweb_client.api import DICOMwebClient

client = DICOMwebClient(
    url="https://mydicomwebserver.com",
    ca_bundle="/path/to/ca.crt",
    cert="/path/to/cert.pem"
)
```

3.1.1 STOW-RS StoreInstances

Store a single dataset obtained from a PS3.10 file:

```
from dicomweb_client.api import DICOMwebClient
import pydicom

filename = "/path/to/file.dcm"
dataset = pydicom.dcmread(filename)
client.store_instances(datasets=[dataset])
```

3.1.2 QIDO-RS SeachForStudies

Search for all studies:

```
studies = client.search_for_studies()
```

Search for studies filtering by *PatientID*:

```
studies = client.search_for_studies(search_filters={'PatientID': 'ABC123'})
```

Note that attributes can be specified in `search_filters` using either the keyword or the tag:

```
studies = client.search_for_studies(search_filters={'00100020': 'ABC123'})
```

Search for all studies but limit the number of returned results using the `limit` parameter.

```
studies_subset = client.search_for_studies(limit=100)
```

A server may also automatically limit the number of results that it returns per search request. In this case, the method can be called repeatedly to request remaining results using the `offset` parameter.

```
studies = []
offset = 0
while True:
    subset = client.search_for_studies(offset=offset)
    if len(subset) == 0:
        break
    studies.extend(subset)
    offset += len(subset)
```

3.1.3 QIDO-RS SeachForSeries

Search for all series:

```
series = client.search_for_series()
```

Search for series of a given study:

```
series = client.search_for_series('1.2.826.0.1.3680043.8.1055.1.20111103111148288.
↪98361414.79379639')
```

Search for series filtering by *AccessionNumber*:

```
series = client.search_for_series(search_filters={'AccessionNumber': '123456'})
```

Search for series filtering by *AccessionNumber* (using wildcard `?` to match a range of numbers):

```
series = client.search_for_series(search_filters={'AccessionNumber': '12345?'})
```

Search for series filtering by *SeriesDescription*:

```
series = client.search_for_series(search_filters={'SeriesDescription': 'T2 AXIAL'})
```

Search for series filtering by *SeriesDescription* (using wildcard $\star$ to match a range of descriptions):

```
series = client.search_for_series(search_filters={'SeriesDescription': 'T2 AX*'})
```

Search for series filtering by *Modality*:

```
series = client.search_for_series(search_filters={'Modality': 'CT'})
```

3.1.4 QIDO-RS SeachForInstances

Search for all instances:

```
instances = client.search_for_instances()
```

Search for instances of a given study and series:

```
instances = client.search_for_instances(
    study_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.
↪79379639',
    series_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.49685336.
↪24517034'
)
```

Search for instances filtering by *SOPClassUID*:

```
instances = client.search_for_instances(search_filters={'SOPClassUID': '1.2.840.10008.
↪5.1.4.1.1.2'})
```

3.1.5 WADO-RS RetrieveStudy

Retrieve instances of a given study:

```
instances = client.retrieve_study('1.2.826.0.1.3680043.8.1055.1.20111103111148288.
↪98361414.79379639')
```

3.1.6 WADO-RS RetrieveSeries

Retrieve instances of a given series:

```
instances = client.retrieve_series(
    study_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.
↪79379639',
    series_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.49685336.
↪24517034'
)
```

Retrieve full instances of a given series using specific JPEG 2000 transfer syntax for encoding of bulk data:

```
instance = client.retrieve_instance(
    study_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.
↪79379639',
    series_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.49685336.
↪24517034'
    media_types=(('application/dicom', '1.2.840.10008.1.2.4.90', ), )
)
```

Retrieve bulk data of instances of a given series using specific JPEG 2000 transfer syntax:

```
instance = client.retrieve_instance(
    study_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.
↪79379639',
    series_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.49685336.
↪24517034'
    media_types=(('image/jp2', '1.2.840.10008.1.2.4.90', ), )
)
```

3.1.7 WADO-RS RetrieveInstance

Retrieve full instance using default Explicit VR Little Endian transfer syntax for encoding of bulk data:

```
instance = client.retrieve_instance(
    study_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.
↪79379639',
    series_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.49685336.
↪24517034'
    sop_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.40440871.13152534
↪'
)
```

Retrieve full instance using specific JPEG 2000 transfer syntax for encoding of bulk data:

```
instance = client.retrieve_instance(
    study_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.
↪79379639',
    series_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.49685336.
↪24517034'
    sop_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.40440871.13152534
↪',
    media_types=(('application/dicom', '1.2.840.10008.1.2.4.90', ), )
)
```

Retrieve bulk data of instance using specific JPEG 2000 transfer syntax:

```
instance = client.retrieve_instance(
    study_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.
↪79379639',
    series_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.49685336.
↪24517034'
    sop_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.40440871.13152534
↪',
    media_types=(('image/jp2', '1.2.840.10008.1.2.4.90', ), )
)
```

3.1.8 WADO-RS RetrieveMetadata

Retrieve metadata for instances of a given study:

```
metadata = client.retrieve_study_metadata('1.2.826.0.1.3680043.8.1055.1.
↪20111103111148288.98361414.79379639')
```

Retrieve metadata for instances of a given series:

```
metadata = client.retrieve_series_metadata(
    study_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.
↪79379639',
    series_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.49685336.
↪24517034'
)
```

Retrieve metadata for a particular instance:

```
metadata = client.retrieve_instance_metadata(
    study_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.
↪79379639',
    series_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.49685336.
↪24517034'
```

(continues on next page)

(continued from previous page)

```
    sop_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.40440871.13152534
↪'
)
```

Note: WADO-RS RetrieveMetadata always returns metadata at the instance-level, `retrieve_study_metadata()` and `retrieve_series_metadata()` return an array of metadata items for each instance belonging to a given study and series, respectively.

3.1.9 WADO-RS RetrieveFrames

Retrieve a set of frames with default transfer syntax ("application/octet-stream"):

```
frames = client.retrieve_instance_frames(
    study_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.
↪79379639'
    series_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.49685336.
↪24517034'
    sop_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.40440871.13152534
↪'
    frame_numbers=[1, 2]
)
```

Retrieve a set of frames of a given instances as JPEG compressed image:

```
frames = client.retrieve_instance_frames(
    study_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.
↪79379639'
    series_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.49685336.
↪24517034'
    sop_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.40440871.13152534
↪'
    frame_numbers=[1, 2],
    media_types=('image/jpeg', )
)
```

Retrieve a set of frames of a given instances as compressed image in any available format:

```
frames = client.retrieve_instance_frames(
    study_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.
↪79379639'
    series_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.49685336.
↪24517034'
    sop_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.40440871.13152534
↪'
    frame_numbers=[1, 2],
    media_types=('image/*', )
)
```

Retrieve a set of frames of a given instances as either JPEG 2000 or JPEG-LS compressed image:

```
frames = client.retrieve_instance_frames(
    study_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.
↪79379639'
```

(continues on next page)

(continued from previous page)

```
    series_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.49685336.
↪24517034'
    sop_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.40440871.13152534
↪'
    frame_numbers=[1, 2],
    media_types=('image/jp2', 'image/x-jpls', )
)
```

Retrieve a set of frames of a given instances as either JPEG, JPEG 2000 or JPEG-LS lossless compressed image using specific transfer syntaxes:

```
frames = client.retrieve_instance_frames(
    study_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.
↪79379639'
    series_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.49685336.
↪24517034'
    sop_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.40440871.13152534
↪'
    frame_numbers=[1, 2],
    media_types=(
        ('image/jpeg', '1.2.840.10008.1.2.4.57', ),
        ('image/jp2', '1.2.840.10008.1.2.4.90', ),
        ('image/x-jpls', '1.2.840.10008.1.2.4.80', ),
    )
)
```

3.1.10 WADO-RS RetrieveBulkdata

Retrieve bulk data given a URL:

```
data = client.retrieve_bulkdata('https://mydicomwebserver.com/studies/...')
```

3.1.11 WADO-RS RetrieveRenderedTransaction

Retrieve a single-frame image instance rendered as a PNG compressed image:

```
frames = client.retrieve_instance_rendered(
    study_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.
↪79379639'
    series_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.49685336.
↪24517034'
    sop_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.40440871.13152534
↪'
    media_types=('image/png', )
)
```

Retrieve a single frame of a multi-frame image instance rendered as a high-quality JPEG compressed image that includes an ICC profile:

```
frames = client.retrieve_instance_frames_rendered(
    study_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.
↪79379639'
    series_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.49685336.
↪24517034'
```

(continues on next page)

(continued from previous page)

```
    sop_instance_uid='1.2.826.0.1.3680043.8.1055.1.20111103111208937.40440871.13152534
↪'
    frame_numbers=[1],
    media_types=('image/jpeg', ),
    params={'quality': 95, 'iccprofile': 'yes'}
)
```

When frames are retrieved in image format, they can be converted into a *NumPy* array using the *PIL* module:

```
from io import BytesIO

import numpy as np
from PIL import Image

image = Image.open(BytesIO(frames[0]))
array = np.array(image)
```

Warning: Retrieving images using lossy compression methods may lead to image recompression artifacts if the images have been stored lossy compressed.

3.1.12 Loading JSON Data To `pydicom`

Load metadata from JSON format into a `pydicom.dataset.Dataset` object. A common use for this is translating metadata received from a `RetrieveMetadata` or a `SearchFor`-style request:

```
from dicomweb_client.api import load_json_dataset

metadata = client.retrieve_study_metadata('1.2.826.0.1.3680043.8.1055.1.
↪20111103111148288.98361414.79379639')
metadata_datasets = [load_json_dataset(ds) for ds in metadata]
```

3.2 Command Line Interface (CLI)

Search for studies:

```
dicomweb_client --url https://dicomcloud.azurewebsites.net/qidors search studies
```

Retrieve metadata for all instances of a given study:

```
dicomweb_client --url https://dicomcloud.azurewebsites.net/wadors \
    retrieve studies \
    --study 1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.79379639 \
    metadata
```

The output can be *dicomized* for human interpretation:

```
dicomweb_client --url https://dicomcloud.azurewebsites.net/wadors \
    retrieve studies \
    --study 1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.79379639 \
    metadata \
    --dicomize
```

Retrieve the full Part 3.10 files for all instances of a given study:

```
dicomweb_client --url https://dicomcloud.azurewebsites.net/wadors \
    retrieve studies \
    --study 1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.79379639 \
    full
```

Retrieve a single frame of a given instances as JPEG compressed image:

```
dicomweb_client --url https://dicomcloud.azurewebsites.net/wadors \
    retrieve instances \
    --study 1.2.826.0.1.3680043.8.1055.1.20111103111148288.98361414.79379639 \
    --series 1.2.826.0.1.3680043.8.1055.1.20111103111208937.49685336.24517034 \
    --instance 1.2.826.0.1.3680043.8.1055.1.20111103111208937.40440871.13152534 \
    frames \
    --numbers 1 \
    --media-type image/jpeg
```

CHAPTER 4

Developer guide

Source code is available at Github and can be cloned via git:

```
git clone https://github.com/clindatsci/dicomweb-client ~/dicomweb-client
```

The `dicomweb_client` package can be installed in *develop* mode for local development:

```
pip install -e ~/dicomweb-client
```

4.1 Pull requests

Don't commit code changes to the `master` branch. New features should be implemented in a separate branch called `feature/*` and bug fixes should be applied in separate branch called `bugfix/*`.

Before creating a pull request on Github, read the coding style guideline, run the tests and check PEP8 compliance.

4.2 Coding style

Code must comply with PEP 8. The flake8 package is used to enforce compliance.

The project uses numpydoc for documenting code according to PEP 257 docstring conventions. Further information and examples for the NumPy style can be found at the NumPy Github repository and the website of the Napoleon sphinx extension.

All API classes, functions and modules must be documented (including "private" functions and methods). Each docstring must describe input parameters and return values. Types must be specified using type hints as specified by PEP 484 (see typing module).

4.3 Running tests

The project uses pytest to write and runs unit tests. Tests should be placed in a separate `tests` folder within the package root folder. Files containing actual test code should follow the pattern `test_*.py`.

Install requirements:

```
pip install -r ~/dicomweb-client/requirements_test.txt
```

Run tests (including checks for PEP8 compliance):

```
cd ~/dicomweb-client
pytest --flake8
```

4.4 Building documentation

Install requirements:

```
pip install -r ~/dicomweb-client/requirements_docs.txt
```

Build documentation in *HTML* format:

```
cd ~/dicomweb-client
sphinx-build -b html docs/ docs/build/
```

The built `index.html` file will be located in `docs/build`.

CHAPTER 5

Conformance statement

5.1 QIDO-RS

Method	Resource	Implemented
GET	SearchForStudies	Y
GET	SearchForSeries	Y
GET	SearchForInstances	Y

5.2 WADO-RS

Method	Resource	Implemented
GET	RetrieveStudy	Y
GET	RetrieveSeries	Y
GET	RetrieveInstance	Y
GET	RetrieveMetadata	Y*
GET	RetrieveBulkdata	Y
GET	RetrieveFrames	Y
GET	RetrieveRenderedTransaction	Y

- *Metadata* resource representations are requested in JSON format according to the DICOM JSON model using `application/dicom+json` media type. Retrieval of metadata in XML form using `application/dicom+xml` is not supported.

5.3 STOW-RS

Method	Resource	Implemented
POST	StoreInstances	Y

CHAPTER 6

License

DICOMweb Client is free and open source software licensed under the permissive MIT license.

CHAPTER 7

Source code

7.1 dicomweb_client package

7.1.1 dicomweb_client.api module

Application Programming Interface (API).

class dicomweb_client.api.**DICOMwebClient**(*url*, *username=None*, *password=None*, *ca_bundle=None*, *cert=None*, *qido_url_prefix=None*, *wado_url_prefix=None*, *stow_url_prefix=None*, *proxies=None*, *headers=None*, *callback=None*, *auth=None*, *gcp_service_account_key_file=None*)

Bases: object

Class for connecting to and interacting with a DICOMweb RESTful service.

base_url
unique resource locator of the DICOMweb service

Type str

protocol
name of the protocol, e.g. "https"

Type str

host
IP address or DNS name of the machine that hosts the server

Type str

port
number of the port to which the server listens

Type int

url_prefix
URL path prefix for DICOMweb services (part of *base_url*)

Type str

qido_url_prefix
URL path prefix for QIDO-RS (not part of *base_url*)

Type Union[str, NoneType]

wado_url_prefix
URL path prefix for WADO-RS (not part of *base_url*)

Type Union[str, NoneType]

stow_url_prefix
URL path prefix for STOW-RS (not part of *base_url*)

Type Union[str, NoneType]

static lookup_keyword(*tag*)
Looks up the keyword of a DICOM attribute.

Parameters **tag** (*Union[str, int, Tuple[str], pydicom.tag.Tag]*) – attribute tag (e.g. `"00080018"`)

Returns attribute keyword (e.g. `"SOPInstanceUID"`)

Return type str

static lookup_tag(*keyword*)
Looks up the tag of a DICOM attribute.

Parameters **keyword** (*str*) – attribute keyword (e.g. `"SOPInstanceUID"`)

Returns attribute tag as HEX string (e.g. `"00080018"`)

Return type str

retrieve_bulkdata(*url*, *media_types=None*, *byte_range=None*)
Retrieves bulk data from a given location.

Parameters

- **url** (*str*) – location of the bulk data
- **media_types** (*Tuple[Union[str, Tuple[str, str]]], optional*) – acceptable media types and optionally the UIDs of the corresponding transfer syntaxes
- **byte_range** (*Tuple[int], optional*) – start and end of byte range

Returns bulk data items

Return type List[bytes]

retrieve_instance(*study_instance_uid*, *series_instance_uid*, *sop_instance_uid*, *media_types=None*)
Retrieves an individual DICOM instance.

Parameters

- **study_instance_uid** (*str*) – unique study identifier
- **series_instance_uid** (*str*) – unique series identifier
- **sop_instance_uid** (*str*) – unique instance identifier

- **media_types** (*Tuple[Union[str, Tuple[str, str]]], optional*) – acceptable media types and optionally the UIDs of the corresponding transfer syntaxes

Returns data set

Return type pydicom.dataset.Dataset

retrieve_instance_frames(*study_instance_uid*, *series_instance_uid*, *sop_instance_uid*, *frame_numbers*, *media_types=None*)

Retrieves one or more frames of an individual DICOM instance.

Parameters

- **study_instance_uid** (*str*) – unique study identifier
- **series_instance_uid** (*str*) – unique series identifier
- **sop_instance_uid** (*str*) – unique instance identifier
- **frame_numbers** (*List[int]*) – one-based positional indices of the frames within the instance
- **media_types** (*Tuple[Union[str, Tuple[str, str]]], optional*) – acceptable media types and optionally the UIDs of the corresponding transfer syntaxes

Returns pixel data for each frame

Return type List[bytes]

retrieve_instance_frames_rendered(*study_instance_uid*, *series_instance_uid*, *sop_instance_uid*, *frame_number*, *media_types=None*, *params=None*)

Retrieves one or more server-side rendered frames of an individual DICOM instance.

Parameters

- **study_instance_uid** (*str*) – unique study identifier
- **series_instance_uid** (*str*) – unique series identifier
- **sop_instance_uid** (*str*) – unique instance identifier
- **frame_number** (*int*) – one-based positional index of the frame within the instance
- **media_types** (*Tuple[Union[str, Tuple[str]]], optional*) – acceptable media type (choices: `"image/jpeg"`, `"image/jp2"`, `"image/gif"`, `"image/png"`)
- **params** (*Dict[str], optional*) – additional parameters relevant for given *media_type*, e.g., `{"quality": 95}` for `"image/jpeg"` media type

Returns rendered frames

Return type bytes

Note: Not all media types are compatible with all SOP classes.

---

retrieve_instance_metadata(*study_instance_uid*, *series_instance_uid*, *sop_instance_uid*)

Retrieves metadata of an individual DICOM instance.

Parameters

- **study_instance_uid** (*str*) – unique study identifier
- **series_instance_uid** (*str*) – unique series identifier

- **sop_instance_uid** (*str*) – unique instance identifier

Returns metadata in DICOM JSON format

Return type Dict[str, dict]

retrieve_instance_rendered(*study_instance_uid*, *series_instance_uid*, *sop_instance_uid*, *media_types=None*, *params=None*)

Retrieves an individual, server-side rendered DICOM instance.

Parameters

- **study_instance_uid** (*str*) – unique study identifier
- **series_instance_uid** (*str*) – unique series identifier
- **sop_instance_uid** (*str*) – unique instance identifier
- **media_types** (*Tuple[union[str, Tuple[str]]], optional*) – acceptable media types (choices: `"image/jpeg"`, `"image/jp2"`, `"image/gif"`, `"image/png"`, `"video/gif"`, `"video/mp4"`, `"video/h265"`, `"text/html"`, `"text/plain"`, `"text/xml"`, `"text/rtf"`, `"application/pdf"`)
- **params** (*Dict[str], optional*) – additional parameters relevant for given *media_type*, e.g., `{"quality": 95}` for `"image/jpeg"`

Returns rendered instance

Return type bytes

retrieve_series(*study_instance_uid*, *series_instance_uid*, *media_types=None*)

Retrieves instances of a given DICOM series.

Parameters

- **study_instance_uid** (*str*) – unique study identifier
- **series_instance_uid** (*str*) – unique series identifier
- **media_types** (*Tuple[Union[str, Tuple[str, str]]], optional*) – acceptable media types and optionally the UIDs of the corresponding transfer syntaxes

Returns data sets

Return type List[pydicom.dataset.Dataset]

retrieve_series_metadata(*study_instance_uid*, *series_instance_uid*)

Retrieves metadata for instances of a given DICOM series.

Parameters

- **study_instance_uid** (*str*) – unique study identifier
- **series_instance_uid** (*str*) – unique series identifier

Returns metadata in DICOM JSON format

Return type Dict[str, dict]

retrieve_series_rendered(*study_instance_uid*, *series_instance_uid*, *media_types=None*, *params=None*)

Retrieves an individual, server-side rendered DICOM series.

Parameters

- **study_instance_uid** (*str*) – unique study identifier
- **series_instance_uid** (*str*) – unique series identifier

- **media_types** (*Tuple[Union[str, Tuple[str]]], optional*) – acceptable media types (choices: `"image/jpeg"`, `"image/jp2"`, `"image/gif"`, `"image/png"`, `"video/gif"`, `"video/mp4"`, `"video/h265"`, `"text/html"`, `"text/plain"`, `"text/xml"`, `"text/rtf"`, `"application/pdf"`)
- **params** (*Dict[str], optional*) – additional parameters relevant for given *media_type*, e.g., `{"quality": 95}` for `"image/jpeg"`

Returns rendered series

Return type bytes

retrieve_study(*study_instance_uid*, *media_types=None*)

Retrieves instances of a given DICOM study.

Parameters

- **study_instance_uid** (*str*) – unique study identifier
- **media_types** (*Tuple[Union[str, Tuple[str, str]]], optional*) – acceptable media types and optionally the UIDs of the corresponding transfer syntaxes

Returns data sets

Return type List[pydicom.dataset.Dataset]

retrieve_study_metadata(*study_instance_uid*)

Retrieves metadata of instances of a given DICOM study.

Parameters **study_instance_uid** (*str*) – unique study identifier

Returns metadata in DICOM JSON format

Return type Dict[str, dict]

search_for_instances(*study_instance_uid=None*, *series_instance_uid=None*, *fuzzymatching=None*, *limit=None*, *offset=None*, *fields=None*, *search_filters=None*)

Searches for DICOM instances.

Parameters

- **study_instance_uid** (*str, optional*) – unique study identifier
- **series_instance_uid** (*str, optional*) – unique series identifier
- **fuzzymatching** (*bool, optional*) – whether fuzzy semantic matching should be performed
- **limit** (*int, optional*) – maximum number of results that should be returned
- **offset** (*int, optional*) – number of results that should be skipped
- **fields** (*Union[list, tuple, set], optional*) – names of fields (attributes) that should be included in results
- **search_filters** (*Dict[str, Union[str, int, float]], optional*) – search filter criteria as key-value pairs, where *key* is a keyword or a tag of the attribute and *value* is the expected value that should match

Returns instance representations (see Instance Result Attributes)

Return type List[Dict[str, dict]]

Note: The server may only return a subset of search results. In this case, a warning will notify the client that there are remaining results. Remaining results can be requested via repeated calls using the *offset* parameter.

search_for_series(*study_instance_uid=None*, *fuzzymatching=None*, *limit=None*, *offset=None*, *fields=None*, *search_filters=None*)

Searches for DICOM series.

Parameters

- **study_instance_uid** (*str, optional*) – unique study identifier
- **fuzzymatching** (*bool, optional*) – whether fuzzy semantic matching should be performed
- **limit** (*int, optional*) – maximum number of results that should be returned
- **offset** (*int, optional*) – number of results that should be skipped
- **fields** (*Union[list, tuple, set], optional*) – names of fields (attributes) that should be included in results
- **search_filters** (*Dict[str, Union[str, int, float]], optional*) – search filter criteria as key-value pairs, where *key* is a keyword or a tag of the attribute and *value* is the expected value that should match

Returns series representations (see Series Result Attributes)

Return type List[Dict[str, dict]]

Note: The server may only return a subset of search results. In this case, a warning will notify the client that there are remaining results. Remaining results can be requested via repeated calls using the *offset* parameter.

search_for_studies(*fuzzymatching=None*, *limit=None*, *offset=None*, *fields=None*, *search_filters=None*)

Searches for DICOM studies.

Parameters

- **fuzzymatching** (*bool, optional*) – whether fuzzy semantic matching should be performed
- **limit** (*int, optional*) – maximum number of results that should be returned
- **offset** (*int, optional*) – number of results that should be skipped
- **fields** (*List[str], optional*) – names of fields (attributes) that should be included in results
- **search_filters** (*dict, optional*) – search filter criteria as key-value pairs, where *key* is a keyword or a tag of the attribute and *value* is the expected value that should match

Returns study representations (see Study Result Attributes)

Return type List[Dict[str, dict]]

Note: The server may only return a subset of search results. In this case, a warning will notify the client that there are remaining results. Remaining results can be requested via repeated calls using the *offset* parameter.

store_instances(*datasets*, *study_instance_uid=None*)
Stores DICOM instances.

Parameters

- **datasets** (*List[pydicom.dataset.Dataset]*) – instances that should be stored
- **study_instance_uid** (*str, optional*) – unique study identifier

dicomweb_client.api.**load_json_dataset**(*dataset*)
Loads DICOM Data Set in DICOM JSON format.

Parameters **dataset** (*Dict[str, dict]*) – mapping where keys are DICOM *Tags* and values are mappings of DICOM *VR* and *Value* key-value pairs

Returns

Return type pydicom.dataset.Dataset

7.1.2 dicomweb_client.cli module

Command Line Interface (CLI)

dicomweb_client.cli.**main**()
Main entry point for the `dicomweb_client` command line program.

dicomweb_client

Client for DICOMweb RESTful services.

```
usage: dicomweb_client [-h] [-v] [-u NAME] [-p PASSWORD] [--ca CERT-FILE]
                       [--cert CERT-FILE] [--url URL]
                       {search,retrieve,store} ...
```

-h, --help
show this help message and exit

-v, --verbosity
logging verbosity that maps to a logging level (default: error, -v: warning, -vv: info, -vvv: debug, -vvvv: debug + traceback); all log messages are written to standard error

-u <name>**, --user** <name>
username for authentication with the DICOMweb service

-p <password>**, --password** <password>
password for authentication with the DICOMweb service

--ca <cert-file>
path to a CA bundle file

--cert <cert-file>
path to a client certificate file in PEM format

--url <url>
uniform resource locator of the DICOMweb service

dicomweb_client retrieve

WADO-RS: Web Access to DICOM Objects by RESTful Services.

```
usage: dicomweb_client retrieve [-h] INFORMATION ENTITIES ...
```

-h, --help
show this help message and exit

dicomweb_client retrieve bulkdata

Retrieve bulk data of a DICOM object from a known location.

```
usage: dicomweb_client retrieve bulkdata [-h]
                                         [--media-type MEDIATYPE [MEDIATYPE ...]]
                                         --uri URI
```

-h, --help
show this help message and exit

--media-type <mediatype>
acceptable media type and the optionally the UID of a corresponding tranfer syntax separted by a whitespace(e.g., "image/jpeg" or "image/jpeg 1.2.840.10008.1.2.4.50")

--uri <uri>
unique resource identifier of bulk data element

dicomweb_client retrieve instances

Retrieve data for an individual DICOM instance.

```
usage: dicomweb_client retrieve instances [-h] --study UID --series UID
                                          --instance UID
                                          {metadata,full,frames} ...
```

-h, --help
show this help message and exit

--study <uid>
unique study identifier (StudyInstanceUID)

--series <uid>
unique series identifier (SeriesInstanceUID)

--instance <uid>
unique instance identifier (SOPInstanceUID)

dicomweb_client retrieve instances frames

Retrieve one or more frames of the pixel data element of an invidividual DICOM instance.

```
usage: dicomweb_client retrieve instances frames [-h] [--save]
                                                 [--output-dir PATH]
                                                 [--media-type MEDIATYPE [MEDIATYPE ..
↪.]]
                                                 [--numbers NUM [NUM ...]]
                                                 [--show]
```

-h, --help
show this help message and exit

--save
whether downloaded data should be saved

--output-dir <path>
path to directory where downloaded data should be saved

--media-type <mediatype>
acceptable media type and the optionally the UID of a corresponding tranfer syntax separted by a whitespace(e.g., "image/jpeg" or "image/jpeg 1.2.840.10008.1.2.4.50")

--numbers <num>
frame numbers

--show
display retrieved images

dicomweb_client retrieve instances full

Retrieve a DICOM instance.

```
usage: dicomweb_client retrieve instances full [-h] [--save]
                                               [--output-dir PATH]
                                               [--media-type MEDIATYPE [MEDIATYPE ...
↪]]
```

-h, --help
show this help message and exit

--save
whether downloaded data should be saved

--output-dir <path>
path to directory where downloaded data should be saved

--media-type <mediatype>
acceptable media type and the optionally the UID of a corresponding tranfer syntax separted by a whitespace(e.g., "image/jpeg" or "image/jpeg 1.2.840.10008.1.2.4.50")

dicomweb_client retrieve instances metadata

Retrieve metadata of an invidividual DICOM instance.

```
usage: dicomweb_client retrieve instances metadata [-h]
                                                   [--prettify | --dicomize]
                                                   [--save]
                                                   [--output-dir PATH]
```

-h, --help
show this help message and exit

--prettify
pretty print JSON response message

--dicomize
convert JSON response message to DICOM data set

--save
whether downloaded data should be saved

--output-dir <path>
path to directory where downloaded data should be saved

dicomweb_client retrieve series

Retrieve data for all DICOM instances of a given DICOM series.

```
usage: dicomweb_client retrieve series [-h] --study UID --series UID
                                       {metadata,full} ...
```

-h, --help
show this help message and exit

--study <uid>
unique study identifier (StudyInstanceUID)

--series <uid>
unique series identifier (SeriesInstanceUID)

dicomweb_client retrieve series full

Retrieve DICOM instances of a given DICOM series.

```
usage: dicomweb_client retrieve series full [-h] [--save] [--output-dir PATH]
                                            [--media-type MEDIATYPE [MEDIATYPE ...]]
```

-h, --help
show this help message and exit

--save
whether downloaded data should be saved

--output-dir <path>
path to directory where downloaded data should be saved

--media-type <mediatype>
acceptable media type and the optionally the UID of a corresponding tranfer syntax separted by a whitespace(e.g., “image/jpeg” or “image/jpeg 1.2.840.10008.1.2.4.50”)

dicomweb_client retrieve series metadata

Retrieve metadata of DICOM instances of a given DICOM series.

```
usage: dicomweb_client retrieve series metadata [-h] [--prettify | --dicomize]
                                                [--save] [--output-dir PATH]
```

-h, --help
show this help message and exit

--prettify
pretty print JSON response message

--dicomize
convert JSON response message to DICOM data set

--save
whether downloaded data should be saved

--output-dir <path>
path to directory where downloaded data should be saved

dicomweb_client retrieve studies

Retrieve data for all DICOM instances of a given DICOM study.

```
usage: dicomweb_client retrieve studies [-h] --study UID {metadata,full} ...
```

-h, --help
show this help message and exit

--study <uid>
unique study identifier (StudyInstanceUID)

dicomweb_client retrieve studies full

Retrieve DICOM instances of a given DICOM study.

```
usage: dicomweb_client retrieve studies full [-h] [--save] [--output-dir PATH]
                                             [--media-type MEDIATYPE [MEDIATYPE ...]]
```

-h, --help
show this help message and exit

--save
whether downloaded data should be saved

--output-dir <path>
path to directory where downloaded data should be saved

--media-type <mediatype>
acceptable media type and the optionally the UID of a corresponding tranfer syntax separted by a whitespace(e.g., “image/jpeg” or “image/jpeg 1.2.840.10008.1.2.4.50”)

dicomweb_client retrieve studies metadata

Retrieve metadata of DICOM instances of a given DICOM study.

```
usage: dicomweb_client retrieve studies metadata [-h]
                                                 [--prettify | --dicomize]
                                                 [--save] [--output-dir PATH]
```

-h, --help
show this help message and exit

--prettify
pretty print JSON response message

--dicomize
convert JSON response message to DICOM data set

--save
whether downloaded data should be saved

--output-dir <path>
path to directory where downloaded data should be saved

dicomweb_client search

QIDO-RS: Query based on ID for DICOM Objects by RESTful Serices.

```
usage: dicomweb_client search [-h] INFORMATION ENTITIES ...
```

-h, --help
show this help message and exit

dicomweb_client search instances

Search for DICOM instances.

```
usage: dicomweb_client search instances [-h] [--prettify | --dicomize]
                                        [--filter KEY=VALUE] [--field NAME]
                                        [--limit NUM] [--offset NUM] [--fuzzy]
                                        [--study UID] [--series UID]
```

-h, --help
show this help message and exit

--prettify
pretty print JSON response message

--dicomize
convert JSON response message to DICOM data set

--filter <key=value>
query filter criterion

--field <name>
field that should be included in response

--limit <num>
number of items that should be maximally retrieved

--offset <num>
number of items that should be skipped

--fuzzy
perform fuzzy matching

--study <uid>
unique study identifer (StudyInstanceUID)

--series <uid>
unique series identifier (SeriesInstanceUID)

dicomweb_client search series

Search for DICOM series.

```
usage: dicomweb_client search series [-h] [--filter KEY=VALUE] [--field NAME]
                                     [--limit NUM] [--offset NUM] [--fuzzy]
                                     [--prettify | --dicomize] [--study UID]
```

-h, --help
show this help message and exit

--filter <key=value>
query filter criterion

--field <name>
field that should be included in response

--limit <num>
number of items that should be maximally retrieved

--offset <num>
number of items that should be skipped

--fuzzy
perform fuzzy matching

--prettify
pretty print JSON response message

--dicomize
convert JSON response message to DICOM data set

--study <uid>
unique study identifer (StudyInstanceUID)

dicomweb_client search studies

Search for DICOM studies.

```
usage: dicomweb_client search studies [-h] [--filter KEY=VALUE] [--field NAME]
                                      [--limit NUM] [--offset NUM] [--fuzzy]
                                      [--prettify | --dicomize]
```

-h, --help
show this help message and exit

--filter <key=value>
query filter criterion

--field <name>
: field that should be included in response

--limit <num>
: number of items that should be maximally retrieved

--offset <num>
: number of items that should be skipped

--fuzzy
: perform fuzzy matching

--prettify
: pretty print JSON response message

--dicomize
: convert JSON response message to DICOM data set

dicomweb_client store

STOW-RS: Store Over the Web by RESTful Services.

```
usage: dicomweb_client store [-h] INFORMATION ENTITIES ...
```

-h, --help
: show this help message and exit

dicomweb_client store instances

Store DICOM instances.

```
usage: dicomweb_client store instances [-h] [--study UID] PATH [PATH ...]
```

path
: paths to DICOM files that should be loaded

-h, --help
: show this help message and exit

--study <uid>
: unique study identifer (StudyInstanceUID)

7.1.3 dicomweb_client.error module

Custom error classes

exception dicomweb_client.error.**DICOMJSONError**
: Bases: ValueError

  Exception class for malformatted DICOM JSON.

exception dicomweb_client.error.**HTTPError**(**args*, ***kwargs*)
: Bases: requests.exceptions.HTTPError

  Exception class for HTTP requests with failure status codes.

7.1.4 dicomweb_client.log module

Utility functions for logging configuration

dicomweb_client.log.**configure_logging**(*verbosity*)

Configures the root logger with a "stderr" stream handler that directs logging messages to standard error (to allow capturing program standard output, e.g. in order to redirect it to a file).

Logging verbosity maps to levels as follows:

```
0 -> no messages
1 -> CRITICAL, ERROR & WARN/WARNING messages
2 -> CRITICAL, ERROR, WARN/WARNING, & INFO messages
3 -> CRITICAL, ERROR, WARN/WARNING, INFO & DEBUG messages
4 -> all messages
```

Parameters **verbosity** (*int*) – logging verbosity

Returns package root logger

Return type logging.Logger

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index

Symbols

B

C

D

H

L

M

P

Q

R

S

U

W