OpenREM

Release 1.0.0b2

OpenREM Contributers

25.06.2023

Inhaltsverzeichnis

OpenREM ist eine freie, Opensource-Anwendung zur Überwachung der Patientenexposition. Die Software können die Daten einer großen Anzahl von Röntgengeräten importiert und gefiltert über entsprechende graphische Darstellungen analysiert werden. Die Software ermöglicht auch den Export der Daten für eine spätere Analsyse durch qualifiziertes Personal aus der Medizinphysik.

Details finden sich unter openrem.org.

Diese Dokumente beziehen sich auf die Version von OpenREM, die sich derzeit in der Entwicklung befindet. Informationen zur aktuellen Version finden Sie unter https://docs.openrem.org/

Diese Dokumente sind in britischem Englisch - wenn Sie helfen möchten, sie in andere Sprachen zu übersetzen, kontaktieren Sie uns bitte: @_OpenREM, Bitbucket, Google Groups oder gehen Sie zu Weblate, um loszulegen. Weitere Einzelheiten: *Translating OpenREM strings*.

Inhalt:

Installation

1.1 Installationsoptionen

There are three supported installation options for OpenREM v1.0:

- Docker
- Native install on Linux
- Native install on Windows

1.1.1 Docker

This is the quickest and easiest way of installing a fully functioning OpenREM instance, complete with database, web server and DICOM server, on any operating system that supports Docker with Linux containers. This includes Windows 10 with Docker Desktop, but currently excludes Windows Server, though this may change with availability of WSL2 for Windows Server 2022.

The Docker installation has mostly been tested with Ubuntu server, but has also been used successfully with Podman on Redhat Enterprise Linux and other distributions.

Existing Windows or Linux installations of OpenREM 0.10 can be upgraded to run in a Docker installation.

It is advisable that the server OpenREM is installed on has access to the internet to get images from Docker and security updates for the operating system. However, if this is not possible the Docker images can be obtained on a computer that does have access to the internet and transferred to the ‚offline‘ server for installation.

Docker install

Preparation

- Install Docker and Docker Compose (may be installed automatically with Docker)

- Download https://bitbucket.org/openrem/docker/get/1.0.0b2.zip

Install

- Extract the ZIP file and open a shell (command prompt) in the new folder

- Customise variables in the following two files:

    - `.env.prod`

    - the `orthanc_1` section of `docker-compose.yml`

- If you are using SELinux, you will also need to edit the nginx and orthanc bind mounts in `docker-compose.yml`

A full description of the options are found in:

Docker env configuration

Edit the `.env.prod` file to customise your installation. There should be no space between the variable name, the = and the value. Everything after the = until the end of the line is transferred as the value. These settings take effect when docker-compose is started or restarted.

Variablen, die auf alle Fälle angepasst werden sollten

Wert für den geheimen Schlüssel festlgen. Entweder selbst erstellen oder über Tools wie http://www.miniwebtool.com/django-secret-key-generator/ generieren lassen:

```
SECRET_KEY=
```

`DJANGO_ALLOWED_HOSTS` is a string of hostnames or IPs with a space between each:

- `nginx` is required for internal use

- `localhost 127.0.0.1 [::]` allows access on the server using the localhost name or IP (using IPv4 or IPv6)

- add the name and/or IP address of your server so it can be accessed from other computers on your network.

For example: `DJANGO_ALLOWED_HOSTS=nginx localhost 127.0.0.1 [::1] myservername`

```
DJANGO_ALLOWED_HOSTS=nginx localhost 127.0.0.1 [::1]
```

Variablen, die bei Problemen bei der Fehlersuche helfen

Set to 1 to enable Django debugging mode.

```
DEBUG=
```

Set the log level. Options are `DEBUG`, `INFO`, `WARNING`, `ERROR`, and `CRITICAL`, with progressively less logging.

```
LOG_LEVEL=
LOG_LEVEL_QRSCU=
LOG_LEVEL_EXTRACTOR=
```

Variablen, die an die eigene Ümgebung angepasst werden sollten

E-mail server settings

```
EMAIL_HOST=
EMAIL_PORT=
EMAIL_HOST_USER=
EMAIL_HOST_PASSWORD=
EMAIL_USE_TLS=
EMAIL_USE_SSL=
EMAIL_DOSE_ALERT_SENDER=
EMAIL_OPENREM_URL=
```

The host name and port of the e-mail server that you wish to use must be entered in the `EMAIL_HOST` and `EMAIL_PORT` fields. `EMAIL_HOST` might be your institution's Outlook/Exchange server.

If the e-mail server is set to only allow authenticated users to send messages then a suitable user and password must be entered in the `EMAIL_HOST_USER` and `EMAIL_HOST_PASSWORD` fields. If this approach is used then it may be useful to request that an e-mail account be created specifically for sending these OpenREM alert messages.

It may be possible to configure the e-mail server to allow sending of messages that originate from the OpenREM server without authentication, in which case the user and password settings should not be required.

Die Optionen `EMAIL_USE_TLS` und `EMAIL_USE_SSL` sollten gemäß den Vorgaben zur Verschlüsselung auf dem E-Mail Server konfiguriert werden. Verwenden Sie eine `0` (Standard) wenn dies nicht zutrifft und 1 wenn dies Option verwendet werden soll. Nur jeweils eine der beiden Optionen sollte auf 1 gesetzt werden.

The `EMAIL_DOSE_ALERT_SENDER` should contain the e-mail address that you want to use as the sender address.

The `EMAIL_OPENREM_URL` must contain the URL of your OpenREM installation in order for hyperlinks in the e-mail alert messages to function correctly.

Regionalisierung

Local time zone for this installation. Choices can be found here: http://en.wikipedia.org/wiki/List_of_tz_zones_by_name although not all choices may be available on all operating systems:

```
TIME_ZONE=Europe/London
```

Sprach-Code für die Installation. Mögliche Werte finden sich unter: http://www.i18nguy.com/unicode/language-identifiers.html

```
LANGUAGE_CODE=en-us
```

Wird dieser Wert auf Falsch gestellt, wird Django einige Optimierungen durchführen:

```
USE_I18N=True
```

Wird dieser Wert auf False gesetzt, wird Django die Formate für Datum, Zahlen und Kalenderdaten nicht anhand der aktuellen Lokalisierung ändern:

```
USE_L10N=True
```

Wird dieser Wert auf False (Standard) gesetzt, wird Django die zeitzonenbezogenen Datumsangaben nicht verwenden:

```
USE_TZ=False
```

XLSX date and time settings for exports:

```
XLSX_DATE=dd/mm/yyyy
XLSX_TIME=hh:mm:ss
```

Einstellungen für den virtuellen Ordner

See *Running the OpenREM website in a virtual directory* for details of these variables - normally these can be left commented out.

Device Observer UID settings

OpenREM users have found one x-ray system which incorrectly sets the Device Observer UID to be equal to the Study Instance UID. In this situation a new entry is created in the display name settings for every new exam that arrives in OpenREM, making the display name table fill with many duplicate entries for the same system. To avoid this problem a list of models can be specified using the variable below - OpenREM will ignore the Device Observer UID value when creating new display names for any model in this list. The model name text must exactly match what is contained in the system's Manufacturer's Model Name DICOM tag (0008,1090).

```
IGNORE_DEVICE_OBSERVER_UID_FOR_THESE_MODELS = ['GE OEC Fluorostar']
```

Variables that should only be changed if you know what you are doing

```
## Database settings
SQL_HOST=db
SQL_ENGINE=django.db.backends.postgresql
SQL_PORT=5432
DATABASE=postgres
POSTGRES_USER=openremuser
POSTGRES_PASSWORD=openrem_pass
POSTGRES_DB=openrem_prod

## Paths
MEDIA_ROOT=/home/app/openrem/mediafiles
STATIC_ROOT=/home/app/openrem/staticfiles
LOG_ROOT=/logs
```

Variables that shouldn't be changed

Changing this will mean some OpenREM functions will fail

```
DOCKER_INSTALL=1
```

DICOM store configuration (Orthanc)

Orthanc provides the DICOM Store functionality to enable scanners to send directly to OpenREM, and for query-retrieve to function. Configuration is in the orthanc section of `docker-compose.yml`

OpenREM Lua script configuration

This file is formatted as YAML:

- Strings need to quoted or placed on a new line after a |

- A : and a space separate the variable name and the value, and spaces are used at the start of the line to create a hierarchy. See the examples below.

Für Änderungen die Datei `docker-compose.yml` editieren. Diese werden beim nächsten Start von `docker-compose` wirksam.

Find the `orthanc_1` definition near the end of the file.

Objects to be ignored

Lists of things to ignore. Orthanc will ignore anything matching the content of these comma separated lists: they will not be imported into OpenREM. Some examples have been added below - note the formatting syntax. `STATION_NAMES_TO_IGNORE` has the value on a new line with a | instead of being quoted, to show this syntax option:

```
environment:
  MANUFACTURERS_TO_IGNORE: "{'Faxitron X-Ray LLC', 'Gendex-KaVo'}"
  MODEL_NAMES_TO_IGNORE: "{'CR 85', 'CR 75'}"
```

```
STATION_NAMES_TO_IGNORE: |
  {'CR85 Main', 'CR75 Main'}
SOFTWARE_VERSIONS_TO_IGNORE: "{'VixWin Platinum v3.3'}"
DEVICE_SERIAL_NUMBERS_TO_IGNORE: "{'SCB1312016'}"
```

Extraktor für ältere Toshiba Dosisdateien

Funktion für das extrahieren der Dosisinformation aus älteren Toshiba und GE Geräte ein- oder ausschalten und für welchen Gerätetyp dies erfolgen soll. Jedes System sollte in der Form {'Manufacturer', 'Model name'} aufgeführt werden, wie das nachstehende Beispiel zeigt:

```
environment:
  USE_TOSHIBA_CT_EXTRACTOR: "true"
  TOSHIBA_EXTRACTOR_SYSTEMS: |
    {{'Toshiba', 'Aquilion'}, {'GE Medical Systems', 'Discovery STE'},}
```

Physics Filtering

Diesen Wert auf true setzen, wenn Orthanc auch physikalische Teststudien speichern und diese im Ordner physics_to_keep_folder ablegen soll. Wird dieser Wert auf "false" gesetzt ist diese Funktion deaktiviert

```
environment:
  USE_PHYSICS_FILTERING: "true"
```

Eine Liste die mit Patientenname und ID abgeglichen wird um zu entscheiden ob die Bilder gespeichert werden sollen. Alles was mit dieser Liste übereinstimmt wird im Ordner physics_to_keep_folder abgelegt.

```
environment:
  PHYSICS_TO_KEEP: "{'physics',}"
```

Orthanc Konfiguration

This section is formatted as JSON. It can contain any configuration options that appear in the standard Orthanc orthanc.json file, but the ones that are needed for OpenREM are included as standard and described below.

- Strings need to quoted with double quotes ".

DICOM Application Entity Title

Application Entity Title of the Store Server. Should be up to 16 characters, no spaces. This server isn't fussy by default, so if remote nodes connect using a different AETitle that is ok.

```
ORTHANC_JSON: |
  {
    // DICOM Store configuration
    "DicomAet" : "OPENREM",
  }
```

DICOM Port

The default port for DICOM store is set to `104`.

To use a different port, change the first number of the pair in ports. The first number is the port exposed outside of Docker, the second number is used internally by the Orthanc container.

For example, to use port 8104:

```
ports:
# DICOM store port (first number)
  - 8104:4242
```

Orthanc web interface

There will normally not be any studies in the Orthanc database once they have been processed, but if you want to enable the Orthanc web viewer, enable the port in and set `RemoteAccessAllowed` to `true` in the `ORTHANC_JSON` section. The first number in the port configuration can be changed if required:

```
ports:
# Othanc web interface
  - 8042:8042
```

```
ORTHANC_JSON: |
  {
    "Name" : "OpenREM Orthanc",
    "RemoteAccessAllowed" : true,
    "AuthenticationEnabled" : true,
    "RegisteredUsers" : {
      "orthancuser": "demo"
    },
  }
```

Pfad für Lua Skripte

Hier wird der Pfad im Orthanc-Container zum OpenREM Lua-Skript angegeben – dieser sollte nicht verändert werden (weitere Optionen siehe unten).

```
ORTHANC_JSON: |
  {
    // OpenREM Lua Script
    "LuaScripts" : [
      "/etc/share/orthanc/scripts/openrem_orthanc_config_docker.lua"
    ]
  }
```

Advanced options

Multiple stores

If you need more than one DICOM Store server, to listen on a different port for example, copy the whole `orthanc_1` section in `docker-compose.yml` and paste it after the `orthanc_1` block. Rename to `orthanc_2`.

Beim nächsten Starten von `docker-compose` wird der zusätzliche Orthanc Container gestartet. `docker-compose.yml` kann auch zum anhalten der Container verwendet werden, wenn Sie also den zusätzlichen Othanc Container löschen wollen, stoppen Sie zunächst die Container.

Advanced Orthanc configuration

Any of the Orthanc configuration settings can be set in the `ORTHANC_JSON` section. The default configuration can be seen on the Orthanc Server webpages including documentation as to how they are used.

A custom version of the `openrem_orthanc_config_docker.lua` script can be used if required. Copy the existing one and place the new one, with a new name, in the `orthanc/` folder, and set the `LuaScripts` value in `ORTHANC_JSON` to match.

Pay special attention to the first sections, up to the `ToAscii` function, these sections have been changed for the Docker implementation.

Docker SELinux configuration

SELinux will prevent bind mounts in Docker with the standard configuration, which will be seen because Orthanc fails to start. SELinux is commonly enabled on Red Hat, Fedora and associated distributions.

The `docker-compose.yml` file needs to be edited to fix this.

Change nginx configuration

Find the following section:

```
nginx:
  container_name: openrem-nginx
  restart: unless-stopped
  image: nginx:1.17.8-alpine
  volumes:
    - media_volume:/home/app/openrem/mediafiles
    - static_volume:/home/app/openrem/staticfiles
# For SELinux (RedHat, Fedora etc), add :z to the end of next two lines
    - ./nginx-conf/conf.d:/etc/nginx/conf.d
    - ./nginx-conf/certs:/etc/ssl/private
```

Follow the instruction to edit the `nginx-conf` lines, like this:

```
# For SELinux (RedHat, Fedora etc), add :z to the end of next two lines
    - ./nginx-conf/conf.d:/etc/nginx/conf.d:z
    - ./nginx-conf/certs:/etc/ssl/private:z
```

Change the Orthanc configuration

Find the following section:

```
orthanc_1:
  container_name: openrem-orthanc-1
  restart: unless-stopped
  image: openrem/orthanc
  volumes:
    - imports_volume:/imports
# For SELinux (RedHat, Fedora etc), add :z to the end of next line
    - ./orthanc:/etc/share/orthanc/scripts/
```

Follow the instruction to edit the `orthanc_1` line, like this:

```
# For SELinux (RedHat, Fedora etc), add :z to the end of next line
    - ./orthanc:/etc/share/orthanc/scripts/:z
```

Start the containers with:

```
$ docker-compose up -d
```

Get the database and translations ready:

```
$ docker-compose exec openrem python manage.py makemigrations remapp --noinput
$ docker-compose exec openrem python manage.py migrate --noinput
$ docker-compose exec openrem python manage.py loaddata openskin_safelist.json
$ docker-compose exec openrem python manage.py collectstatic --noinput --clear
$ docker-compose exec openrem python manage.py compilemessages
$ docker-compose exec openrem python manage.py createsuperuser
```

Open a web browser and go to http://localhost/

If you want to run the OpenREM in a virtual directory (like http://server/dms/) there is further configuration to be done - go to *Running the OpenREM website in a virtual directory*.

Upgrade to Docker

These instructions assume:

- You are upgrading from 0.10.0.

- You are using a PostgreSQL database in the existing installation.

- That existing Linux installs followed the instructions in the previous releases, with the *openrem-function* format that changed in the 0.9.1 release (*Systemd service names in Ubuntu installs*).

If not you will need to adapt the instructions as necessary.

- **Upgrades from 0.9.1 or earlier should review** *Upgrade to OpenREM 0.10.0 from 0.7.3 or later*. – needs changing

Upgrade to OpenREM 0.10.0 from 0.7.3 or later

Upgrades to OpenREM 1.0 can only be made from version 0.10.0. Installations earlier than that need to be updated to version 0.10.0 before updating to version 1.0.

These instructions can be used to upgrade any database from version 0.7.3 or later. 0.7.3 was released in August 2016. For upgrades from versions earlier than that, please review the upgrade instructions for that version in the 0.10.0-docs.

Upgrade preparation

Python 2.7.9 or later must be installed, but it must still be Python 2.7 and not any of the Python 3 releases.

To check the Python version, activate the virtualenv if you are using one, then:

```
$ python -V
```

If the version is earlier than `2.7.9`, then an upgrade is needed. If the version is `3.x`, then Python 2.7 must be installed.

Ubuntu Linux

- Check which version of Ubuntu is installed (`lsb_release -a`)

- If it is 14.04 LTS (Trusty), then an operating system upgrade or migration to a new server is required. If migrating, ensure the version of OpenREM installed on the new server is the same as the one on the old server, then *Database restore* following the instructions and when up and running again perform the upgrade on the new server

- 16.04 LTS (Xenial) or later should have 2.7.11 or later available.

- For other Linux distributions check in their archives for which versions are available.

Windows

- A newer version of Python 2.7 can be downloaded from python.org and installed over the current version.

Linux and Windows

- With a version of Python 2.7.9 or later, setuptools can be updated (activate virtualenv if using one):

    ```
    $ pip install setuptools -U
    ```

Upgrade

- Back up your database

    - For PostgreSQL on linux you can refer to *Database backup*

    - For PostgreSQL on Windows you can refer to *Windows installations*

    - For a non-production SQLite3 database, simply make a copy of the database file

- Stop any Celery workers

- Consider temporarily disabling your DICOM Store SCP, or redirecting the data to be processed later

- If you are using a virtualenv, activate it

- Install specific versions of some packages that are needed:

```
$ pip install django-crispy-forms==1.8.1
$ pip install django-solo==1.1.5
$ pip install flower==0.9.5
```

- Install specific version of Celery:

    Linux server:

    ```
    $ pip install celery==4.2.2
    ```

    Windows server:

    ```
    D:\>pip install celery==3.1.25
    ```

- Install the new version of OpenREM:

    ```
    $ pip install openrem==0.10.0
    ```

Update the configuration

Locate and edit your local_settings file

- Ubuntu linux: `/usr/local/lib/python2.7/dist-packages/openrem/openremproject/local_settings.py`

- Other linux: `/usr/lib/python2.7/site-packages/openrem/openremproject/local_settings.py`

- Linux virtualenv: `vitualenvfolder/lib/python2.7/site-packages/openrem/openremproject/local_settings.py`

- Windows: `C:\Python27\Lib\site-packages\openrem\openremproject\local_settings.py`

- Windows virtualenv: `virtualenvfolder\Lib\site-packages\openrem\openremproject\local_settings.py`

Add additional log file configuration - changed with 0.8

Add the new extractor log file configuration to the `local_settings.py` - you can copy the ‚Logging configuration‘ section here if you haven't made any changes. The addition that needs to be inserted are the lines relating to the `extractor` log file. This is only for upgrading the database - the `local_settings.py` file will be updated again for the upgrade to 1.0:

```python
# Logging configuration
# Set the log file location. The example places the log file in the media directory.␣
↪Change as required - on linux
# systems you might put these in a subdirectory of /var/log/. If you want all the logs␣
↪in one file, set the filename
# to be the same for each one.
import os
LOG_ROOT = MEDIA_ROOT
logfilename = os.path.join(LOG_ROOT, "openrem.log")
qrfilename = os.path.join(LOG_ROOT, "openrem_qr.log")
storefilename = os.path.join(LOG_ROOT, "openrem_store.log")
extractorfilename = os.path.join(LOG_ROOT, "openrem_extractor.log")
```

(Fortsetzung auf der nächsten Seite)

```
LOGGING['handlers']['file']['filename'] = logfilename          # General logs
LOGGING['handlers']['qr_file']['filename'] = qrfilename        # Query Retrieve SCU logs
LOGGING['handlers']['store_file']['filename'] = storefilename  # Store SCP logs
LOGGING['handlers']['extractor_file']['filename'] = extractorfilename  # Extractor logs

# Set log message format. Options are 'verbose' or 'simple'. Recommend leaving as 'verbose'.
LOGGING['handlers']['file']['formatter'] = 'verbose'        # General logs
LOGGING['handlers']['qr_file']['formatter'] = 'verbose'     # Query Retrieve SCU logs
LOGGING['handlers']['store_file']['formatter'] = 'verbose'  # Store SCP logs
LOGGING['handlers']['extractor_file']['formatter'] = 'verbose'  # Extractor logs

# Set the log level. Options are 'DEBUG', 'INFO', 'WARNING', 'ERROR', and 'CRITICAL', with␣
↪progressively less logging.
LOGGING['loggers']['remapp']['level'] = 'INFO'                  # General logs
LOGGING['loggers']['remapp.netdicom.qrscu']['level'] = 'INFO'    # Query Retrieve SCU␣
↪logs
LOGGING['loggers']['remapp.netdicom.storescp']['level'] = 'INFO'  # Store SCP logs
LOGGING['loggers']['remapp.extractors.ct_toshiba']['level'] = 'INFO'  # Toshiba RDSR␣
↪creation extractor logs
```

Migrate the database

In a shell/command window, move into the `openrem` folder:

- Ubuntu linux: `/usr/local/lib/python2.7/dist-packages/openrem/`

- Other linux: `/usr/lib/python2.7/site-packages/openrem/`

- Linux virtualenv: `vitualenvfolder/lib/python2.7/site-packages/openrem/`

- Windows: `C:\Python27\Lib\site-packages\openrem\`

- Windows virtualenv: `virtualenvfolder\Lib\site-packages\openrem\`

```
python manage.py makemigrations remapp
python manage.py migrate remapp
```

Systemd service names in Ubuntu installs

Systemd service files were renamed in the the 0.9.1 docs to use *openrem-function* rather than *function-openrem*. To update the service files accordingly, follow the following steps. **This is optional**, but will make finding them easier (e.g. `sudo systemctl status openrem-[tab][tab]` will list them) and these names are assumed for the *Upgrade to Docker* and *Upgrading a native Linux install* docs. However, only the `gunicorn` service remains after the upgrade to 1.0, so you may find it easier just to remember the only service names, or just rename that one.

```
sudo systemctl stop gunicorn-openrem.service
sudo systemctl stop celery-openrem.service
sudo systemctl stop flower-openrem.service

sudo systemctl disable gunicorn-openrem.service
sudo systemctl disable celery-openrem.service
```

```
sudo systemctl disable flower-openrem.service

sudo mv /etc/systemd/system/{gunicorn-openrem,openrem-gunicorn}.service
sudo mv /etc/systemd/system/{celery-openrem,openrem-celery}.service
sudo mv /etc/systemd/system/{flower-openrem,openrem-flower}.service

sudo systemctl enable openrem-gunicorn.service
sudo systemctl enable openrem-celery.service
sudo systemctl enable openrem-flower.service

sudo systemctl start openrem-gunicorn.service
sudo systemctl start openrem-celery.service
sudo systemctl start openrem-flower.service
```
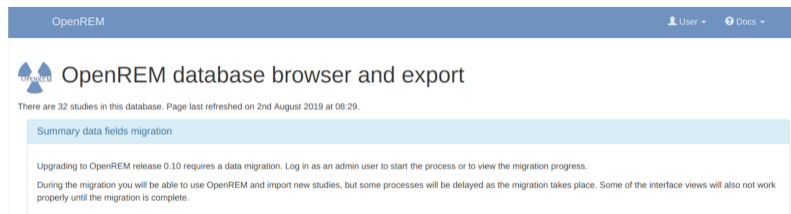
Upgrade to 1.0

Now return to *Installation* instructions to follow the instructions to 1.0 for your preferred server solution.

After upgrading to version 1.0, there will be automatic tasks that are created to populate the summary fields introduced in version 0.10.

Log in as an administrator to start the migration process. If you have a large number of studies in your database this can take some time. A large database (several hundred studies) on slow disks might take a day or two, on faster disks or with a smaller database it could take from a few minutes to an hour or so. You will be able to monitor the progress on the home page as seen in the figure at the bottom of this page.
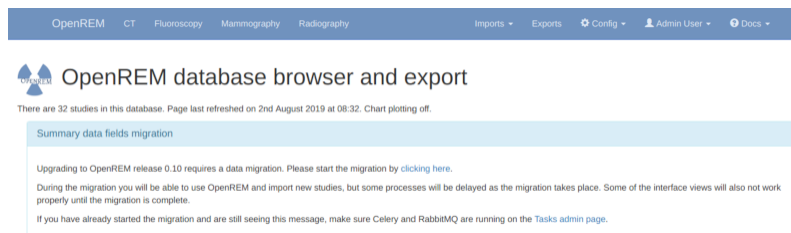
One task per modality type (CT, fluoroscopy, mammography and radiography) is generated to create a task per study in each modality to populate the new fields for that study. If the number of workers is the same or less than the number of modality types in your database then the study level tasks will all be created before any of them are executed as all the workers will be busy. Therefore there might be a delay before the progress indicators on the OpenREM front page start to update. You can review the number of tasks being created on the `Config -> Tasks` page.

Before the migration is complete, some of the information on the modality pages of OpenREM will be missing, such as the dose information for example, but otherwise everything that doesn't rely on Celery workers will work as normal. Studies sent directly to be imported will carry on during the migration, but query-retrieve tasks will get stuck behind the migration tasks.

When the process is complete the ‚Summary data fields migration' panel will disappear and will not be seen again.

Upgrade process from a PostgresQL database

Stop the existing services

- Linux:

```
$ sudo systemctl stop orthanc
$ sudo systemctl stop nginx
$ sudo systemctl stop openrem-gunicorn
$ sudo systemctl stop openrem-flower
$ sudo systemctl stop openrem-celery
$ sudo systemctl stop rabbitmq-server
$ sudo systemctl disable orthanc
$ sudo systemctl disable nginx
$ sudo systemctl disable openrem-gunicorn
$ sudo systemctl disable openrem-flower
$ sudo systemctl disable openrem-celery
$ sudo systemctl disable rabbitmq-server
```

- Windows: stop the following services

  - Orthanc or Conquest

  - IIS OpenREM site or other webserver

  - Flower

  - Celery

  - RabbitMQ

Establish existing database details

Review the current `local_settings.py` for the database settings and location of the `MEDIA_ROOT` folder. The file is in:

- Ubuntu linux: `/usr/local/lib/python2.7/dist-packages/openrem/openremproject/local_settings.py`

- Other linux: `/usr/lib/python2.7/site-packages/openrem/openremproject/local_settings.py`

- Linux virtualenv: `vitualenvfolder/lib/python2.7/site-packages/openrem/openremproject/local_settings.py`

- Windows: `C:\Python27\Lib\site-packages\openrem\openremproject\local_settings.py`

- Windows virtualenv: `virtualenvfolder\Lib\site-packages\openrem\openremproject\local_settings.py`

Export the database

- Open a command line window

- Windows: go to Postgres bin folder, for example:

```
$ cd "C:\Program Files\PostgreSQL\9.6\bin"
```

- Dump the database:

  - Use the username (`-U openremuser`) and database name (`-d openremdb`) from `local_settings.py`

  - Use the password from `local_settings.py` when prompted

  - For linux, the command is `pg_dump` (no `.exe`)

  - Set the path to somewhere suitable to dump the exported database file

```
$ pg_dump.exe -U openremuser -d openremdb -F c -f path/to/export/
→openremdump.bak
```

Set up the new installation

- Install Docker

- Download and extract https://bitbucket.org/openrem/docker/get/develop.zip and open a shell (command window) in the new folder

- Customise variables in `.env.prod`, the `orthanc_1` section in `docker-compose.yml` and in `orthanc_1.json` as necessary. A full description of the options are found in:

Start the containers with:

```
$ docker-compose up -d
```

Copy the database backup to the postgres docker container and import it. If you have changed the database variables, ensure that:

- the database user (`-U openremuser`) matches `POSTGRES_USER` in `.env.prod`

- the database name (`-d openrem_prod`) matches `POSTGRES_DB` in `.env.prod`

They don't have to match the old database settings. The filename in both commands (`openremdump.bak`) should match your backup filename.

```
$ docker cp /path/to/openremdump.bak openrem-db:/db_backup/
```

```
$ docker-compose exec db pg_restore --no-privileges --no-owner -U openremuser -d openrem_
→prod /db_backup/openremdump.bak
```

It is normal to get an error about the public schema, for example:

```
pg_restore: while PROCESSING TOC:
pg_restore: from TOC entry 3; 2615 2200 SCHEMA public postgres
pg_restore: error: could not execute query: ERROR:  schema "public" already exists
Command was: CREATE SCHEMA public;

pg_restore: warning: errors ignored on restore: 1
```

Rename the 0.10 upgrade migration file, migrate the database (the steps and fakes are required as it is not a new database), and create the static files:

```
$ docker-compose exec openrem mv remapp/migrations/0001_initial.py.1-0-upgrade remapp/
↪migrations/0001_initial.py
```

```
$ docker-compose exec openrem python manage.py migrate --fake-initial
```

```
$ docker-compose exec openrem python manage.py migrate remapp --fake
```

```
$ docker-compose exec openrem python manage.py makemigrations remapp
```

```
$ docker-compose exec openrem python manage.py migrate
```

```
$ docker-compose exec openrem python manage.py loaddata openskin_safelist.json
```

```
$ docker-compose exec openrem python manage.py collectstatic --noinput --clear
```

Generate translation binary files

```
$ docker-compose exec openrem python manage.py compilemessages
```

The new OpenREM installation should now be ready to be used.

Docker Installation ohne Internetzugriff

OpenREM can be run on a server that is not connected to the internet if required, though access to https://hub.docker.com would make installation and upgrades much easier.

The server will need to have Docker and Docker Compose installed.

Zusammenstellung der Installationsdateien

Auf einem Computer mit Internetzugriff:

- Install Docker - this is required to download the images

- Die Datei https://bitbucket.org/openrem/docker/get/develop.zip herunterladen

- Download the Docker images:

```
$ docker pull openrem/openrem:release-1.0.0b2
```

```
$ docker pull postgres:12.0-alpine
```

```
$ docker pull openrem/nginx
```

```
$ docker pull rabbitmq:3-management-alpine
```

```
$ docker pull openrem/orthanc
```

- Als tar-Dateien speichern:

```
$ docker save -o openrem.tar openrem/openrem:develop
```

```
$ docker save -o openrem-postgres.tar postgres:12.0-alpine
```

```
$ docker save -o openrem-nginx.tar openrem/nginx
```

```
$ docker save -o openrem-rabbitmq.tar rabbitmq:3-management-alpine
```

```
$ docker save -o openrem-orthanc.tar openrem/orthanc
```

Wenn sowohl der Computer mit Internetzugriff als auch der Ziel-Server mit Linux oder MacOS laufen, können die Images mittels gzip verkleinert werden, z.B.:

```
$ docker save openrem/openrem:develop | gzip > openrem.tar.gz
```

Copy all the tar files and the zip file to the server where OpenREM is to be installed.

Load the docker images

On the server where OpenREM is to be installed, in the folder containing the Docker images:

```
$ docker load -i openrem.tar
```

```
$ docker load -i openrem-postgres.tar
```

```
$ docker load -i openrem-nginx.tar
```

```
$ docker load -i openrem-rabbitmq.tar
```

```
$ docker load -i openrem-orthanc.tar
```

If you have compressed the images with gzip the command is the same but with the `.gz` suffix, for example:

```
$ docker load -i openrem.tar.gz
```

Überprüfen ob die folgenden Images geladen sind:

```
$ docker images
```

Continue to *Install*

1.1.2 Native install on Linux

A native installation on Linux requires Python, a webserver (eg Nginx) a database (ideally PostgreSQL) and a DICOM server (ideally Orthanc) to be installed, with OpenREM and all the other dependencies being installed via Pip.

Existing installations of OpenREM 0.10 can be upgraded, but this release requires a different version of Python to the older releases, and some services that were previously required are no longer needed. Full upgrade instructions are provided, based on an Ubuntu Server installation.

Native Linux install

Document not ready for translation

This install is based on Ubuntu 22.04 using:

- Python 3.10 running in a virtualenv
- Datenbank: PostgreSQL
- DICOM Store SCP: Orthanc running on port 104
- Webserver: NGINX with Gunicorn
- All OpenREM files in `/var/dose/` with group owner of `openrem`
- Collects any Physics (QA) images and zips them

The instructions should work for Ubuntu 20.04 too, references to jammy will be focal instead.

There are various commands and paths that reference the Python version 3.10 in these instructions. If you are using Python 3.8 or Python 3.9 then these will need to be modified accordingly.

If you are upgrading an existing installation to a new Linux server, go to the *Upgrading to a new Linux server* docs first.

If you are installing OpenREM on a Linux server with limited internet access, go to the *Offline installation or upgrade* docs.

If you are installing on a different Linux OS you can adapt these instructions or consider using a *Docker install* instead.

Initial prep

Install apt packages

Apt sources

We will need the `universe` repository enabled. Check first:

```
$ less /etc/apt/sources.list
```

Suchen nach:

```
deb http://archive.ubuntu.com/ubuntu/ jammy universe
deb http://archive.ubuntu.com/ubuntu/ jammy-updates universe
```

If these two lines are not there or are commented out (line starts with a #), add them in or remove the # (`sudo nano /etc/apt/sources.list`).

```
$ sudo -- sh -c 'apt update && apt upgrade'
```

```
$ sudo apt install acl python3.10 python3.10-dev python3.10-distutils python3.10-venv
→python3-pip \
postgresql nginx orthanc dcmtk default-jre zip gettext
```

Redis

Redis is used to temporarily store the background tasks.

```
$ sudo apt install lsb-release
```

```
$ curl -fsSL https://packages.redis.io/gpg | sudo gpg --dearmor -o /usr/share/keyrings/
→redis-archive-keyring.gpg
$ echo "deb [signed-by=/usr/share/keyrings/redis-archive-keyring.gpg] https://packages.
→redis.io/deb $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/redis.list
$ sudo apt-get update
$ sudo apt-get install redis
```

Folders and permissions

Groups

Now create new group openrem and add your user to it ($USER will automatically substitute for the user you are running as):

```
$ sudo groupadd openrem
$ sudo adduser $USER openrem
```

Add orthanc and www-data users to openrem group:

```
$ sudo -- sh -c 'adduser orthanc openrem && adduser www-data openrem'
```

Bemerkung: Zu einem späteren Zeitpunkt kann ein zweiter Administrator der openrem Gruppe auf die gleiche Weise hinzugefügt werden.

Folders

Create the folders we need, and set the permissions. The ‚sticky' group setting and the access control list setting (setfacl) below will enable both orthanc user and www-data user as well as you and your colleagues to write to the logs and access the ‚Physics' images etc:

```
$ sudo -- sh -c 'mkdir /var/dose && chmod 775 /var/dose'
```

```
$ sudo chown $USER:openrem /var/dose
```

```
$ cd /var/dose
```

```
$ mkdir {log,media,pixelmed,static,veopenrem3}
```

```
$ mkdir -p orthanc/dicom && mkdir -p orthanc/physics
```

```
$ sudo chown -R $USER:openrem /var/dose/*
```

```
$ sudo chmod -R g+s /var/dose/*
```

Find the `uid` of your user and the `gid` of the `openrem` group:

```
$ id
$ getent group openrem
```

Take note of the `uid` number and the `gid` in the third field of the group information and use it in the next command, replacing `1001` (user `uid`) and `1002` (`openrem` group `gid`) as appropriate:

```
$ sudo setfacl -PRdm u:1001:rwx,g:1002:rwx,o::r /var/dose/
```

Pixelmed download

```
$ cd /var/dose/pixelmed
$ wget http://www.dclunie.com/pixelmed/software/webstart/pixelmed.jar
```

Create the virtualenv

Eine virtualenv in dem erstellten Ordner erzeugen (lokale Python Umgebung):

```
$ python3.10 -m venv /var/dose/veopenrem3
```

Die virtualenv aktivieren

Activate the virtualenv (note the `.` – you can also use the word `source`):

```
$ . /var/dose/veopenrem3/bin/activate
```

Install Python packages

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

```
$ pip install openrem==1.0.0b2
```

Konfiguration von Datenbank und OpenREM

Setup PostgreSQL database

Create a postgres user, and create the database. You will be asked to enter a new password (twice). This will be needed when configuring the `local_settings.py` file later:

```
$ sudo -u postgres createuser -P openremuser
```

```
$ sudo -u postgres createdb -T template1 -O openremuser -E 'UTF8' openremdb
```

For upgrades use a different template

If this is an upgrade to a new Linux server and not a new install, use `template0` instead:

```
$ sudo -u postgres createdb -T template0 -O openremuser -E 'UTF8' openremdb
```

Update the PostgreSQL client authentication configuration. Add the following line anywhere near the bottom of the file, for example in the gap before `# DO NOT DISABLE` or anywhere in the table that follows. The number of spaces between each word is not important (one or more). If you are not using PostgreSQL 14 then substitute the version number in the file path.

```
$ sudo nano /etc/postgresql/14/main/pg_hba.conf
```

```
local   all      openremuser                    md5
```

Reload postgres:

```
$ sudo systemctl reload postgresql
```

OpenREM konfigurieren

Navigate to the Python openrem folder and copy the example `local_settings.py` and `wsgi.py` files to remove the `.linux` and `.example` suffixes:

```
$ cd /var/dose/veopenrem3/lib/python3.10/site-packages/openrem/
$ cp openremproject/local_settings.py{.linux,}
$ cp openremproject/wsgi.py{.example,}
```

Edit `local_settings.py` as needed - make sure you change the PASSWORD, the SECRET_KEY (to anything, just change it), the ALLOWED_HOSTS list, regionalisation settings and the EMAIL configuration. You can modify the email settings later if necessary. Some settings are not shown here but are documented in the settings file or elsewhere in the docs. For details on the final variable see *Systems where Device Observer UID is not static*.

Upgrading to a new server

If you are upgrading to a new Linux server, review the `local_settings.py` file from the old server to copy over the NAME, USER and PASSWORD, ALLOWED_HOSTS list and the EMAIL configuration, and check all the other settings. Change the SECRET_KEY from the default, but it doesn't have to match the one on the old server. For details on the final variable see *Systems where Device Observer UID is not static*.

```
$ nano openremproject/local_settings.py
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'openremdb',
        'USER': 'openremuser',
        'PASSWORD': 'mysecretpassword',      # This is the password you set earlier
        'HOST': '',
        'PORT': '',
    }
}

MEDIA_ROOT = '/var/dose/media/'

STATIC_ROOT = '/var/dose/static/'
JS_REVERSE_OUTPUT_PATH = os.path.join(STATIC_ROOT, 'js', 'django_reverse')

# Change secret key
SECRET_KEY = 'hmj#)-$smzqk*=wuz9^a46rex30^$_j$rghp+1#y&amp;i+pys5b@$'

# DEBUG mode: leave the hash in place for now, but remove it and the space (so DEBUG
# is at the start of the line) as soon as something doesn't work. Put it back
# when you get it working again.
# DEBUG = True

ALLOWED_HOSTS = [
    # Add the names and IP address of your host, for example:
    'openrem-server',
    'openrem-server.ad.abc.nhs.uk',
    '10.123.213.22',
]

LOG_ROOT = '/var/dose/log'
LOG_FILENAME = os.path.join(LOG_ROOT, 'openrem.log')
QR_FILENAME = os.path.join(LOG_ROOT, 'openrem_qr.log')
EXTRACTOR_FILENAME = os.path.join(LOG_ROOT, 'openrem_extractor.log')

# Removed comment hashes to enable log file rotation:
LOGGING['handlers']['file']['class'] = 'logging.handlers.RotatingFileHandler'
LOGGING['handlers']['file']['maxBytes'] = 10 * 1024 * 1024  # 10*1024*1024 = 10 MB
LOGGING['handlers']['file']['backupCount'] = 5  # number of log files to keep before␣
→deleting the oldest one
LOGGING['handlers']['qr_file']['class'] = 'logging.handlers.RotatingFileHandler'
LOGGING['handlers']['qr_file']['maxBytes'] = 10 * 1024 * 1024  # 10*1024*1024 = 10 MB
LOGGING['handlers']['qr_file']['backupCount'] = 5  # number of log files to keep before␣
→deleting the oldest one
LOGGING['handlers']['extractor_file']['class'] = 'logging.handlers.RotatingFileHandler'
LOGGING['handlers']['extractor_file']['maxBytes'] = 10 * 1024 * 1024  # 10*1024*1024 =␣
→10 MB
LOGGING['handlers']['extractor_file']['backupCount'] = 5  # number of log files to keep␣
→before deleting the oldest one
```

```
# Regionalisation settings
#   Date format for exporting data to Excel xlsx files.
#   Default in OpenREM is dd/mm/yyyy. Override it by uncommenting and customising below;
→a full list of codes is
#   available at https://msdn.microsoft.com/en-us/library/ee634398.aspx.
# XLSX_DATE = 'mm/dd/yyyy'
#   Local time zone for this installation. Choices can be found here:
#   http://en.wikipedia.org/wiki/List_of_tz_zones_by_name
#   although not all choices may be available on all operating systems.
#   In a Windows environment this must be set to your system time zone.
TIME_ZONE = 'Europe/London'
#   Language code for this installation. All choices can be found here:
#   http://www.i18nguy.com/unicode/language-identifiers.html
LANGUAGE_CODE = 'en-us'


DCMTK_PATH = '/usr/bin'
DCMCONV = os.path.join(DCMTK_PATH, 'dcmconv')
DCMMKDIR = os.path.join(DCMTK_PATH, 'dcmmkdir')
JAVA_EXE = '/usr/bin/java'
JAVA_OPTIONS = '-Xms256m -Xmx512m -Xss1m -cp'
PIXELMED_JAR = '/var/dose/pixelmed/pixelmed.jar'
PIXELMED_JAR_OPTIONS = '-Djava.awt.headless=true com.pixelmed.doseocr.OCR -'


# E-mail server settings - see https://docs.djangoproject.com/en/2.2/topics/email/
EMAIL_HOST = 'localhost'
EMAIL_PORT = 25
EMAIL_HOST_USER = ''
EMAIL_HOST_PASSWORD = ''
EMAIL_USE_TLS = 0          # Use 0 for False, 1 for True
EMAIL_USE_SSL = 0          # Use 0 for False, 1 for True
EMAIL_DOSE_ALERT_SENDER = 'your.alert@email.address'
EMAIL_OPENREM_URL = 'http://your.openrem.server'


IGNORE_DEVICE_OBSERVER_UID_FOR_THESE_MODELS = ['GE OEC Fluorostar']
```

Jetzt die Datenbank erzeugen. Stellen Sie sicher, dass Sie sich noch im openrem python Ordner befinden und das virtualenv aktiv ist – die Eingabeaufforderung sieht wie folgt aus

```
(veopenrem3)username@hostname:/var/dose/veopenrem3/lib/python3.10/site-packages/openrem/$
```

Otherwise see *Die virtualenv aktivieren* and navigate back to that folder.

Upgrading to a new server

If you are upgrading to a new Linux server, use these additional commands before continuing with those below:

```
$ mv remapp/migrations/0001_initial.py{.1-0-upgrade,}
```

Import the database - update the path to the database backup file you copied from the old server:

```
$ pg_restore --no-privileges --no-owner -U openremuser -d openremdb /path/to/pre-1-0-
→upgrade-dump.bak
```

Migrate the database:

```
$ python manage.py migrate --fake-initial
```

```
$ python manage.py migrate remapp --fake
```

```
$ python manage.py makemigrations remapp
$ python manage.py migrate
$ python manage.py loaddata openskin_safelist.json
$ python manage.py collectstatic --no-input --clear
$ python manage.py compilemessages
$ python manage.py createsuperuser
```

Webserver

Configure NGINX and Gunicorn

Copy in the OpenREM site config file

```
$ cd /var/dose/veopenrem3/lib/python3.10/site-packages/openrem/
$ sudo cp sample-config/openrem-server /etc/nginx/sites-available/openrem-server
```

Bemerkung: Content of NGINX config file:

```
server {
    listen 80;
    server_name openrem-server;

    location /static {
        alias /var/dose/static;
    }

    location / {
        proxy_pass http://unix:/tmp/openrem-server.socket;
        proxy_set_header Host $host;
        proxy_read_timeout 300s;
    }
}
```

Remove the default config and make ours active:

```
$ sudo rm /etc/nginx/sites-enabled/default
```

```
$ sudo ln -s /etc/nginx/sites-available/openrem-server /etc/nginx/sites-enabled/openrem-
↪server
```

Copy the Gunicorn systemd service file into place:

```
$ cd /var/dose/veopenrem3/lib/python3.10/site-packages/openrem/
$ sudo cp sample-config/openrem-gunicorn.service /etc/systemd/system/openrem-gunicorn.
↪service
```

Bemerkung: Content of systemd file:

```
[Unit]
Description=Gunicorn server for OpenREM

[Service]
Restart=on-failure
User=www-data
WorkingDirectory=/var/dose/veopenrem3/lib/python3.10/site-packages/openrem

ExecStart=/var/dose/veopenrem3/bin/gunicorn \
    --bind unix:/tmp/openrem-server.socket \
    openremproject.wsgi:application --timeout 300

[Install]
WantedBy=multi-user.target
```

Copy the task queue consumer systemd service file into place:

```
$ cd /var/dose/veopenrem3/lib/python3.10/site-packages/openrem/
$ sudo cp sample-config/openrem-consumer.service /etc/systemd/system/openrem-consumer.
↪service
```

Bemerkung: Content of systemd file:

```
[Unit]
Description=Huey consumer for OpenREM

[Service]
Restart=on-failure
User=www-data
WorkingDirectory=/var/dose/veopenrem3/lib/python3.10/site-packages/openrem

ExecStart=/var/dose/veopenrem3/bin/python \
    manage.py run_huey

[Install]
WantedBy=multi-user.target
```

Die neue systemd Konfiguration laden:

```
$ sudo systemctl daemon-reload
```

Set the new Gunicorn and consumer services to start on boot:

```
$ sudo systemctl enable openrem-gunicorn.service
$ sudo systemctl enable redis-server.service
$ sudo systemctl enable openrem-consumer.service
```

Start the Gunicorn and consumer services, and restart the NGINX service:

```
$ sudo -- sh -c 'systemctl start openrem-gunicorn.service && systemctl start redis-
↪server.service && systemctl start openrem-consumer.service && systemctl restart nginx.
↪service'
```

Test the webserver

Jetzt sollte der Zugriff auf den OpenREM Server von anderen Rechner möglich sein.

Ob NGINX und Gunicorn bereit sind kann mit den folgenden zwei Befehlen überprüft werden:

```
$ sudo systemctl status openrem-gunicorn.service
```

```
$ sudo systemctl status nginx.service
```

DICOM Store SCP

Copy the Lua file to the Orthanc folder. This will control how we process the incoming DICOM objects.

```
$ cd /var/dose/veopenrem3/lib/python3.10/site-packages/openrem/
$ cp sample-config/openrem_orthanc_config_linux.lua /var/dose/orthanc/
```

Edit the Orthanc Lua configuration options:

```
$ nano /var/dose/orthanc/openrem_orthanc_config_linux.lua
```

Set `use_physics_filtering` to true if you want Orthanc to keep physics test studies, and have it put them in the `/var/dose/orthanc/physics/` folder. Set it to `false` to disable this feature. Add names or IDs to `physics_to_keep` as a comma separated list.

```lua
-- Set this to true if you want Orthanc to keep physics test studies, and have it
-- put them in the physics_to_keep_folder. Set it to false to disable this feature
local use_physics_filtering = true

-- A list to check against patient name and ID to see if the images should be kept.
-- Orthanc will put anything that matches this in the physics_to_keep_folder.
local physics_to_keep = {'physics'}
```

Lists of things to ignore. Orthanc will ignore anything matching the content of these comma separated lists; they will not be imported into OpenREM.

```lua
-- Lists of things to ignore. Orthanc will ignore anything matching the content of
-- these lists: they will not be imported into OpenREM.
local manufacturers_to_ignore = {'Faxitron X-Ray LLC', 'Gendex-KaVo'}
local model_names_to_ignore = {'CR 85', 'CR 75', 'CR 35', 'CR 25', 'ADC_5146', 'CR975'}
local station_names_to_ignore = {'CR85 Main', 'CR75 Main'}
```

```lua
local software_versions_to_ignore = {'VixWin Platinum v3.3'}
local device_serial_numbers_to_ignore = {'SCB1312016'}
```

Funktion für das extrahieren der Dosisinformation aus älteren Toshiba und GE Geräte ein- oder ausschalten und für wel-
chen Gerätetyp dies erfolgen soll. Jedes System sollte in der Form {`'Manufacturer'`, `'Model name'`} aufgeführt
werden, wie das nachstehende Beispiel zeigt:

```lua
-- Set this to true if you want to use the OpenREM Toshiba CT extractor. Set it to
-- false to disable this feature.
local use_toshiba_ct_extractor = true

-- A list of CT make and model pairs that are known to have worked with the Toshiba CT
↪extractor.
-- You can add to this list, but you will need to verify that the dose data created
↪matches what you expect.
local toshiba_extractor_systems = {
        {'Toshiba', 'Aquilion'},
        {'GE Medical Systems', 'Discovery STE'},
}
```

Edit the Orthanc configuration:

```
$ sudo nano /etc/orthanc/orthanc.json
```

Dieses Lua Skript zur Othanc Konfiguration hinzufügen:

```
// List of paths to the custom Lua scripts that are to be loaded
// into this instance of Orthanc
"LuaScripts" : [
"/var/dose/orthanc/openrem_orthanc_config_linux.lua"
],
```

Set the AE Title and port:

```
// The DICOM Application Entity Title
"DicomAet" : "OPENREM",

// The DICOM port
"DicomPort" : 104,
```

Bemerkung: Optionally, you may also like to enable the HTTP server interface for Orthanc (although if the Lua script
is removing all the objects as soon as they are processed, you won't see much!):

```
// Whether remote hosts can connect to the HTTP server
"RemoteAccessAllowed" : true,

// Whether or not the password protection is enabled
"AuthenticationEnabled" : false,
```

To see the Orthanc web interface, go to http://openremserver:8042/ – of course change the server name to that of your
server!

Allow Orthanc to use DICOM port

By default, Orthanc uses port 4242. If you wish to use a lower port, specifically the DICOM port of 104, you will need to give the Orthanc binary special permission to do so:

```
$ sudo setcap CAP_NET_BIND_SERVICE=+eip /usr/sbin/Orthanc
```

Finish off

Restart Orthanc:

```
$ sudo systemctl restart orthanc.service
```

Neue Nutzer, und schneller Zugriff auf den Physik-Ordner

This is for new Linux users; for new OpenREM users, refer to *Configure the settings*

If you left `local use_physics_filtering = true` in the Orthanc configuration, you might like to give your colleagues a quick method of accessing the physics folder from their home folder. Then if they use a program like WinSCP it is easy to find and copy the QA images to another (Windows) computer on the network. WinSCP can also be run directly from a USB stick if you are unable to install software :-)

Add the new user (replace `newusername` as appropriate):

```
$ sudo adduser newusername
```

Then add the new user to the *openrem* group (again, replace the user name):

```
$ sudo adduser newusername openrem
```

Now add a ‚sym-link‘ to the new users home directory (again, replace the user name):

```
$ sudo ln -sT /var/dose/orthanc/physics /home/newusername/physicsimages
```

Der neue Nutzer sollte, sobald er eingeloggt ist, jetzt durch Klicken auf `physicsimages` Link auf den Physik-Ordner zugreifen können, die ZIP Dateien und Ordner anzeigen, kopieren und löschen können.

Asciinema demo of this install

Link to asciinema demo of this install

Upgrading a native Linux install

These instructions assume a configuration similar to the ‚One page complete Ubuntu install‘ provided with release 0.8.1 and later. If you are running an older distribution, consider upgrading the operating system or migrating the service to a new host. The test system for these upgrade instructions was upgraded from 18.04 to 20.04 and then 22.04 before the OpenREM upgrade was started. If you are using a different distribution or have set up your system differently, it might be better to start afresh following or adapting the *Upgrading to a new Linux server* docs instead.

If upgrading to a new host, follow the *Upgrading to a new Linux server* docs.

This release will run on Python 3.8 or 3.9, but Python 3.10 is recommended. If a different release of Python is being used, substitute 3.10 for that version where necessary below.

If you are upgrading OpenREM on a Linux server with limited internet access, go to the *Offline installation or upgrade* docs.

- **Upgrades from 0.9.1 or earlier should review** *Upgrade to OpenREM 0.10.0 from 0.7.3 or later* first. Upgrading to 1.0 is only possible from 0.10.0.

Preparation

Back up the database - you will need the password for `openremuser` that will be in your `local_settings.py` file. You'll need this file again later so open it in a different window:

```
$ less /var/dose/veopenrem/lib/python2.7/site-packages/openrem/openremproject/local_
→settings.py
```

Backup the database, in the main window:

```
$ pg_dump -U openremuser -d openremdb -F c -f pre-1-0-upgrade-dump.bak
```

Stop any Celery workers, Flower, RabbitMQ, Gunicorn, NGINX, and Orthanc (OpenREM service names will be reversed if they weren't changed with the 0.9.1 upgrade):

```
$ sudo systemctl stop openrem-celery
$ sudo systemctl stop openrem-flower
$ sudo systemctl stop openrem-gunicorn
$ sudo systemctl stop rabbitmq-server
$ sudo systemctl stop nginx
$ sudo systemctl stop orthanc
```

Update apt and install any updates:

```
$ sudo -- sh -c 'apt update && apt upgrade'
```

Install Python 3.10 and other packages:

```
$ sudo apt install acl python3.10 python3.10-dev python3.10-distutils python3.10-venv
→python3-pip \
postgresql nginx orthanc dcmtk default-jre zip gettext
```

Reset the permissions for the `/var/dose` folder:

```
$ sudo chmod -R 775 /var/dose
$ sudo chown -R $USER:openrem /var/dose
$ sudo chmod -R g+s /var/dose/*
```

Now find the `uid` of your user and the `gid` of the `openrem` group:

```
$ id
$ getent group openrem
```

Take note of the `uid` number and the `gid` in the third field of the group information and use it in the next command, replacing `1001` (user `uid`) and `1002` (`openrem` group `gid`) as appropriate:

```
$ sudo setfacl -PRdm u:1001:rwx,g:1002:rwx,o::r /var/dose/
```

What are we doing with the permissions?

These settings enable the web server user `www-data`, the DICOM server user `orthanc` and the OpenREM server users (you and your colleagues) to all read, write and execute the OpenREM files. The `setfacl` command relies on Access Control Lists being available on your system - they are usually enabled on ext4 and can be enabled on others. See *Neue Nutzer, und schneller Zugriff auf den Physik-Ordner* for adding colleagues access to the Linux folders.

Create a new Python virtual environment:

```
$ python3.10 -m venv /var/dose/veopenrem3
```

Activate the virtualenv:

```
$ . /var/dose/veopenrem3/bin/activate
```

Install the new version of OpenREM

Ensure the new virtualenv is active — prompt will look like

```
(veopenrem3)username@hostname:~$
```

Upgrade Pip and install OpenREM

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

```
$ pip install openrem==1.0.0b1
```

Configure the local_settings.py file

Navigate to the Python openrem folder and copy the example `local_settings.py` and `wsgi.py` files to remove the `.linux` and `.example` suffixes:

```
$ cd /var/dose/veopenrem3/lib/python3.10/site-packages/openrem/
$ cp openremproject/local_settings.py{.linux,}
$ cp openremproject/wsgi.py{.example,}
```

Review the old `local_settings.py` file that was opened earlier - see the first part of the Preparation section. Edit the new `local_settings.py` as needed - make sure you update the database `NAME`, `USER` and `PASSWORD`, the `ALLOWED_HOSTS` list and the `EMAIL` configuration and check all the other settings. Change the `SECRET_KEY` from the default:

```
$ nano openremproject/local_settings.py
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'openremdb',
```

```
        'USER': 'openremuser',
        'PASSWORD': 'mysecretpassword',      # This is the password you set earlier
        'HOST': '',
        'PORT': '',
    }
}

MEDIA_ROOT = '/var/dose/media/'

STATIC_ROOT = '/var/dose/static/'

# Change secret key
SECRET_KEY = 'hmj#)-$smzqk*=wuz9^a46rex30^$_j$rghp+1#y&amp;i+pys5b@$'

# DEBUG mode: leave the hash in place for now, but remove it and the space (so DEBUG
# is at the start of the line) as soon as something doesn't work. Put it back
# when you get it working again.
# DEBUG = True

ALLOWED_HOSTS = [
    # Add the names and IP address of your host, for example:
    'openrem-server',
    'openrem-server.ad.abc.nhs.uk',
    '10.123.213.22',
]

LOG_ROOT = '/var/dose/log'
LOG_FILENAME = os.path.join(LOG_ROOT, 'openrem.log')
QR_FILENAME = os.path.join(LOG_ROOT, 'openrem_qr.log')
EXTRACTOR_FILENAME = os.path.join(LOG_ROOT, 'openrem_extractor.log')

# Removed comment hashes to enable log file rotation:
LOGGING['handlers']['file']['class'] = 'logging.handlers.RotatingFileHandler'
LOGGING['handlers']['file']['maxBytes'] = 10 * 1024 * 1024  # 10*1024*1024 = 10 MB
LOGGING['handlers']['file']['backupCount'] = 5  # number of log files to keep before
→deleting the oldest one
LOGGING['handlers']['qr_file']['class'] = 'logging.handlers.RotatingFileHandler'
LOGGING['handlers']['qr_file']['maxBytes'] = 10 * 1024 * 1024  # 10*1024*1024 = 10 MB
LOGGING['handlers']['qr_file']['backupCount'] = 5  # number of log files to keep before
→deleting the oldest one
LOGGING['handlers']['extractor_file']['class'] = 'logging.handlers.RotatingFileHandler'
LOGGING['handlers']['extractor_file']['maxBytes'] = 10 * 1024 * 1024  # 10*1024*1024 =
→10 MB
LOGGING['handlers']['extractor_file']['backupCount'] = 5  # number of log files to keep
→before deleting the oldest one

# Regionalisation settings
#   Date format for exporting data to Excel xlsx files.
#   Default in OpenREM is dd/mm/yyyy. Override it by uncommenting and customising below;
→a full list of codes is
#   available at https://msdn.microsoft.com/en-us/library/ee634398.aspx.
# XLSX_DATE = 'mm/dd/yyyy'
```

```
#   Local time zone for this installation. Choices can be found here:
#   http://en.wikipedia.org/wiki/List_of_tz_zones_by_name
#   although not all choices may be available on all operating systems.
#   In a Windows environment this must be set to your system time zone.
TIME_ZONE = 'Europe/London'
#   Language code for this installation. All choices can be found here:
#   http://www.i18nguy.com/unicode/language-identifiers.html
LANGUAGE_CODE = 'en-us'


DCMTK_PATH = '/usr/bin'
DCMCONV = os.path.join(DCMTK_PATH, 'dcmconv')
DCMMKDIR = os.path.join(DCMTK_PATH, 'dcmmkdir')
JAVA_EXE = '/usr/bin/java'
JAVA_OPTIONS = '-Xms256m -Xmx512m -Xss1m -cp'
PIXELMED_JAR = '/var/dose/pixelmed/pixelmed.jar'
PIXELMED_JAR_OPTIONS = '-Djava.awt.headless=true com.pixelmed.doseocr.OCR -'


# E-mail server settings - see https://docs.djangoproject.com/en/2.2/topics/email/
EMAIL_HOST = 'localhost'
EMAIL_PORT = 25
EMAIL_HOST_USER = ''
EMAIL_HOST_PASSWORD = ''
EMAIL_USE_TLS = 0          # Use 0 for False, 1 for True
EMAIL_USE_SSL = 0          # Use 0 for False, 1 for True
EMAIL_DOSE_ALERT_SENDER = 'your.alert@email.address'
EMAIL_OPENREM_URL = 'http://your.openrem.server'
```

Migrate the database

In a shell/command window, move into the openrem folder:

```
$ cd /var/dose/veopenrem3/lib/python3.10/site-packages/openrem/
```

Prepare the migrations folder:

- Rename `0001_initial.py.1-0-upgrade` to `0001_initial.py`

```
$ mv remapp/migrations/0001_initial.py{.1-0-upgrade,}
```

Migrate the database:

```
$ python manage.py migrate --fake-initial
```

```
$ python manage.py migrate remapp --fake
```

```
$ python manage.py makemigrations remapp
```

Rename questions

There will be some questions about fields being renamed - answer N to all of them.

```
$ python manage.py migrate
```

```
$ python manage.py loaddata openskin_safelist.json
```

Update static files and translations

```
$ python manage.py collectstatic --clear
```

Warning about deleting all files

You will get a warning about all files in the static files location being deleted. As long as the folder is correct, type `yes` to continue.

Virtual directory users

If you are running your website in a virtual directory, you also have to update the reverse.js file. To get the file in the correct path, take care that you insert just after the declaration of `STATIC_ROOT` the following line in your `local_settings.py` (see also the sample `local_settings.py.example`):

```
JS_REVERSE_OUTPUT_PATH = os.path.join(STATIC_ROOT, 'js', 'django_reverse')
```

To update the reverse.js file execute the following command:

```
$ python manage.py collectstatic_js_reverse
```

See *Running the OpenREM website in a virtual directory* for more details.

Generate translation binary files

```
$ python manage.py compilemessages
```

Update all the services configurations

Edit the Gunicorn systemd file `WorkingDirectory` and `ExecStart`:

```
$ sudo nano /etc/systemd/system/openrem-gunicorn.service
```

```
WorkingDirectory=/var/dose/veopenrem3/lib/python3.10/site-packages/openrem

ExecStart=/var/dose/veopenrem3/bin/gunicorn \
    --bind unix:/tmp/openrem-server.socket \
    openremproject.wsgi:application --timeout 300 --workers 4
```

Celery, Flower and RabbitMQ are no longer required for this release, so their Systemd control files can be disabled, and RabbitMQ can be removed (assuming it is not in use for any other services on this server):

```
$ sudo systemctl disable openrem-celery.service
$ sudo systemctl disable openrem-flower.service
```

```
$ sudo apt remove rabbitmq-server
$ sudo apt purge rabbitmq-server
```

Reload systemd and restart the services

```
$ sudo systemctl daemon-reload
```

Start and check Gunicorn:

```
$ sudo systemctl start openrem-gunicorn.service
$ sudo systemctl status openrem-gunicorn.service
```

Start and check NGINX:

```
$ sudo systemctl start nginx.service
$ sudo systemctl status nginx.service
```

Start and check Orthanc:

```
$ sudo systemctl start orthanc.service
$ sudo systemctl status orthanc.service
```

Registered Users error

If Orthanc fails to start, check the Orthanc log file:

```
$ sudo less /var/log/orthanc/Orthanc.log
```

If there is an error: `Bad file format: The configuration section "RegisteredUsers" is defined in 2 different configuration files` this might be due to changes in the installed version of Orthanc.

Edit the main Orthanc configuration file to remove the setting, as it is now in a `credentials.json` configuration file.

```
$ sudo nano /etc/orthanc/orthanc.json
```

Remove the `RegisteredUsers` setting and try again:

```
$ sudo systemctl start orthanc.service
$ sudo systemctl status orthanc.service
```

If there is still an issue, check the log again. If the problem this time is due to the `TCP port of the DICOM server`, you might need to give it permission again:

```
$ sudo setcap CAP_NET_BIND_SERVICE=+eip /usr/sbin/Orthanc
```

And restart Orthanc once more.

Test the webserver

You should now be able to browse to the web interface of your upgraded OpenREM system and have a look around.

Update the DICOM Store settings

Log in to the web interface, and navigate to `Config`, `DICOM networking`.

The remote nodes should be correct from the old system, but the DICOM Store SCP settings will need updating. Modify the store, and add the hostname `localhost`.

After you have clicked `Submit`, the status page should show the server is alive. If it isn't, go and check the status of Orthanc again (we may have checked it too quickly before).

Upgrading to a new Linux server

If OpenREM has been running on an older Linux distribution, or you wish to move to Linux to host OpenREM and don't want to use Docker, these instructions will guide you through upgrading an existing database to a new server.

- **Upgrades from 0.9.1 or earlier should review** *Upgrade to OpenREM 0.10.0 from 0.7.3 or later* first. Upgrading to 1.0 is only possible from 0.10.0.

This install is based on Ubuntu 22.04 using:

- Python 3.10 running in a virtualenv
- Datenbank: PostgreSQL
- DICOM Store SCP: Orthanc running on port 104
- Webserver: NGINX with Gunicorn
- All OpenREM files in `/var/dose/` with group owner of `openrem`
- Collects any Physics (QA) images and zips them

Get the local_settings.py file

Get `local_settings.py` file from the old server - it should be in one of these locations:

- Ubuntu 'One page install': `/var/dose/veopenrem/lib/python2.7/site-packages/openrem/openremproject/local_settings.py`
- Ubuntu linux: `/usr/local/lib/python2.7/dist-packages/openrem/openremproject/local_settings.py`
- Other linux: `/usr/lib/python2.7/site-packages/openrem/openremproject/local_settings.py`
- Linux virtualenv: `vitualenvfolder/lib/python2.7/site-packages/openrem/openremproject/local_settings.py`
- Windows: `C:\Python27\Lib\site-packages\openrem\openremproject\local_settings.py`
- Windows virtualenv: `virtualenvfolder\Lib\site-packages\openrem\openremproject\local_settings.py`

Export the database

Export the old database on the old server - you will need the password for `openremuser` that will be in your `local_settings.py` file, and you might need to change the `openremuser` database username and the `openremdb` name of the database:

```
$ pg_dump -U openremuser -d openremdb -F c -f pre-1-0-upgrade-dump.bak
```

Transfer the files

Copy these two files to your new server.

Continue on the new server

Now follow the *Native Linux install* instructions looking out for the additional steps for upgrading to a new Linux server.

Offline installation or upgrade

In order to install or upgrade OpenREM on a Windows server that does not have access to the internet you will need to download all the packages and dependencies on another computer and copy them across.

If you have trouble when installing the Python packages on Windows due to incorrect architecture, you may need to either download on a Windows system similar to the server (matching 32-bit/64-bit), or to download the files from http://www.lfd.uci.edu/~gohlke/pythonlibs/ instead. Alternatively there are ways to tell `pip` to download binary packages for specific platforms.

It is expected and highly recommended that server operating systems have access to security updates even when other internet access is blocked.

The instructions that follow are for a Windows server that doesn't have access to the internet. For Linux servers, it is recommended to allow access to the distribution's repositories to install and update the software. It is technically possible to use a local repository mirror/cache, or to download all the packages manually, but this is beyond the scope of these instructions.

An *Docker Installation ohne Internetzugriff* might be easier on an offline Linux server, once Docker and Docker Compose are installed.

On a computer with internet access

Download independent binaries

Download all the software in the *Installing packages* section except IIS:

- Python
- Orthanc
- PostgreSQL
- gettext
- Pixelmed
- dcmtk

- 7Zip

- Notepad++

- WinSW

Download Python packages from PyPI

In a console, navigate to a suitable place and create an empty directory to collect all the packages in, then use `pip` to download them all - Python 3 (including Pip) will need to be installed on the computer with internet access to download the packages, ideally Python 3.10:

```
C:\Users\me\Desktop> mkdir openremfiles
C:\Users\me\Desktop> pip3 download -d openremfiles pip
C:\Users\me\Desktop> pip3 download -d openremfiles openrem==1.0.0b2
C:\Users\me\Desktop> pip3 download -d openremfiles wfastcgi
```

Copy everything to the Server

- Copy this directory plus the binaries to the offline server.

On the server without internet access

Follow the *Native Windows install*, *Upgrading to a new Windows server*, or *Upgrading a native Windows install* instructions, installing the binary packages that were copied across as well as IIS. The **Install OpenREM** section has instructions on how to install OpenREM python packages from the folder you have copied across.

1.1.3 Native install on Windows

A native installation on Windows Server requires Python, a database (ideally PostgreSQL) and a DICOM server (ideally Orthanc) to be installed, with OpenREM and all the other dependencies being installed via Pip. IIS is the recommended webserver to use on Windows.

This installation process can be used with Windows 10 (and probably 11), but this is not advised for production use as Windows 10 and 11 are not designed to be servers.

As for native Linux installs, existing installations of OpenREM 0.10 can be upgraded, but this release requires a different version of Python to the older releases, and some services that were previously required are no longer needed. Full upgrade instructions are provided, based on a Windows Server 2019 installation.

Native Windows install

Document not ready for translation

This install is based on Windows Server 2022 using:

- Python 3.10 running in a virtualenv

- Datenbank: PostgreSQL

- DICOM Store SCP: Orthanc running on port 104

- Webserver: Microsoft IIS running on port 80

- WinSW to run background tasks as services

- Notepad++ for editing files

- Database files stored on D:

- OpenREM files stored on E:

- With Physics (QA) images being collected and zipped for retrieval

The instructions should work for Windows Server 2016 and 2019; and will probably work with Windows 10/11 with some modification. Desktop editions of Windows are not recommended for a production OpenREM install.

If you are upgrading an existing installation to a new Windows server, go to the *Upgrading to a new Windows server* first.

If you are upgrading an existing Windows Server installation in-place, go to *Upgrading a native Windows install* instead.

If you are installing on a server with no internet access, go to *Offline installation or upgrade* to download the packages.

These instructions assume the following disk layout - there is more information about the reasoning in the box below:

- `C:` OS disk

- `D:` Database disk

- `E:` Data disk

Initial prep

Creating folders

Why D: and E: drives?

OpenREM data are stored on drive E: to keep the data away from the operating system drive so that it is easier for building/recreating the server and knowing what needs to be backed up.

For the same reason, we will install PostgreSQL so that the database data are store on drive D: - this makes it possible to provide a different configuration of disk for the database drive, with different backup policies.

However, it is also possible to store all the data on the C: drive if that works better for your installation. In this case, it would be advisable to create a folder C:\OpenREM\ and create all the folders specified below into that folder.

You can also use different drive letters if that works better for your installation. **In both cases paths will need to be modified in the instructions to suite**.

Create the following folders. The instructions here are for a `CMD` window but they can be created in Windows Explorer instead:

```
C:\Users\openrem>D:
D:\>mkdir database
D:\>E:
E:\>mkdir log media pixelmed dcmtk 7-zip static
↪task_queue venv orthanc\dicom orthanc\physics orthanc\storage winsw
```
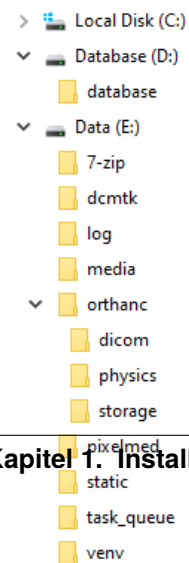
Set permissions

- Right click on the `E:\log` folder and click `Properties`

- In the `Security` tab click `Edit...` and `Add...`

If the server is connected to a domain

If the server is connected to a domain, the `From this location:` will have the name of the domain. Click `Locations...` and choose the name of the server instead of the domain name.
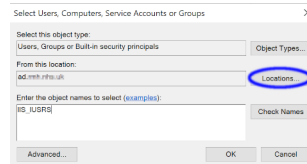
Abb. 2: Figure 2: Set account location

- Enter the object name `IIS_IUSRS` and click `OK`

- Tick the `Modify Allow` to enable read and write permissions

- Click `OK` twice to close the dialogues

- Repeat for the `E:\media` and `E:\task_queue` folders

Installing packages

Python

Download the latest version for Windows from https://www.python.org/downloads/ as long as it is in the 3.10 series. OpenREM v1.0 has not been tested with Python 3.11 yet.

Open the downloaded file to start the installation:

- Customize installation

- Leave all the Optional Features ticked, and click `Next`

- Tick `Install for all users` - this will automatically tick `Precompile standard library`

- `Install`

- Click to `Disable path length limit` - might not be necessary but might be useful!

- `Close`

Orthanc

Download the 64 bit version from https://www.orthanc-server.com/download-windows.php.

The download file might be blocked because it isn't a commonly downloaded executable. Click the `...` menu and select `Keep`. Then click `Show more` and `Keep anyway`.

Open the downloaded file to start the installation:

- Click `Next >`, accept the agreement and `Next >` again.

- Default install location, `Next >`

- Select Orthanc storage directory - `Browse...` to `E:\orthanc\storage`, `OK` and `Next >`

- Click `Next >` for a Full installation

- Start Menu Folder `Next >`

- Ready to Install `Install`

- `Finish`

PostgreSQL

Download the latest version of PostgreSQL from https://www.enterprisedb.com/downloads/postgres-postgresql-downloads - choose the Windows x86-64 version. OpenREM v1.0 has been tested with PostgreSQL v14.5.

Open the downloaded file to start the installation:

- Some Microsoft redistributables will install

- Click `Next >` to start

- Default Installation Directory `Next >`

- All components `Next >`

- Data Directory - browse to `D:\database` then `Select folder` and `Next >`

- Create a password for the `postgres` superuser - you will need this to setup the database with pgAdmin 4 later

- Enter it twice and `Next >`

- Default port `Next >`

- Default Locale `Next >`

- Pre Installation Summary `Next >`

- Ready to Install `Next >` and the installation will begin

- Untick `Launch Stack Builder at exit`

- `Finish`

gettext

Download the 64 bit static version of gettext 0.21 from https://mlocati.github.io/articles/gettext-iconv-windows.html. Use the `.exe` version (software install icon, not the zip icon)

gettext 0.21 and iconv 1.16 - Binaries for Windows

gettext version	libiconv version	Operating system	Flavor	Download
0.21	1.16	32 bit	shared[1]	
0.21	1.16	32 bit	static[2]	
0.21	1.16	64 bit	shared[1]	
0.21	1.16	64 bit	static[2]	

Abb. 3: Figure 3: gettext download page

Open the downloaded file to start the installation:

- Accept the agreement `Next >`
- Default installation directory `Next >`
- Additional Tasks leave both boxes ticked `Next >`
- Ready to Install `Install`
- `Finish`

What is gettext for?

The gettext binary enables the translations to be available to users of the web interface. It is not essential if you don't want the translations to be available.

Pixelmed

Download DoseUtility from from the page http://www.dclunie.com/pixelmed/software/webstart/DoseUtilityUsage.html - find `How to install it (locally)` near the bottom of the page and click the `Windows executable that does not require Java to be installed` link.

How to install it (locally)

If for some reason you do not want to start the application using Java Web Start, but instead want to download it and install it, several versions are available:

- Windows executable that does not require Java to be installed [approx. 45 MB] (includes its own JRE, internationalized fonts, and JIIO libraries)
- Windows executable that requires Java 1.7 or later to already be installed [approx. 3.9 MB] (includes its own JIIO libraries, since these are often not installed)
- MacOS executable that requires Java 1.7 or later to already be installed [approx. 2.1 MB] (includes pure Java JIIO libraries for limited decompression support)

Abb. 4: Figure 4: Pixelmed download page

- Open the downloaded zip file and open a new file browser at `E:\pixelmed`
- Drag the contents of the zip file to the `pixelmed` folder

DCMTK

Download from https://dcmtk.org/dcmtk.php.en - look for the `DCMTK executable binaries` section, and download the 64 bit DLL build for Windows.
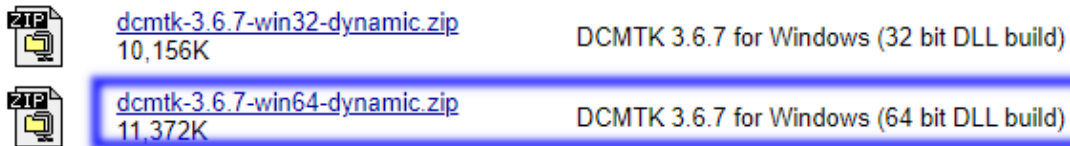
Abb. 5: Figure 5: DCMTK download page

- Open the downloaded zip file and open a new file browser at `E:\dcmtk`

- Drag the contents of the dcmtk-3.x.x-win64-dynamic folder in the zip file to the `dcmtk` folder

- You should end up with `E:\dcmtk\bin\` etc

7Zip

Download the 64-bit x64 exe file from https://www.7-zip.org/

- Type, or click on the `...` to browse to `E:\7-zip\`

- `Install`

- `Close`

WinSW

Download the 64-bit x64 exe file from https://github.com/winsw/winsw/releases/tag/v2.12.0

- Open a new file browser at `E:\winsw`

- Drag the exe file to the `winsw` folder

- Rename the exe file from `WinSW-x64` to `WinSW`

Notepad++

Download the latest version of Notepad++ from https://notepad-plus-plus.org/downloads/

Open the downloaded file to start the installation:

- Select a language `OK`

- Welcome `Next >`

- License Agreement `I Agree`

- Install Location `Next >`

- Choose Components `Next >`

- `Install`

- `Finish` (you can untick the `Run Notepad++` option, we don't need it yet)

IIS

- Open the Control Panel

- Search for `windows features`

- Select `Turn Windows features on or off`

- Start the wizard `Next >`

- Role-based or feature-based installation `Next >`

- Leave the current server highlighted `Next >`

- Check the `Web Server (IIS)` box

- In the pop-up dialogue for adding IIS Management Console, click `Add Features`

- `Next >`

- Features, `Next >`

- Web Server Role (IIS) `Next >`

- Expand the `Application Development` section

- Check the `CGI` box, `Next >`

- `Install`

- `Close`

You can check the server is running by browsing to http://localhost/ on the server. You should see the default IIS Welcome page. It might not work immediately, check again in a few minutes.

Installing Python packages

Create and activate the virtualenv

Open a `CMD` window:

```
C:\Users\openrem>e:
E:\>py -m venv venv
E:\>venv\Scripts\activate
(venv) E:\>
```

Install OpenREM

Installing on a server with no internet access

Make sure the virtualenv is activated (command line will have the name of the virtualenv as a prefix: `(venv) E:\`), then navigate to where the `openremfiles` directory is that you copied from the computer *with* internet access, eg if it is in your desktop folder:

```
(venv) E:\>c:
(venv) C:\>cd Users\openrem\Desktop
```

Now upgrade `pip` and install OpenREM and its dependencies:

```
(venv) C:\Users\openrem\Desktop>pip install --no-index --find-links=openremfiles --
→upgrade pip
(venv) C:\Users\openrem\Desktop>pip install --no-index --find-links=openremfiles openrem
```

```
(venv) E:\>pip install --upgrade pip
(venv) E:\>pip install openrem==1.0.0b2
(venv) E:\>pip install wfastcgi
```

OpenREM configuration and database creation

PostgreSQL database creation

Start pgAdmin 4 - you will need the password you set when installing PostgreSQL

Create user

- Click on `Servers` to expand, enter the password again
- Right click `Login/Group Roles`, `Create`, `Login/Group Role...`
- Name: `openremuser`
- Definition, Password: add a password for the openremuser
- Privileges: activate `Can login?` and `Create database?`
- `Save`

Create database

- Right click `Databases`, `Create`, `Database...`
- Database: `openremdb`
- Owner: `openremuser`
- `Save`

OpenREM konfigurieren

Open the `E:\venv\Lib\site-packages\openrem\openremproject` folder and rename the example `local_settings.py` and `wsgi.py` files to remove the `.windows` and `.example` suffixes. Removing the file name extension will produce a warning to check if you are sure - `Yes`:
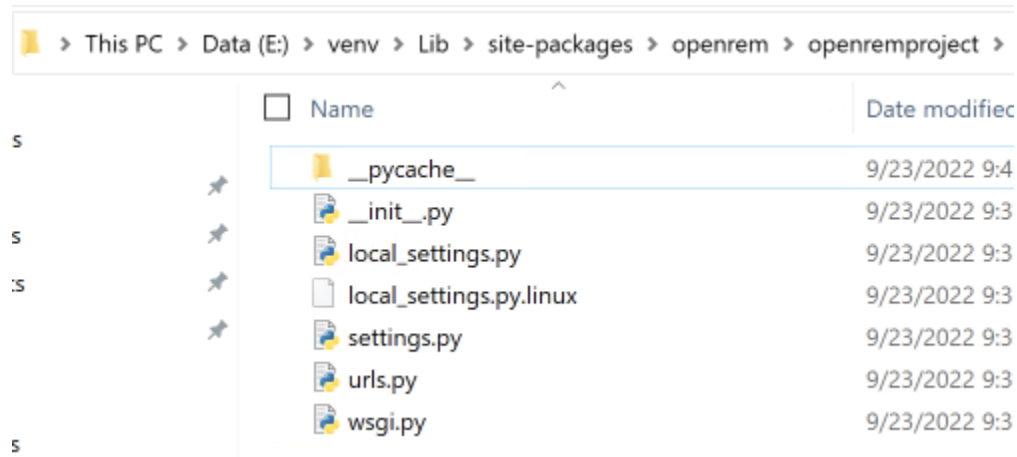
Abb. 6: Figure 6: openremproject folder

Edit `local_settings.py` as needed (right click `Edit with Notepad++`) Make sure you change the `PASSWORD`, the `SECRET_KEY` (to anything, just change it), the `ALLOWED_HOSTS` list, regionalisation settings and the `EMAIL` configuration. You can modify the email settings later if necessary. Some settings are not shown here but are documented in the settings file or elsewhere in the docs. For details on the final variable see *Systems where Device Observer UID is not static*.

Upgrading to a new server

If you are upgrading to a new Linux server, review the `local_settings.py` file from the old server to copy over the `ALLOWED_HOSTS` list and the `EMAIL` configuration, and check all the other settings. Change the `SECRET_KEY` from the default, but it doesn't have to match the one on the old server. The database `NAME`, `USER` and `PASSWORD` will be the ones you created on the new server. For details on the final variable see *Systems where Device Observer UID is not static*.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',  # Add 'postgresql', 'mysql', 'sqlite3'
→or 'oracle'.
        'NAME': 'openremdb',                        # Or path to database file if using
→sqlite3.
        'USER': 'openremuser',                 # Not used with sqlite3.
        'PASSWORD': '',                        # Not used with sqlite3.
        'HOST': '',                            # Set to empty string for localhost.
→Not used with sqlite3.
        'PORT': '',                            # Set to empty string for default. Not
→used with sqlite3.
    }
}

TASK_QUEUE_ROOT = 'E:/task_queue/'
```

```
MEDIA_ROOT = 'E:/media/'


STATIC_ROOT = 'E:/static/'
JS_REVERSE_OUTPUT_PATH = os.path.join(STATIC_ROOT, 'js', 'django_reverse')


# Change secret key
SECRET_KEY = 'hmj#)-$smzqk*=wuz9^a46rex30^$_j$rghp+1#y&amp;i+pys5b@$'


# DEBUG mode: leave the hash in place for now, but remove it and the space (so DEBUG
# is at the start of the line) as soon as something doesn't work. Put it back
# when you get it working again.
# DEBUG = True


ALLOWED_HOSTS = [
    # Add the names and IP address of your host, for example:
    'openrem-server',
    'openrem-server.ad.abc.nhs.uk',
    '10.123.213.22',
]


LOG_ROOT = 'E:/log/'
LOG_FILENAME = os.path.join(LOG_ROOT, 'openrem.log')
QR_FILENAME = os.path.join(LOG_ROOT, 'openrem_qr.log')
EXTRACTOR_FILENAME = os.path.join(LOG_ROOT, 'openrem_extractor.log')


# Regionalisation settings
#   Date format for exporting data to Excel xlsx files.
#   Default in OpenREM is dd/mm/yyyy. Override it by uncommenting and customising below;␣
↪a full list of codes is
#   available at https://msdn.microsoft.com/en-us/library/ee634398.aspx.
# XLSX_DATE = 'mm/dd/yyyy'
#   Local time zone for this installation. Choices can be found here:
#   http://en.wikipedia.org/wiki/List_of_tz_zones_by_name
#   although not all choices may be available on all operating systems.
#   In a Windows environment this must be set to your system time zone.
TIME_ZONE = 'Europe/London'
#   Language code for this installation. All choices can be found here:
#   http://www.i18nguy.com/unicode/language-identifiers.html
LANGUAGE_CODE = 'en-us'


DCMTK_PATH = 'E:/dcmtk/bin'
DCMCONV = os.path.join(DCMTK_PATH, 'dcmconv.exe')
DCMMKDIR = os.path.join(DCMTK_PATH, 'dcmmkdir.exe')
JAVA_EXE = 'E:/pixelmed/windows/jre/bin/java.exe'
JAVA_OPTIONS = '-Xms256m -Xmx512m -Xss1m -cp'
PIXELMED_JAR = 'E:/pixelmed/pixelmed.jar'
PIXELMED_JAR_OPTIONS = '-Djava.awt.headless=true com.pixelmed.doseocr.OCR -'


# E-mail server settings - see https://docs.djangoproject.com/en/2.2/topics/email/
EMAIL_HOST = 'localhost'
EMAIL_PORT = 25
```

```
EMAIL_HOST_USER = ''
EMAIL_HOST_PASSWORD = ''
EMAIL_USE_TLS = 0          # Use 0 for False, 1 for True
EMAIL_USE_SSL = 0          # Use 0 for False, 1 for True
EMAIL_DOSE_ALERT_SENDER = 'your.alert@email.address'
EMAIL_OPENREM_URL = 'http://your.openrem.server'


IGNORE_DEVICE_OBSERVER_UID_FOR_THESE_MODELS = ['GE OEC Fluorostar']
```

Populate OpenREM database and collate static files

In a `CMD` window, move to the openrem Python folder and activate the virtualenv:

```
C:\Users\openrem>e:
E:\>cd venv\Lib\site-packages\openrem
E:\venv\Lib\site-packages\openrem>e:\venv\Scripts\activate
(venv) E:\venv\Lib\site-packages\openrem>
```

Upgrading to a new server

If you are upgrading to a new Windows server, do these additional steps before continuing with those below:

- Rename `E:\venv\Lib\site-packages\openrem\remapp\migrations\0001_initial.py.1-0-upgrade` to `0001_initial.py`

Import the database - update the path to the database backup file you copied from the old server. These steps can take a long time depending on the size of the database and the resources of the server:

```
C:\Users\openrem>"c:\Program Files\PostgreSQL\14\bin\pg_restore.exe" --no-privileges --
→no-owner -U openremuser -d openremdb -W windump.bak
```

Migrate the database:

```
(venv) E:\venv\Lib\site-packages\openrem>python manage.py migrate --fake-initial
```

```
(venv) E:\venv\Lib\site-packages\openrem>python manage.py migrate remapp --fake
```

```
(venv) E:\venv\Lib\site-packages\openrem>python manage.py makemigrations remapp
```

> **Warnung:** Make sure you didn't get a `RuntimeWarning` when running the last command - scroll back up to the command and check you *don't* see the following:
>
> ```
> (venv) E:\venv\Lib\site-packages\openrem>python manage.py makemigrations remapp
> E:\venv\lib\site-packages\django\core\management\commands\makemigrations.py:105:␣
> →RuntimeWarning:
>
> Got an error checking a consistent migration history performed for database␣
> →connection 'default': unable to
> open database file
> ```

> If you do, check the database name and password settings in the `local_settings.py` file. You will need to delete the file `E:\venv\Lib\site-packages\openrem\remapp\migrations\0001_initial.py` before trying again.

```
(venv) E:\venv\Lib\site-packages\openrem>python manage.py migrate
(venv) E:\venv\Lib\site-packages\openrem>python manage.py loaddata openskin_safelist.json
(venv) E:\venv\Lib\site-packages\openrem>python manage.py collectstatic --no-input --
→clear
```

Create the translation files, assuming `gettext` was installed:

```
(venv) E:\venv\Lib\site-packages\openrem>python manage.py compilemessages
```

If this is a new install, not an upgrade, create the superuser account:

```
(venv) E:\venv\Lib\site-packages\openrem>python manage.py createsuperuser
```

Webserver

Configure IIS

- Open `Internet Information Services (IIS) Manager` from the Start menu or the Administrative Tools.
- Click on the name of your server in the `Connections` pane on the left
- Double click on `FastCGI Settings`
- In the `Actions` pane on the right, click `Add Application`
- In the `Full Path:` box type or browse to `E:\venv\Scripts\python.exe`
- In the `Arguments` box type the path to wfastcgi.py: `E:\venv\Lib\site-packages\wfastcgi.py`
- Under FastCGI properties, click on `(Collection)` next to `Environment Variables` and click on the grey `...` box
- In the EnvironmentVariables Collection Editor click `Add`
- Change the value of `Name` to `DJANGO_SETTINGS_MODULE` (must be upper-case)
- Set the `Value` to `openremproject.settings`
- Click `Add` again and add the variable name `PYTHONPATH` with the value `E:\venv\Lib\site-packages\openrem`
- Click `Add` again and add the variable name `WSGI_HANDLER` with the value `django.core.wsgi.get_wsgi_application()`
- Click `OK`
- Under FastCGI Properties -> Process Model click on the `Activity Timeout` value and change it to `1200`

Activity Timeout on slow running systems

If you encounter issues with long-running requests failing on slow running systems, you might try increasing the value of the `Activity Timeout` further.
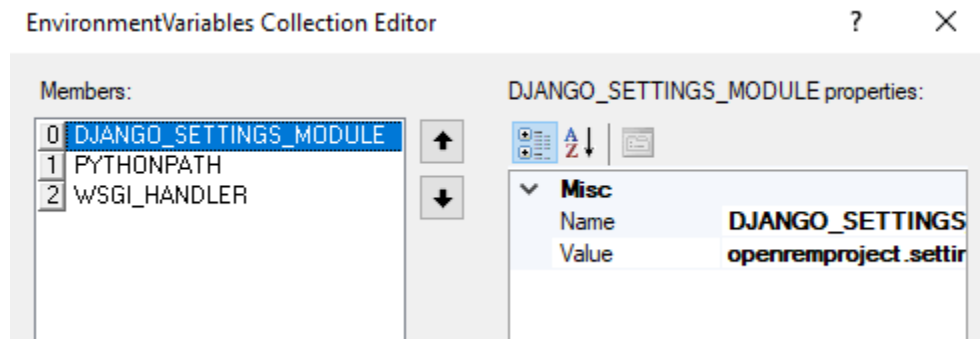
Abb. 7: Figure 7: Environment Variables Collection Editor
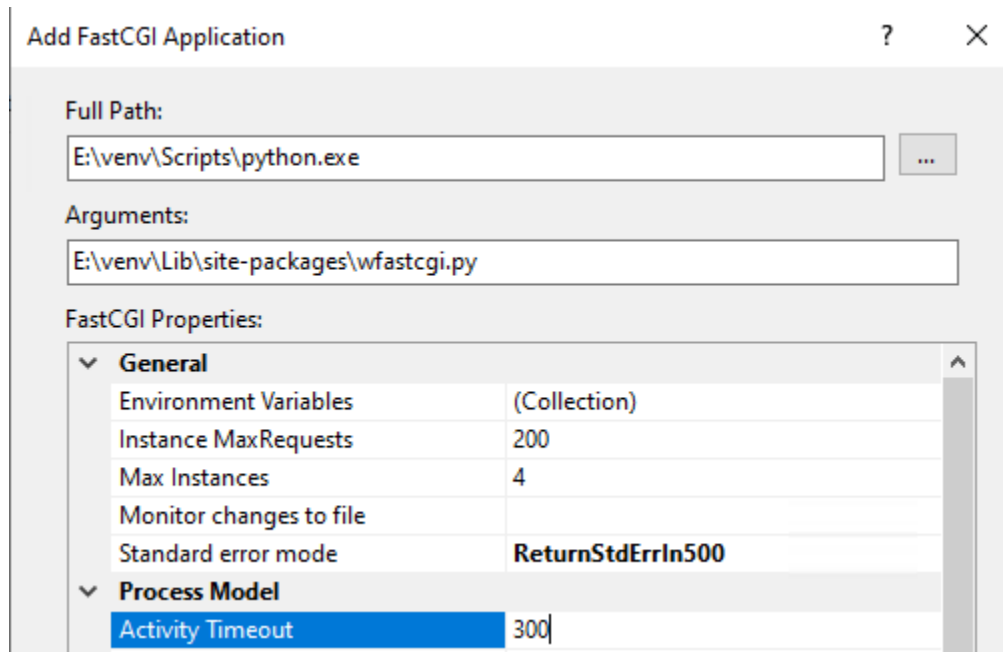
Abb. 8: Figure 8: Add FastCGI Application settings

- Click OK to close the dialogue box

Create a new website

- In the Connections pane expand the tree under server name
- Expand the Sites folder, right click on Default Website and click Remove
- Click Yes
- Right click on Sites and click Add Website...
- Enter Site name as OpenREM
- Under Content Directory Physical path enter or browse to E:\venv\Lib\site-packages\openrem
- Click OK

Configure the new website

- Click on the OpenREM site under Connections in the left pane
- Double click on Handler Mappings
- In the right pane, under Actions click Add Module Mapping...
- In the Request Path box enter an asterix (*)
- In the Module box select FastCgiModule (*not the CgiModule*)
- In the Executable box enter E:\venv\Scripts\python.exe|E:\venv\Lib\site-packages\wfastcgi.py
- In Name type OpenREM CGI handler (value of name is not important)
- Click Request Restrictions and untick the Invoke handler only if request is mapped to: checkbox
- Click OK twice to close the Request Restrictions dialog and the Add Module Mapping dialogue
- When prompted Do you want to create a FastCGI application for this executable? click No

Quick test!

You can now browse on the server to http://localhost/ and you should see an „ugly" version of the website. It will look better after we have configured the static files, next!

Configure IIS to server the static files

- Right click on the OpenREM site under Connections in the left pane
- Click Add Virtual Directory
- Enter static as the Alias
- Enter or browse to E:\static as the Physical path
- Click OK
- Double click on Handler Mappings in the middle pane

- Click on `View Ordered List...` in the right pane

- Select `StaticFile`

- Click `Move Up` in the `Action` pane on the right until `StaticFile` is at the top

- There will be a warning about the list order being changed - click `Yes` to continue

Test the webserver

Browse to http://localhost/ on the server, or browse to the servername in a browser on another machine, and you should be able to see the new OpenREM web service.

Task queue

Running OpenREM on Windows 10 or Windows 11?

For non-server environments, where task executors don't need to be persistent across system restarts, there is a shortcut for starting workers. You can start a single worker in a new console as follows:

```
C:\Users\openrem>E:
E:\>cd venv\Lib\site-packages\openrem
E:\venv\Lib\site-packages\openrem>e:\venv\Scripts\activate
(venv) E:\venv\Lib\site-packages\openrem>python manage.py run_huey
```

If you want more than one worker to run tasks in parallel, you will need to repeat the previous steps for each additional worker in a new console.

You can stop a worker by pressing `Ctrl + C` in the appropriate console

If you cannot start a worker or you are getting error messages, please make sure that your current user has read and write permissions in the `E:\task_queue` directory.

OpenREM uses a task queue to run its background tasks. Therefore, we need additional Windows services that allow us to run these tasks separately from the web application.

To accomplish that we need to do the following:

Create local service account

First we need to create an account that will allow the IIS worker to control the task workers. Most importantly, to kill a task if necessary.

There is a difference if you are connected to an Active Directory or not. Whatever suits your setup, follow the guide `A` if you are not in an Active Directory or `B` if you are.

Guide A

For a Windows instance which is not associated to an Active Directory, it suffices to create a local user account:

- Open the `Search Tab`
- Search for `Add, edit, or remove other users`
- In the menu, click `Add someone else to this PC`
- In the left pane right click on `Users`
- Click `New User...`
- Fill in all fields with the data of a new user account (see image)
- Untick `User must change password at next login`
- Click `Create`
- In the left pane click on `Groups`
- Right click on `IIS_IUSRS`
- Click `Add to Group...`
- Click on the `Add` button
- In the textfield, enter the username of the previously created account
- Click `Ok` twice

Guide B

For a Windows instance that is connected to an Active Directory, or even a controller of one, follow this guide:

- Open the `Server Manager`
- In the navigation bar, click on `Tools`
- Click `Active Directory Users and Computers`
- In the left pane, expand your domain
- Right click on `Users`
- Hove over `New`
- Click on `User`
- Fill in all required fields with the data of a new user account
- Click `Next`
- Enter the new user password twice and untick `User must change password at next login`
- Click `Next` and then `Finish` to create the service account

Creating worker services

Copy the file from

- `E:\venv\Lib\site-packages\openrem\sample-config\queue-init.bat` to

- `E:\winsw\`

Make sure that the previously downloaded and renamed `WinSW.exe` file is in the same folder (`E:\winsw\`).

- Double click the `queue-init.bat` file

- Enter your Domain name or leave empty if not applicable

- Enter the username of the previously created account

- Enter the associated password

- Enter the number of workers you would like to spawn, this number should no exceed the number of CPU cores available to your system

- Wait for the services to get registered and started up (Notice: many windows may appear and disappear quickly)

Adjusting IIS Application Pool Identity

- Open `Internet Information Services (IIS) Manager` from the Start menu or the Administrative Tools.

- In the `Connections` pane expand the tree under server name

- Click on `Application Pools`

- Right click on `OpenREM` in the middle pane

- Click `Advanced Settings...`

- Under `Process Model` click on `Identity` and then on the grey `...` box

- Select the `Custom account:` radio button

- Click on `Set...`

- Enter the credentials of the preivously created account. If you are in an Active Directory prefix ther usernmae with `<YOUR-DOMAIN>\`

- Click `OK` three times

DICOM Store SCP

Copy the Lua file to the Orthanc folder. This will control how we process the incoming DICOM objects.

Copy the file from

- `E:\venv\Lib\site-packages\openrem\sample-config\openrem_orthanc_config_windows.lua` to

- `E:\orthanc\`

Edit the Orthanc Lua configuration options - right click on the file you just copied `Edit with Notepad++`

Set `use_physics_filtering` to true if you want Orthanc to keep physics test studies, and have it put them in the `E:\orthanc\dicom\` folder. Set it to `false` to disable this feature. Add names or IDs to `physics_to_keep` as a comma separated list.

```
-- Set this to true if you want Orthanc to keep physics test studies, and have it
-- put them in the physics_to_keep_folder. Set it to false to disable this feature
local use_physics_filtering = true


-- A list to check against patient name and ID to see if the images should be kept.
-- Orthanc will put anything that matches this in the physics_to_keep_folder.
local physics_to_keep = {'physics'}
```

Lists of things to ignore. Orthanc will ignore anything matching the content of these comma separated lists; they will not be imported into OpenREM.

```
-- Lists of things to ignore. Orthanc will ignore anything matching the content of
-- these lists: they will not be imported into OpenREM.
local manufacturers_to_ignore = {'Faxitron X-Ray LLC', 'Gendex-KaVo'}
local model_names_to_ignore = {'CR 85', 'CR 75', 'CR 35', 'CR 25', 'ADC_5146', 'CR975'}
local station_names_to_ignore = {'CR85 Main', 'CR75 Main'}
local software_versions_to_ignore = {'VixWin Platinum v3.3'}
local device_serial_numbers_to_ignore = {'SCB1312016'}
```

Funktion für das extrahieren der Dosisinformation aus älteren Toshiba und GE Geräte ein- oder ausschalten und für welchen Gerätetyp dies erfolgen soll. Jedes System sollte in der Form {'Manufacturer', 'Model name'} aufgeführt werden, wie das nachstehende Beispiel zeigt:

```
-- Set this to true if you want to use the OpenREM Toshiba CT extractor. Set it to
-- false to disable this feature.
local use_toshiba_ct_extractor = true


-- A list of CT make and model pairs that are known to have worked with the Toshiba CT␣
↪extractor.
-- You can add to this list, but you will need to verify that the dose data created␣
↪matches what you expect.
local toshiba_extractor_systems = {
        {'Toshiba', 'Aquilion'},
        {'GE Medical Systems', 'Discovery STE'},
}
```

Save any changes.

Edit the Orthanc configuration. Navigate to `C:\Program Files\Orthanc Server\Configuration` and right click on `orthanc.json` and click `Edit with Notepad++`:

Dieses Lua Skript zur Othanc Konfiguration hinzufügen:

```
// List of paths to the custom Lua scripts that are to be loaded
// into this instance of Orthanc
"LuaScripts" : [
"E:\\orthanc\\openrem_orthanc_config_windows.lua"
],
```

Set the AE Title and port:

```
// The DICOM Application Entity Title
"DicomAet" : "OPENREM",
```

```
// The DICOM port
"DicomPort" : 104,
```

Bemerkung: Optionally, you may also like to enable the HTTP server interface for Orthanc (although if the Lua script is removing all the objects as soon as they are processed, you won't see much!):

```
// Whether remote hosts can connect to the HTTP server
"RemoteAccessAllowed" : true,

// Whether or not the password protection is enabled
"AuthenticationEnabled" : false,
```

You will also need to open the firewall for port 8042.

To see the Orthanc web interface, go to http://openremserver:8042/ – of course change the server name to that of your server!

Save any changes.

Allow DICOM traffic through the firewall

- Type `windows firewall` in the Start menu to open `Windows Defender Firewall`
- Click `Advanced settings` in the left hand pane to open `Windows Defender Firewall with Advanced Security`
- Click `Inbound Rules` in the left hand pane
- Click `New Rule...` in the right hand pane
- Click `Port` and `Next >`
- Leave as TCP and specify port `104` and click `Next >`
- `Allow the connection`, `Next >`
- Leave the boxes ticked for `When does this rule apply` if that is appropriate, `Next >`
- Name `Orthanc DICOM port`
- `Finish`

Finish off

Restart Orthanc:

- Launch `Services` from the start menu
- Find `Orthanc` on the list and click `Restart`
- Orthanc logs can be reviewed at `C:\Program Files\Orthanc Server\Logs` - the current log file will have the latest date and time in the filename - right click `Edit with Notepad++`

You can check if the port is running and allowed through the firewall using the `Network` tab of Resource Monitor.

Upgrading a native Windows install

Release 1.0 of OpenREM uses a newer version of Python and no longer uses RabbitMQ, Erlang and Celery. Instructions are only provided for Orthanc DICOM server, and no longer for Conquest. The built-in DICOM Store node has been removed.

Consider upgrading to a new Windows server instead of upgrading in place. Instructions for *Upgrading to a new Windows server* are provided including exporting and importing the existing PostgreSQL database.

- something about a clean install, and/or not having old services that are no longer required
- something about being a standardised approach which will make upgrade docs and examples easier to follow

Upgrades from 0.9.1 or earlier should review *Upgrade to OpenREM 0.10.0 from 0.7.3 or later* first. Upgrading to 1.0 is only possible from 0.10.0.

Then best effort upgrade docs… a lot of this can be copied from the *Native Windows install* instructions, or depending on what it ends up looking like, we might point there with a few admonitions to point out differences?

- Export database for backup
- Stop all the services
- Install Python 3.10
- Update PostgreSQL, Orthanc, DCMTK, Pixelmed
- Add/update as necessary gettext, 7Zip, Notepad++
- Install IIS if Apache/NGINX previously in use
- Create virtualenv, activate
- Install new OpenREM, wfastcgi
- Configure OpenREM - use new local_settings.py.windows, adjust database name etc
- Will database be available in new version of PostgreSQL? Or does it need to be imported?
- Rename 0001_initial.py file
- Do the fake-initial etc stuff
- Do the rest of the manage.py stuff
- Configure/reconfigure IIS
- Configure/reconfigure Orthanc

Upgrading to a new Windows server

If OpenREM has been running on an older Windows server version, or you wish to move to Windows Server to host OpenREM, these instructions will guide you through upgrading an existing database to a new server.

This install is based on Windows Server 2022 - for details see the main *Native Windows install* docs.

- **Upgrades from 0.9.1 or earlier should review** *Upgrade to OpenREM 0.10.0 from 0.7.3 or later* first. Upgrading to 1.0 is only possible from 0.10.0.

Get the local_settings.py file

Get `local_settings.py` file from the old server - it should be in one of these locations:

- Ubuntu ‚One page install‘: `/var/dose/veopenrem/lib/python2.7/site-packages/openrem/openremproject/local_settings.py`

- Ubuntu linux: `/usr/local/lib/python2.7/dist-packages/openrem/openremproject/local_settings.py`

- Other linux: `/usr/lib/python2.7/site-packages/openrem/openremproject/local_settings.py`

- Linux virtualenv: `vitualenvfolder/lib/python2.7/site-packages/openrem/openremproject/local_settings.py`

- Windows: `C:\Python27\Lib\site-packages\openrem\openremproject\local_settings.py`

- Windows virtualenv: `virtualenvfolder\Lib\site-packages\openrem\openremproject\local_settings.py`

Export the database

Export the old database on the old server - get details from the `local_settings.py` file:

- Check the database username and change in the command below as necessary (`openremuser`)

- Check the database name and change in the command below as necessary (`openremdb`)

- You will need the password for `openremuser`

- You will need to edit the command for the path to `pg_dump.exe` - the 14 is likely to be a lower number

```
C:\Users\openrem>"c:\Program Files\PostgreSQL\14\bin\pg_dump.exe" -U openremuser -d␣
→openremdb -F c -f windump.bak
```

Transfer the files

Copy these two files to your new server.

Continue on the new server

Now follow the *Native Windows install* instructions looking out for the additional steps for upgrading to a new server.

1.2 Datenbanken

1.2.1 Database administration

Docker installations

Database backup

- Open a shell (command prompt) in the Docker folder

```
$ docker-compose exec db pg_dump -U openremuser -d openrem_prod -F c -f /db_backup/
→openremdump.bak
```

- To automate a regular backup (**recommended**) adapt the following command in a bash script:

```bash
#!/bin/bash
TODAY=$(date "+%Y-%m-%d")
docker-compose -f /path/to/docker-compose.yml exec db pg_dump -U openremuser -d openrem_
→prod -F c -f "/db_backup/openremdump-"$TODAY".bak"
```

- or powershell script:

```
$dateString = "{0:yyyy-MM-dd}" -f (get-date)
docker-compose -f C:\Path\To\docker-compose.yml exec db pg_dump -U openremuser -d␣
→openrem_prod -F c -f /db_backup/openremdump-$dateString.bak
```

You will need to ensure the backups are either regularly deleted/moved, or overwritten so that the backups don't fill the disk.

Database restore

To restore a database backup to a new Docker container, install using the *Installation* instructions and bring the containers up, but don't run the database commands. These instructions can also be used to create a duplicate server on a different system for testing or other purposes.

- Requires exactly the same version of OpenREM to be installed as the database was exported from

- Copy the database backup to the db_backup/ folder of the new install (the name is assumed to be openremdump.bak, change as necessary)

- Open a shell (command prompt) in the new install folder (where docker-compose.yml is)

```
$ docker-compose exec db pg_restore --no-privileges --no-owner -U openremuser -d openrem_
→prod /db_backup/openremdump.bak
```

You may get an error about the public schema, this is normal.

- Get the database ready and set up Django:

```
$ docker-compose exec openrem python manage.py migrate --fake-initial
```

```
$ docker-compose exec openrem python manage.py makemigrations remapp
```

```
$ docker-compose exec openrem python manage.py migrate --fake
```

```
$ docker-compose exec openrem python manage.py collectstatic --noinput --clear
```

```
$ docker-compose exec openrem python django-admin compilemessages
```

The OpenREM server should now be ready to use again.

Advanced

These methods should not be required in normal use; only do this if you know what you are doing!

psql

Start the PostgreSQL console:

```
$ docker-compose exec db psql -U openremuser openrem_prod
```

```
-- List users
\du

-- List databases
\l

-- Exit the console
\q
```

pgAdmin or other PostgreSQL connections

To access the database directly by pgAdmin or other software, the ports must be exposed.

- Edit `docker-compose.yml` to add the ports:

```
db:
  ports:
    - 5432:5432
```

- If you have a database already running on the host machine, this port will prevent the container starting. In this case, change the first number in the pair to an alternative port.

- The service will be accessible on the host machine after the containers are taken down and up again:

```
$ docker-compose down
$ docker-compose up -d
```

Linux installations

Database backup

- Check the database username and change in the command below as necessary (`openremuser`)
- Check the database name and change in the command below as necessary (`openremdb`)
- You will need the password for `openremuser`
- Ad hoc:

```
$ sudo -u postgres pg_dump -U openremuser -d openremdb -F c -f openremdump.bak
```

- To automate a regular backup (**recommended**) adapt the following command in a bash script:

```
#! /bin/bash
rm -rf /path/to/db/backups/*
PGPASSWORD="mysecretpassword" /usr/bin/pg_dump -U openremuser -d openremdb -F c -f /path/
↪to/db/backups/openremdump.bak
```

Database restore

- Requires the same version of OpenREM to be installed as the database was exported from, unless you are *Up-grading a native Linux install* or *Upgrading to a new Linux server*.
- Username can be changed on restore by specifying the new user in the restore command. The user must exist in PostgreSQL though - `sudo -u postgres createuser -P openremuser` if required
- `openrem/remapp/migrations/` should be empty except `__init__.py`

```
$ sudo -u postgres createdb -T template0 new_openremdb_name
$ sudo -u postgres pg_restore --no-privileges --no-owner -U openremuser -d new_openremdb_
↪name path-to/openremdump.bak
```

- Update the `local_settings.py` file with the new database details, as per *OpenREM konfigurieren*
- Set up the new database with Django/OpenREM after activating the virtualenv and moving to the `site-packages/openrem` folder:

```
$ python manage.py migrate --fake-initial
$ python manage.py migrate remapp --fake
$ python manage.py makemigrations remapp
$ python manage.py migrate
```

Windows installations

Database backup

- Check the database username and change in the command below as necessary (`openremuser`)
- Check the database name and change in the command below as necessary (`openremdb`)
- You will need the password for `openremuser`
- You will need to edit the command for the path to `pg_dump.exe` - the 14 is likely to be a lower number
- Ad hoc:

```
C:\Users\openrem>"c:\Program Files\PostgreSQL\14\bin\pg_dump.exe" -U openremuser -d␣
↪openremdb -F c -f windump.bak
```

- To automate a regular backup (**recommended**) adapt the following command in a bat script:

> **Warnung:** Content to be added!

Database restore

- Requires the same version of OpenREM to be installed as the database was exported from, unless you are *Upgrading a native Windows install* or *Upgrading to a new Windows server*.

- Username can be changed on restore by specifying the new user in the restore command. The user must exist in PostgreSQL though - create the user in pgAdmin if required

- openrem\remapp\migrations\ should be empty except __init__.py

```
C:\Users\openrem>"c:\Program Files\PostgreSQL\14\bin\pg_restore.exe" --no-privileges --
→no-owner -U openremuser -d openremdb -W windump.bak
```

- Update the `local_settings.py` file with the new database details, as per *OpenREM konfigurieren*

- Set up the new database with Django/OpenREM after activating the virtualenv and moving to the `site-packages\openrem` folder:

```
(venv) E:\venv\Lib\site-packages\openrem>python manage.py migrate --fake-initial
(venv) E:\venv\Lib\site-packages\openrem>python manage.py migrate remapp --fake
(venv) E:\venv\Lib\site-packages\openrem>python manage.py makemigrations remapp
(venv) E:\venv\Lib\site-packages\openrem>python manage.py migrate
```

1.3 Advanced server configuration

1.3.1 Webserver configuration

Webserver timeout

Some long running actions can cause webserver errors if they take longer than the timeout setting in the webserver, particularly generating fluoroscopy *Skin dose maps*. The default setting is 300 seconds, or five minutes. To modify this, change the following two settings:

Edit `docker-compose.yml` in the Docker OpenREM installation folder and change the timeout setting on the following line:

```
services:
  openrem:
    container_name: openrem
    command: gunicorn openremproject.wsgi:application --bind 0.0.0.0:8000 --timeout 300
```

Edit `nginx-conf/conf.d/openrem.conf` and set the same timeout:

```
server {
    listen 80;
    location / {
        proxy_pass http://openremproject;
        # ...
        proxy_read_timeout 300s;
    }
}
```

Reload the containers:

```
$ docker-compose down
$ docker-compose up -d
```

Non-Docker install

Change the same settings as for the Docker install above:

```
$ sudo nano /etc/nginx/sites-available/openrem-server
```

and

```
$ sudo nano /etc/systemd/system/openrem-gunicorn.service
```

```
ExecStart=/var/dose/veopenrem3/bin/gunicorn \
    --bind unix:/tmp/openrem-server.socket \
    openremproject.wsgi:application --timeout 300
```

Adding an SSL certificate

It is advisable to add an SSL certificate to the web server even though it might only be accessible within an institution. There are several reasons for this, but one main one is that over time web browsers will give more and more warnings about entering passwords into non-HTTPS websites.

It is likely that within your institution there will be a corporate trusted root certificate and a mechanism of getting certificates you generate for your servers signed by that root certificate. How to generate a certificate signing request (CSR) and private key are beyond the scope of these documents, but this blog post was helpful when we were learning how to do this at our institution: https://www.endpoint.com/blog/2014/10/30/openssl-csr-with-alternative-names-one

Once you have a signed certificate, place it and the key in `nginx-conf/certs`, where it will be available in the Nginx container at `/etc/ssl/private`.

There are two conf files in `nginx-conf/conf.d` - the default one is `openrem.conf`. There is an alternative one named `openrem-secure.conf.example`. Edit the second file as required, then rename them both so the secure version is the only one to have a `.conf` ending.

Ensure the the following lines are updated for the name of your server and the names of your signed certificate and key:

```
server {
    listen 443 ssl;
    server_name add_server_name_here;
    ssl_certificate /etc/ssl/private/openrem.cer;
    ssl_certificate_key /etc/ssl/private/openrem.key;

    # ...
}
```

1.3.2 Running the OpenREM website in a virtual directory

If you want to run the OpenREM in a virtual directory (like http://server/dms/) you need to configure this in your web server configuration as well as in the OpenREM configuration.

The following steps are necessary:

- Configure virtual directory settings in the Docker `.env.prod` file
- Update Nginx webserver configuration
- Update the `reverse.js` file
- Restart the containers

Docker setup

Stop the containers if they are running before changing the configuration, using a shell (command prompt) in the Docker OpenREM installation folder

```
$ docker-compose down
```

Configure virtual directory settings in .env.prod

Django needs to know the virtual directory name and which URLs the static and media files are served from.

Edit `.env.prod`, uncomment the following lines (remove the #) and set them as appropriate. The `VIRTUAL_DIRECTORY` setting must have a trailing `/`. For example, to serve the website from a subfolder/virtual directory named `dms`:

```
## For installations in a virtual directory
VIRTUAL_DIRECTORY=dms/
MEDIA_URL=/dms/media/
STATIC_URL=/dms/static/
```

Modify webserver configuration

Edit `nginx-conf/conf.d/openrem.conf` to update the locations — again using the example virtual directory `dms`:

```
server {
    listen 80;
    location /dms/ {
        proxy_pass http://openremproject;
        # ...
    }
    location /dms/static/ {
        alias /home/app/openrem/staticfiles/;
    }
    location /dms/media/ {
        alias /home/app/openrem/mediafiles/;
    }
}
```

Start the containers

```
$ docker-compose up -d
```

Update reverse.js

The static reverse.js file should be updated in order to change the URLs in the static javascript files.

Open a shell (command prompt) and navigate to the Docker OpenREM installation folder

```
$ docker-compose exec openrem python manage.py collectstatic_js_reverse
```

Test!

You should now be able to reach the OpenREM interface using the virtual directory address.

Non-Docker install

```
$ sudo systemctl stop openrem-gunicorn.service
$ sudo systemctl stop nginx.service
```

Update local_settings.py

Update `local_settings.py` with the same variables as in the `.env.prod` file. If the values aren't in your copy of the file just add them in:

```
$ cd /var/dose/veopenrem3/lib/python3.10/site-packages/openrem/
$ nano openremproject/local_settings.py
```

```
VIRTUAL_DIRECTORY = "dms/"
STATIC_URL = "/dms/static/"
MEDIA_URL = "/dms/media/"
```

Modify webserver configuration

```
$ sudo nano /etc/nginx/sites-available/openrem-server
```

```
server {
    # ...
    location /dms/static {
        alias /var/dose/static;
    }
    location /dms {
        proxy_pass http://unix:/tmp/openrem-server.socket;
        # ...
```

```
    }
}
```

Update reverse.js

```
$ . /var/dose/veopenrem3/bin/activate
$ cd /var/dose/veopenrem3/lib/python3.8/site-packages/openrem/
$ python manage.py collectstatic_js_reverse
```

Restart the services

```
$ sudo systemctl start openrem-gunicorn.service
$ sudo systemctl start nginx.service
```

Start all the services

2.1 Test web server

In a shell/command window, move into the openrem folder:

- Ubuntu linux: `/usr/local/lib/python2.7/dist-packages/openrem/`

- Other linux: `/usr/lib/python2.7/site-packages/openrem/`

- Linux virtualenv: `vitualenvfolder/lib/python2.7/site-packages/openrem/` (remember to activate the virtualenv)

- Windows: `C:\Python27\Lib\site-packages\openrem\`

- Windows virtualenv: `virtualenvfolder\Lib\site-packages\openrem\` (remember to activate the virtualenv)

2.1.1 Web access on OpenREM server only

Run the built in web server:

```
python manage.py runserver --insecure
```

In a web browser on the same computer, go to http://localhost:8000/ - you should now see the message about creating users.

2.1.2 Web access on other computers

The built-in webserver only provides a service on the computer OpenREM is installed on by default (it's only there really for testing). To view the OpenREM interface on another computer, you need to modify the `runserver` command:

```
python manage.py runserver --insecure 0.0.0.0:8000
```

This will enable the web service to be available from other computers on the network. If your server has several network cards and you want to restrict it to one, then you can use a real address rather than `0.0.0.0`. Likewise you can specify the port (here it is `8000`).

In a web browser on a different computer on the same network, go to http://192.168.1.10:8000/ (**changing the IP address** to the one you are running the server on) and you should see the OpenREM interface and the message about creating users.

Bemerkung: Why are we using the `--insecure` option? With `DEBUG` mode set to `True` the test web server would serve up the static files. In this release, `DEBUG` mode is set to `False`, which prevents the test web server serving those files. The `--insecure` option allows them to be served again.

2.2 Configure the settings

- Follow the link presented on the front page to get to the user and group administration.

> There are no users in any of the groups
>
> You will need to allocate users to a group before using this system - you can do this here. You will need to know the superuser username and password you used when you installed the database.
>
> Make sure there is at least one Admin user. You can return to the user config page later by using the 'Manage users' link on the admin menu.

- After the first users are configured, this link will no longer be presented and instead you can go to `Config -> Users`.
- You will need the superuser username and password you created just after creating the database. The groups are
    - `viewgroup` can browse the data only
    - `importsizegroup` can use the csv import facility to add patient height and weight information
    - `importqrgroup` can use the DICOM query-retrieve facility to pull in studies, as long as they are pre-configured
    - `exportgroup` can view and export data to a spreadsheet
    - `pidgroup` can search using patient names and IDs depending on settings, and export with patient names and IDs if they are also a member of the `exportgroup`
    - `admingroup` can delete studies, configure DICOM Store/QR settings, configure DICOM keep or delete settings, configure patient ID settings, and abort and delete patient size import jobs. *Members of the admingroup no longer inherit the other groups permissions.*
- In addition to adding users to these groups, you may like to grant a second user `superuser` and `staff` status so that there are at least two people who can manage the users
- Return to the OpenREM interface (click on `View site` at the top right)

- Follow the link to see more information about how you want OpenREM to identify non-patient exposures, such as QA. See *Not-patient indicator settings*.

- Go to `Config -> DICOM object delete settings` and configure appropriately (see *Delete objects configuration*)

- Go to `Config -> Patient ID settings` and configure appropriately (see *Patient identifiable data*)

- If you want to use OpenREM as a DICOM store, or to use OpenREM to query remote systems, go to `Config -> Dicom network configuration`. For more information go to *Importing data to OpenREM*.

- With data in the system, you will want to go to `Config -> View and edit display names` and customise the display names. An established system will have several entries for each device, from each time the software version, station name or other elements changes. See *Display names and user-defined modalities* for more information

2.3 Start using it - add some data!

See *Importing data to OpenREM*

Configuration and administration

3.1 Home page options
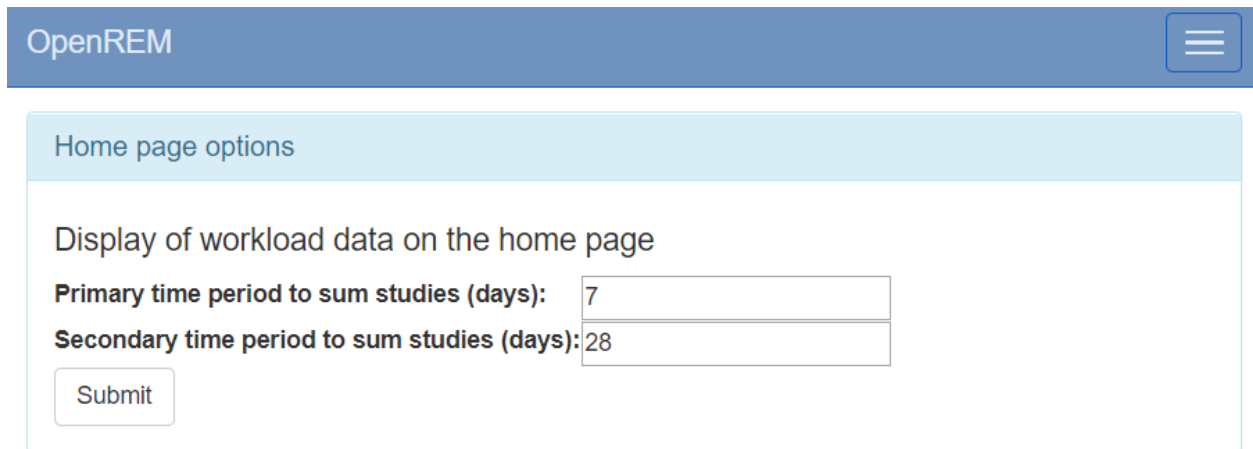
3.1.1 Display of workload information

The home page can be configured to show the number of studies carried out in the past 7 (default) and 28 (default) days for each system. These default values can be changed by logging in, clicking on the `Config` menu at the right-hand end of the navigation bar, and then selecting the `Home page options` entry under `User level config` shown in the upper section of figure 1. This takes the user to a page where the two time periods can be viewed and updated (figure 2).

By default the display of workload information is disabled; this can be changed by an OpenREM administrator via the `Home page options`. When an OpenREM administrator views the home page options a tick box is included that enables or disables the display of workload data on the home page (figure 3).
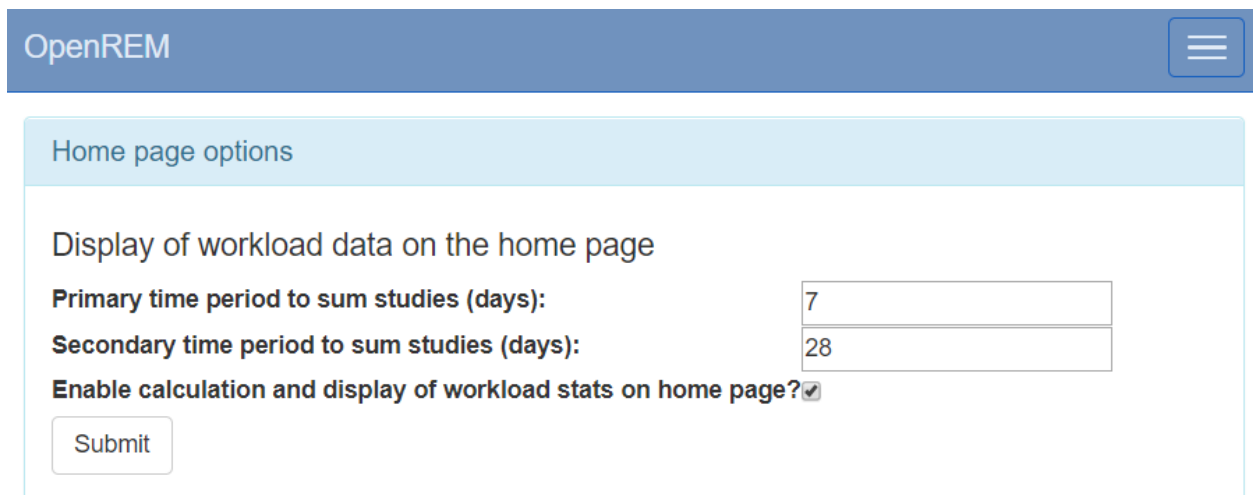
When workload information is displayed, the link to the system data is modified in the workload cells to filter the studies to the same date range.

Tasks ▾   ⚙ Config ▾

User level config

Chart options

Switch charts off

Home page options

System level settings

Users

Display names & modality

DICOM object deletion

DICOM networking

Patient ID

Not-patient indicators

Skin dose map

Fluoro alert levels

Fluoro alert notification

Standard name settings

Abb. 2: Figure 2: The home page options form

Abb. 3: Figure 3: The home page options admin form

3.2 Delete objects configuration

OpenREM is able to automatically delete DICOM objects if they can't be used by OpenREM or if they have been processed. This has the following advantages:

- The server doesn't need to have much storage space

- It can help with information governance if the database is set to not store patient identifiable data (see *Patient identifiable data*)

Warnung: If OpenREM is set to delete objects and you pass a local file to Open-REM using the command line, the source file will be deleted (as long as the file-system permissions allow).

3.2.1 Configure what is deleted

Use the `Config` menu and select `DICOM object deletion`:

This will open the configuration page:

Abb. 4: The `Config` menu

The initial settings are to not delete anything. However, you are likely to want to delete objects that don't match any import filters, and also to delete images such as mammo, DX and Philips CT, as these will take up space much more quickly than the radiation dose structured reports.
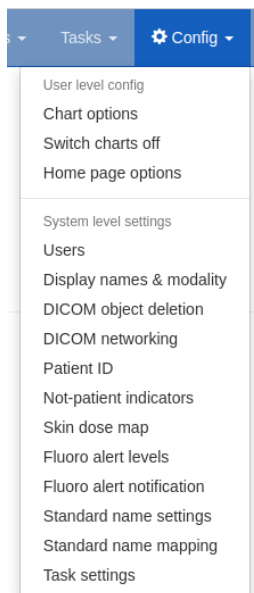
Modify DICOM object deletion policy

Do you want objects that we can't do anything with to be deleted?

☐ Delete objects that don't match any import functions?

The remaining choices are for DICOM objects we have processed and attempted to import to the database:

☐ Delete radiation dose structured reports after processing?

☐ Delete radiography images after processing?

3.2.2 Reviewing the settings

When you have set your preferences, you will be redirected to the DICOM network configuration page, where at the bottom you can review the current settings:

Abb. 6: Deletion policies can be reviewed on the DICOM network configuration page

More information about the DICOM network configuration can be found on the *Direct from modalities* page.

3.3 Display names and user-defined modalities

Inhalt

3.3.1 The display name field

Previous versions of OpenREM used each X-ray system's DICOM `station name` as the identifier for each X-ray system. The front page showed a summary of the number of studies for each unique `station name` stored in the system. This led to a problem if multiple X-ray systems used the same station name: the OpenREM home page would only show one station name entry for these systems, with the number of studies corresponding to the total from all the rooms. The name shown alongside the total was that of the system that had most recently sent data to the system.

This issue has been resolved by introducing a new field called `display name`. This is unique to each piece of X-ray equipment, based on the combination of the following eight fields:

- manufacturer

- institution name

- station name

- department name

- model name

- device serial number

- software version

- gantry id

The default text for `display name` is set to a combination of `institution name` and `station name`. The default display name text can be changed by a user in the `admingroup` — see *Setting display name automatically for known devices*

3.3.2 User defined modality field

OpenREM determines the modality type of a system based on the information in the DICOM radiation dose structured report. However sometimes this mechanism fails because vendors use templates meant for RF also for DX systems. Therefore it is possible from version 0.8.0 to set a modality type for each system manually. A manually set modality type overrides the automatically determined value.

3.3.3 Viewing X-ray system display names and user defined modality

If you log in as a normal user then the `Config` menu becomes available at the right-hand end of the navigation bar at the top of the screen.

The third option, `View display names & modality`, takes you to a page where you can view the list of X-ray systems with data in OpenREM together with their current display name and user defined modality. If the user defined modality is not set, the value contains `None`. The X-ray systems are grouped into modalities and displayed in five tables: CT; mammography; DX and CR; fluoroscopy; and other.

Abb. 8: Example list of display names

3.3.4 Setting display name automatically for known devices

If you are a member of the `admingroup` you can set an option to automatically set the display name of already known devices even if one of the above mentioned `fields` changed. A device can send its Device Observer UID (especially in rdsr-objects). This is a unique ID for the device. If this UID is received by OpenREM it can set the display name and modality type the same as an already known device with the same Device Observer UID. This option can be useful if other parameters that OpenREM looks at frequently change. If you want to see if one of the other parameters changed (like software version), don't tick this option.

3.3.5 Changing X-ray system display names and user defined modality

If you wish to make changes to a display name or to the user defined modality then you must log in as a user that is in the `admingroup`. You will then be able to use the `Display names & modality` item under the `Config` menu:

This will take you to a page where you can view the list of X-ray systems with data in OpenREM. If you wish to change a display name or the user defined modality then click on the corresponding row. The resulting page will allow you to edit these parameters. Click on the `Update` button to confirm your changes:
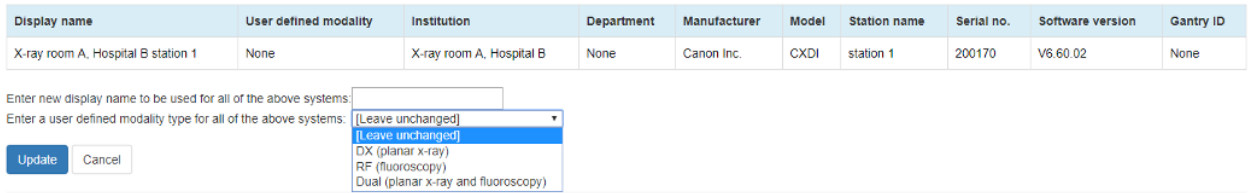
Abb. 10: Example of the page for updating a display name and user defined modality

Abb. 9: The `Config` menu (admin)

You can change multiple rows at once. For display names you may wish to do this if a system has a software upgrade, for example, as this will generate a new default display name for studies carried out after the software upgrade has taken place. The studies from these will be grouped together as a single entry on the OpenREM homepage and individual modality pages.

If you update the user defined modality, the modality type for already imported studies will also be set to the user defined modality type. Only changes from modality DX (planar X-ray) to RF (fluoroscopy) and vice versa are possible.

Dual modality systems

Some systems are dual purpose in that they can be used in both standard planar X-ray mode and in fluoroscopy mode. For these systems you can configure them as ‚Dual' and OpenREM will attempt to reprocess all the studies related to the rows you have selected and assign them to DX or RF. The studies will then be displayed in the right sections in the web interface and will export correctly. New RDSRs relating to that X-ray system will be assigned a modality in the same way.

After an X-ray system has been set to Dual you may wish to reprocess the studies to assign modality again. To do this you can use the ‚reprocess' link in the ‚User defined modality' cell:

Abb. 11: Re-sort studies into planar X-ray and fluoroscopy
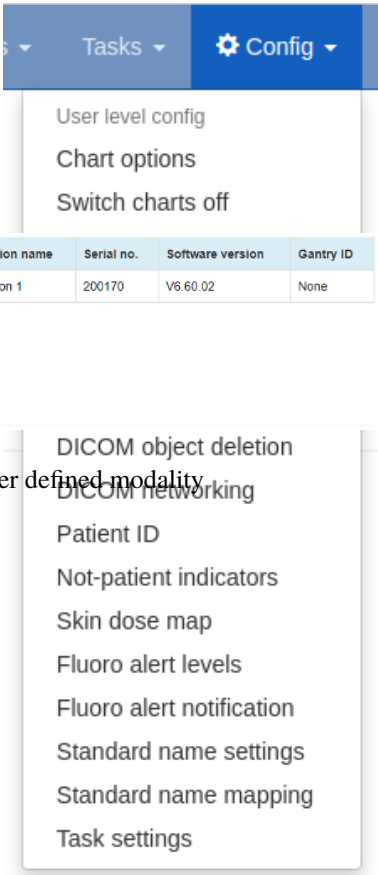
3.3.6 Review of studies that failed to import

Studies that have failed early in the import process might not have an entry in the `unique_equipment_name` table, and therefore will not appear in any of the other tables on this page. The table at the end allows the user to review these studies and delete them. See *Failed import studies* for more details.

3.3.7 Systems where Device Observer UID is not static

OpenREM users have found one x-ray system which incorrectly sets the Device Observer UID to be equal to the Study Instance UID. In this situation a new entry is created in the display name settings for every new exam that arrives in OpenREM, making the display name table fill with many duplicate entries for the same system. To avoid this problem a list of models can be specified using the variable below - OpenREM will ignore the Device Observer UID value when creating new display names for any model in this list. The model name text must exactly match what is contained in the system's Manufacturer's Model Name DICOM tag (0008,1090).

```
IGNORE_DEVICE_OBSERVER_UID_FOR_THESE_MODELS = ['GE OEC Fluorostar']
```

- For Docker installations, this setting is in the *Docker env configuration*.
- For Linux installations, see the *OpenREM konfigurieren* docs.
- For Windows installations, see the *OpenREM konfigurieren* docs.

3.4 Not-patient indicator settings

The standard configuration for OpenREM is to not store any patient identifiable information. Therefore it can be difficult to distinguish between real patients and test or quality assurance exposures.

Changed in 0.8.0

To aid identification of non-patient exposures, the patient name and the patient ID are checked against a set of patterns, and if a match is found then the pattern is recorded in the database before the patient name and ID are deleted or converted to a hash (see *Patient identifiable data* for details).

3.4.1 Setting the patterns to identify non-patient studies

Use the `Config` menu and select `Not-patient indicators`:

The patient name and the ID are matched against the patterns you configure. The patterns make use of wildcards as per the following table, and are case insensitive:

Pattern	Meaning
*	matches everything
?	matches any single character
[seq]	matches any character in seq
[!seq]	matches any character not in seq

To match all studies where the patient name begins with `physics`, the pattern should be set to `physics*`. This would match `Physics^RoutIQ` but not match `Testing^Physics`. The patient name in DICOM is normally formatted `Family name^Given name^Middle name^Prefix^Suffix`. Therefore to match any studies where the first name is `Test`, you would set the pattern to be `*^test*`.
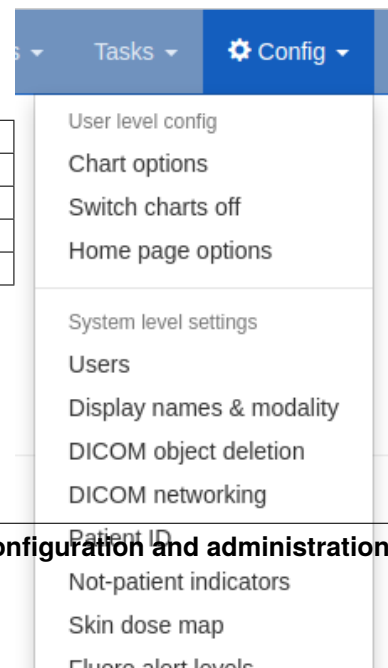
If your test patient name always starts with PHY and then a number, you might use this pattern: `phy[0-9]*`. Here we have used a range for the sequence to match any number, but it will only match one character per sequence, so a `*` is required to match all the characters after the first number. This pattern will match `Phy12345` and `PHY6test` but not `Phyliss`.

The pattern list for patient name and the list for patient ID are separate, so both need to be populated to meet your requirements.

Creating new patterns

Click on `Add ID patterns` or `Add name patterns` in the panel title bar and follow the instructions.

Modifying patterns

Click the `Modify` link in the row of the pattern you wish to modify.

Deleting patterns

Click the `Delete` link in the row of the pattern you wish to delete. You will be asked to confirm the deletion.

3.4.2 Replicating behaviour of release 0.7.4 and earlier

OpenREM releases before 0.8 had the not-patient identification patterns hardcoded. From release 0.8.0 the patterns are (admin) user configurable, but will start with no patterns in place. To add the patterns that would maintain the behaviour of previous releases, use the link at the bottom of the config page, or the link in the add/modify pages.

3.5 Patient identifiable data

Prior to version 0.7, no data that is generally considered to be patient identifiable was stored in the OpenREM database.

The following patient descriptors have always been recorded if they were available:

- Patient age at the time of the study, but not date of birth (though this could be calculated from age)

- Patient sex

- Patient height

- Patient weight

In addition, a key identifier for the exam that is normally not considered patient identifiable was stored:

- Study accession number

It has become apparent that there are reasons where people need to store patient identifiable data to make the most of OpenREM, so this is now configurable from version 0.7 onwards.

3.5.1 Configure what is stored

On the Config menu, select `Patient ID`:

The initial settings are as follows:

The default for patient name, ID and date of birth is to not store them. There isn't an option currently to not store the accession number, though OpenREM continues to work if it is missing.

To store patient identifiable data from now on, select the relevant box and press `Submit`. If you change the setting again later, then data already stored will remain in the database.

3.5.2 Store encrypted data only

If you wish to have the patient name and/or ID available for finding studies relating to a specific patient, but do not need to identify who that patient is, then it is possible to create an ,encrypted' version of the ID or name. In this case, a one-way SHA 256 hash is generated and the hash value is stored instead.

If *exactly* the same name or ID (including spelling, spacing, case etc) occurs more than once, then the same hash will be generated.

The same applies to accession numbers if the option to encrypt the accession number is selected.

3.5.3 Using patient identifiable data

Querying for patient studies

In the modality pages of the OpenREM web interface, if you are in the `pidgroup` you will have a filter for patient name and patient ID available:

Patient name:

Patient ID:

If the values in the database are *not* encrypted, then partial search terms can be used as a case-insensitive ‚contains' query will be applied.

If the values are encrypted, then only the entire string, with exactly the same case, spacing and punctuation will match. This is more likely to be successful with patient ID than with patient name.

Study export with patient identifiers

Users in the `pidgroup` will have extra export buttons available in the modality pages:

Data export

Note: Apply the exam filter first to refine what is exported.

| Export to CSV | With names | With ID | With both |
| Export to XLSX | With names | With ID | With both |

If the IDs or names are encrypted, then these columns will contain the hash rather than the original values. However, it will be possible to see if more than one study belongs to one patient as the values should be the same for both. Due to the nature of the algorithm however, a single change in the name or ID - such as an upper case letter instead of a lower case one - will be recorded as a completely different hash value.

Any exports with either patient name or patient ID included will also have a date of birth column.

3.6 Deleting studies

3.6.1 Individual studies

If you log in as a user that is in the `admingroup`, then an extra column is appended in the filtered view tables to allow studies to be deleted:

ation name	Date	Study description | Accession number	Number of events	Dose Length Product Total mGy.cm	Delete?
TOM CTAWP1234	2013-05-23 10:09	Thorax^TAP120kvlV (Adult) | R⬛⬛⬛⬛⬛⬛⬛1	4	1257.10	Delete
TOM CTAWP1234	2013-05-23 11:05	Thorax^TA_IV120kV (Adult) | R⬛⬛⬛⬛⬛⬛⬛1	4	314.26	Delete
TOM	2013-05-23	Thorax^TAP120kvlV (Adult) |	4	688.99	Delete

Clicking on delete takes you to a confirmation page before the delete takes place.

3.6.2 All studies from one source

If you log in as a user that is in the `admingroup`, on the `Config` menu select `Display names & modality` to get to a list of all the X-ray systems with data in OpenREM. More information about *Display names and user-defined modalities*.

Each row is a unique combination of all the column headers, so if a modality has a software update for example this will usually mean a new row is started.

In the last column is a link to `Review` the studies from that source. This can be useful for troubleshooting a particular source, or you can use it to delete all the studies from one source in one go.

The details for that source are displayed, along with a table showing which sort of data is contained in each study. Above the ‚Study deletion options' panel the following two numbers are indicated:

1. The number of studies associated with this equipment

2. The number of studies associated with this equipment after being filtered by the indicated modality type

If the second number is smaller than the first, this will indicate that some of the studies from the equipment have been labelled with a different modality type. There will therefore be an entry in one of the other tables on the equipment display name page.
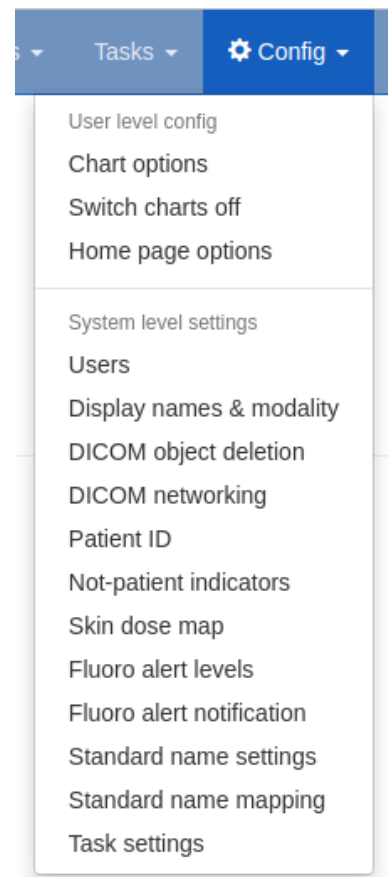
Abb. 13: The `Config` menu (admin)

The entry in the unique equipment names table for these DX studies looks like this

Display name	User defined modality	Institution	Department	Manufacturer	Model	Station name	Serial no.	Software version	Gantry ID
X-ray room C, Hospital A station 3	None	X-ray room C, Hospital A	None	Canon Inc.	CXDI	station 3	200170	V6.60.02	None

There are 77 studies associated with this equipment, and 77 studies in this list which has been filtered by the modality DX (including CR)

Study deletion options

Which should you choose, if you want to remove these studies?

If you have added ths equipment to an equipment name, for example "Imported", then just delete the studies so that new studies that are imported will drop into the same display name. Otherwise delete the studies and the equipment name table entry too.

[Delete studies] [Delete studies and table entry]

Page 1 of 4. next

General		Patient module		CT data			DX/RF/MG		Accumulated data			Irradiation event data				
Date	Time	General	Study	Template	Accumulated	Events	Template	Accumulated	Fluoro & DX	Mammography	Cassette based	Projection	General	Detector	Source	Mechanical
Dec. 6, 2014	2:57 p.m.	Yes	Yes. Age 46.2				Yes	Yes				DAP total 19.67 cGy.cm^2	1 event. e1: Chest AP 19.67 cGy.cm^2	e1 present,	e1 present,	e1 present,
June 15, 2014	6:45 p.m.	Yes	Yes. Age 84.8				Yes	Yes				DAP total 5.83 cGy.cm^2	1 event. e1: Chest AP 5.83 cGy.cm^2	e1 present,	e1 present,	e1 present,
Nov	6:01	Yes	Yes				Yes	Yes				DAP total	1 event. e1:	e1	e1	e1 present,

Abb. 14: Source equipment review page with study delete options

Delete studies and table entry

Use this button if you want to delete all the studies and remove the entry that has been made in the Unique Equipment Names table. Otherwise, the entry would remain but with zero studies associated with it. The deletion takes a second confirmation step.

If there are studies associated with this equipment that are listed with a modality type different to the one shown, those studies will not be deleted and the table entry will not be removed.

Delete studies

If you have associated this table entry with a `Display name` and you want any future studies to fall under the same name, you can leave the entry in the Unique Equipment Names table. You might want to do this for example if you have a Display name of ‚CR' or ‚Imported'. Again, there is a confirmation step.

Again, only the studies associated with this equipment that have the same modality type as shown will be deleted.

3.6.3 Failed import studies

At the bottom of the `Display names & modality` page is a table listing the number of studies that are in the database, but do not have an entry in the `unique_equipment_name` table. This usually indicates a study that has failed early in the import process.

Users in the `admingroup` are able to click on the links to review the studies on a per-modality basis. This will list the information that is available, which might indicate which system they came from, what times, dates and accession numbers.

The user is then able to delete all the failed import studies in the list.

Before release 0.8.2, these studies would appear in the homepage listing as *Error has occurred - import probably unsuccessful*. This has now changed to a link to the review page for that modality with the text *Failed import - review here* for users in the `admingroup` and *Failed import - ask an administrator to review* for other users.

3.7 Adding patient size information from csv using the web interface

Inhalt

- *Adding patient size information from csv using the web interface*
    - *Uploading patient size data*
    - *Importing the size data to the database*
    - *Reviewing previous imports*
    - *Deleting import logs*
- *Adding patient size information from csv using the command line*

3.7.1 Uploading patient size data

If you log in as a user that is in the `admingroup`, then a menu is available at the right hand end of the navigation bar:
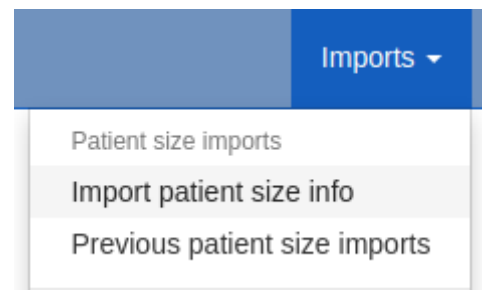
The first option takes you to a page where you can upload a csv file containing details of the patient height and weight, plus either the accession number or the Study Instance UID.

Uploading patient size data to OpenREM

In most instances, dose metrics from the modalities make much more sense when reviewed in conjunction with patient size. This interface allows you to upload a csv file containing patient size information that can then be imported to the existing data in the database.

What needs to be in the csv file?

The csv file needs to contain a column for each of the following, with a column title in the first row. The columns can be in any order; additional columns will be ignored:

- Patient height (in cm)
- Patient weight (in kg)
- Study identifier*
- Study identifier type*

* The study identifier can be either the accession number or the Study Instance UID. The column titles can be anything, and there can be as many other columns as you like.

Select a file:

Choose file | No file chosen

Upload csv to be processed

Notes:

If you have a csv file with weight but not height or vice-versa, just add a column header to a blank column to suit. Similarly, if you have the information in units other than kg and cm, create a new column with those units and use the title of that column instead.

Data already in the database does not get overwritten. So if a study already has a height or weight, or if the same study identifier is used more than once in the csv file on different roles, only the first entry is used.

Select a file:

"/home/mcdonaghe/re§   Choose...

Upload csv to be processed

The csv file needs to have at least the required columns. Additional columns will be ignored. If your source of patient size data does not have either the height or the weight column, simply add a new empty column with just the title in the first row.

When you have selected the csv file, press the button to upload it.

3.7.2 Importing the size data to the database

On the next page select the column header that corresponds to each of the head, weight and ID fields. Also select whether the ID field is an Accession number or a Study UID:

When the column headers are selected, click the ‚Process the data' button.

The progress of the import is then reported on the patient size imports page:

During the import, it is possible to abort the process by clicking the button seen in the image above. The log file is available from the completed table whether it completed or not - there is no indication that the import was aborted.

As soon as the import is complete, the source csv file is deleted from the server.

3.7.3 Reviewing previous imports

After an import is complete, it is listed in the completed import tasks table. You can also get to this page from the Admin menu:

For each import, there is a link to the logfile, which looks something like this. With this import accession numbers weren't available so the patient size information was matched to the study instance UID:

```
Patient size import from sizeupload/2014/07/11/doctored.csv

1.3.12.2.1107.5.4.5.146226.30000012080207411271800000009:
    Height of 166.50 m not inserted as 166.5 cm already in the database
    Weight of 58.15 kg not inserted as 58.15 kg already in the database
1.3.51.0.1.1.192.168.90.77.100000611814.611849:
    Height of 165 m not inserted as 165 cm already in the database
    Weight of 87 kg not inserted as 87 kg already in the database
1.2.840.113704.1.111.5924.1371549177.10:
    Inserted height of 184 cm
    Inserted weight of 113 kg
1.2.840.113704.1.111.5000.1371472141.5:
    Inserted height of 166.10 cm
    Inserted weight of 95.50 kg
1.2.840.113704.1.111.5000.1371472199.6:
    Inserted height of 172 cm
    Inserted weight of 55 kg
```

3.7.4 Deleting import logs

The completed import tasks table also has a delete check box against each record and a delete button at the bottom. The csv file originally imported has already been deleted - this delete function is to remove the record of the import and the log file associated with it from the database/disk.

3.8 Adding patient size information from csv using the command line

Usage:

```
openrem_ptsizecsv.py [-h] [-u] [-v] csvfile id height weight
```

-h, --help
> Print the help text.

-u, --si-uid
> Use Study Instance UID instead of Accession Number.

-v, --verbose
> Print to the standard output the success or otherwise of inserting each value.

csvfile
> csv file containing the height and/or weight information and study identifier. Other columns will be ignored. Use quotes if the filepath has spaces.

id
> Column title for the accession number or study instance UID. Use quotes if the title has spaces.

height
> Column title for the patient height (DICOM size) - if this information is missing simply add a blank column with a suitable title. Use quotes if the title has spaces.

weight
> Column title for the patient weight - if this information is missing simply add a blank column with a suitable title. Use quotes if the title has spaces.

3.9 Fluroscopy high dose alerts
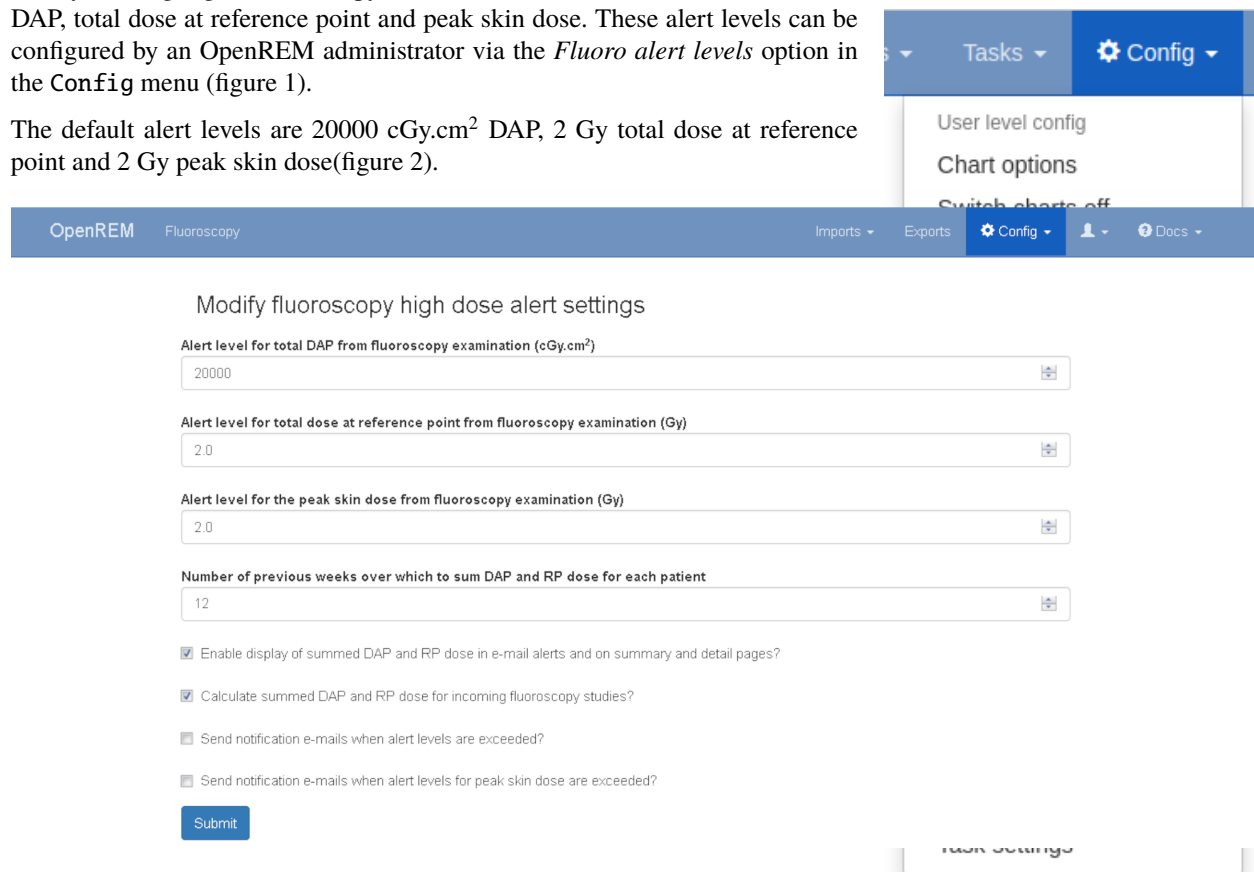
> **Inhalt**
>
> - *Fluroscopy high dose alerts*
>   - *Alert level configuration*
>   - *Alerts for cumulative dose over a period of time*
>   - *Recalculation of summed data*
>   - *E-mail notifications of high dose alerts*

3.9.1 Alert level configuration

The system highlights fluoroscopy studies that have exceeded defined levels of DAP, total dose at reference point and peak skin dose. These alert levels can be configured by an OpenREM administrator via the *Fluoro alert levels* option in the `Config` menu (figure 1).

The default alert levels are 20000 cGy.cm^2 DAP, 2 Gy total dose at reference point and 2 Gy peak skin dose(figure 2).

Abb. 15: Figure 1: The `Config` menu (user and admin)

Abb. 16: Figure 2: Fluoroscopy high dose alert settings

Figures 3 and 4 illustrate how studies that exceed an alert level are highlighted in the filtered and detailed fluoroscopy views.

Institution	Make Model Display name	Date	Study description Procedure Requested Procedure Accession number	Number of events	Total DAP (cGy.cm^2)	Total dose at RP (Gy)	Total DAP summed over 12 weeks before study(cGy.cm^2)	Total dose at RP summed over 12 weeks before study (Gy)	Physician	Delete?
Countryside County Hospital	Siemens AXIOM-Artis County Lab 1	2017-11-01 16:14	CARD None CARD 2F5JD0	230	9544.6	1.89	9544.6 (1 exam)	1.89 (1 exam)	D Foster	Delete
Countryside County Hospital	Siemens AXIOM-Artis County Lab 1	2017-10-27 08:56	CARD None CARD 4M0FKQ	272	8230.7	1.69	13508.0 (2 exams)	2.45 (2 exams)	D Foster	Delete
Countryside County Hospital	Siemens AXIOM-Artis County Lab 1	2017-10-23 17:40	CARD None CARD WUYVFV	117	7134.8	1.23	7134.8 (1 exam)	1.23 (1 exam)	D Foster	Delete
Countryside County Hospital	Siemens AXIOM-Artis County Lab 1	2017-10-23 15:59	CARD None CARD XRV69E	171	7224.5	1.32	7224.5 (1 exam)	1.32 (1 exam)	D Foster	Delete
Countryside County Hospital	Siemens AXIOM-Artis County Lab 1	2017-10-23 08:49	CARD None CARD COOMZ3	191	7887.6	1.79	7887.6 (1 exam)	1.79 (1 exam)	D Foster	Delete
Countryside County Hospital	Siemens AXIOM-Artis County Lab 1	2017-10-20 16:57	CARD None CARD GJ2TEH	195	10793.2	2.31	10793.2 (1 exam)	2.31 (1 exam)	D Foster	Delete

Abb. 17: Figure 3: Filtered view showing the highlighting of some high dose studies

3.9. Flluroscopy high dose alerts

Abb. 18: Figure 4: Detailed view showing high-dose highlighting
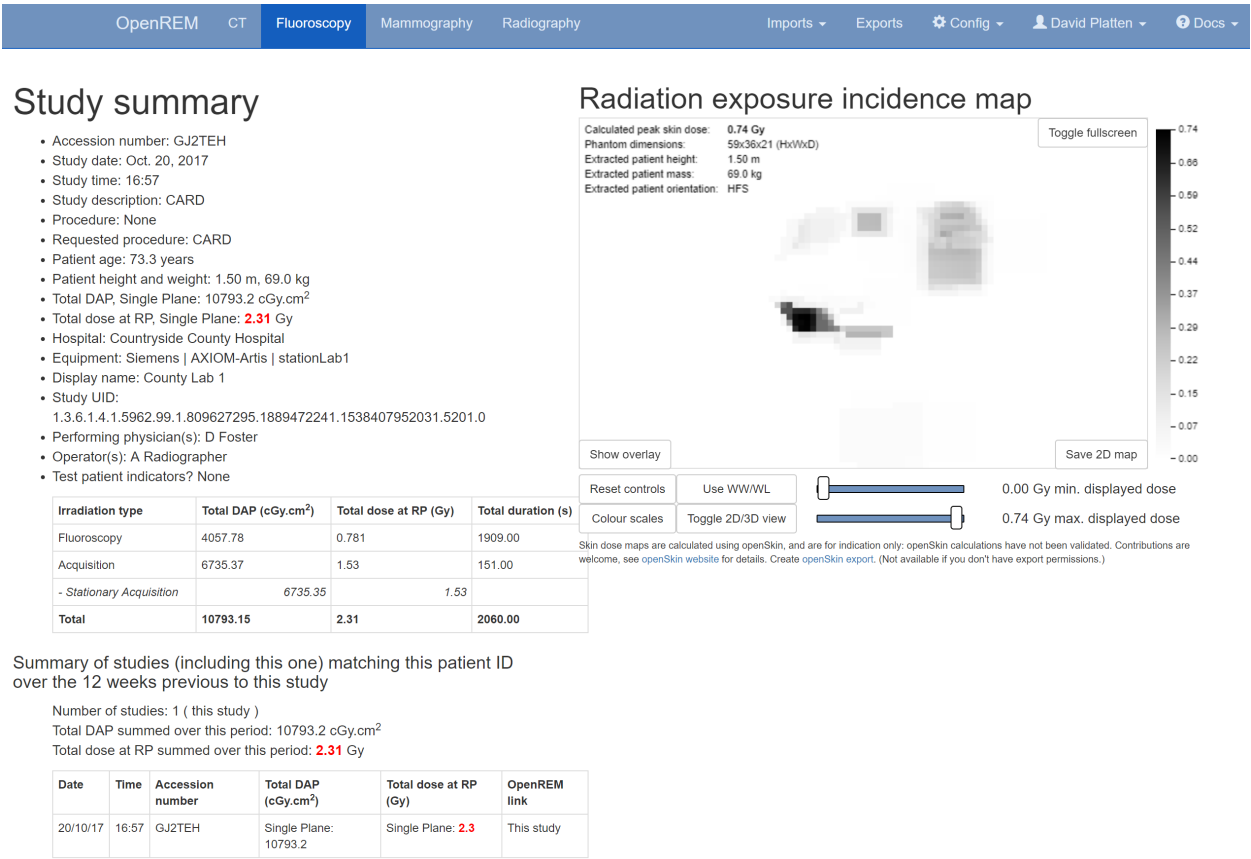
3.9.2 Alerts for cumulative dose over a period of time

As well as alerting to individual studies that exceed alert levels the system can be configured to calculate cumulative dose over a defined number of weeks for studies with matching patient IDs. When this is activated, for each study Open-REM looks for earlier fluoroscopy studies that have taken place that share the same patient ID, or encrypted patient ID, and sums the study DAP and total dose at reference point values. The time period that is used is configured by an Open-REM administrator, and defaults to 12 weeks (figure 2). This feature has not yet been implemented for the skin dose.

For this feature to work the storage of patient ID or encrypted patient ID must be enabled (see the *Patient identifiable data* documentation).

The configuration settings for this feature are (figure 2):

- The number of previous weeks over which to sum DAP and dose at RP for studies with matching patient ID is defined in the options

- The display of summed DAP and dose at RP values in the fluoroscopy filtered and detailed views, and in e-mail notifications

- The automatic calculation of summed DAP and dose at RP for new studies imported into OpenREM

An example of a study where there is another study with matching patient ID is shown below in figure 5. In this example neither of the two individual studies had doses that exceeded an alert level, but when summed together the total dose at RP does exceed the corresponding alert.

3.9.3 Recalculation of summed data

After upgrading from a version of OpenREM prior to 0.8.2, or after changing the alert levels or number of weeks to look for matching data, the summed dose values must be recalculated. The user is prompted to do this via the display of an orange button, as shown in figure 6 below. If settings have changed an information message is also displayed at the top of the screen.

Recalculation of the summed data is likely to take several minutes. During this time the form buttons are faded out and disabled, and a spinning icon is shown in the middle of the page (figure 7). The user must remain on this page until the calculations are complete.

Once all summed data has been recalculated the orange recalculate button is hidden, the other form buttons are reactivated and the user is shown a success message at the top of the screen (figure 8, below).

3.9.4 E-mail notifications of high dose alerts

For this feature to function the e-mail section in `local_settings.py` must be correctly completed (see the : ref :*email_configuration* documentation) and the e-mail server must allow sending of messages that originate from the OpenREM server, or from the authenticated user specified in the e-mail settings.

OpenREM users can be automatically sent e-mail notifications of studies that have exceeded a high dose alert level. This feature can be enabled or disabled by an OpenREM administrator on the *High dose alerts* configuration page as shown in figure 2 above.

Alert recipients users are chosen by navigating to the *Fluoro alert notifcation* page via the *Config* menu. Figure 9 shows an example of the notification page.
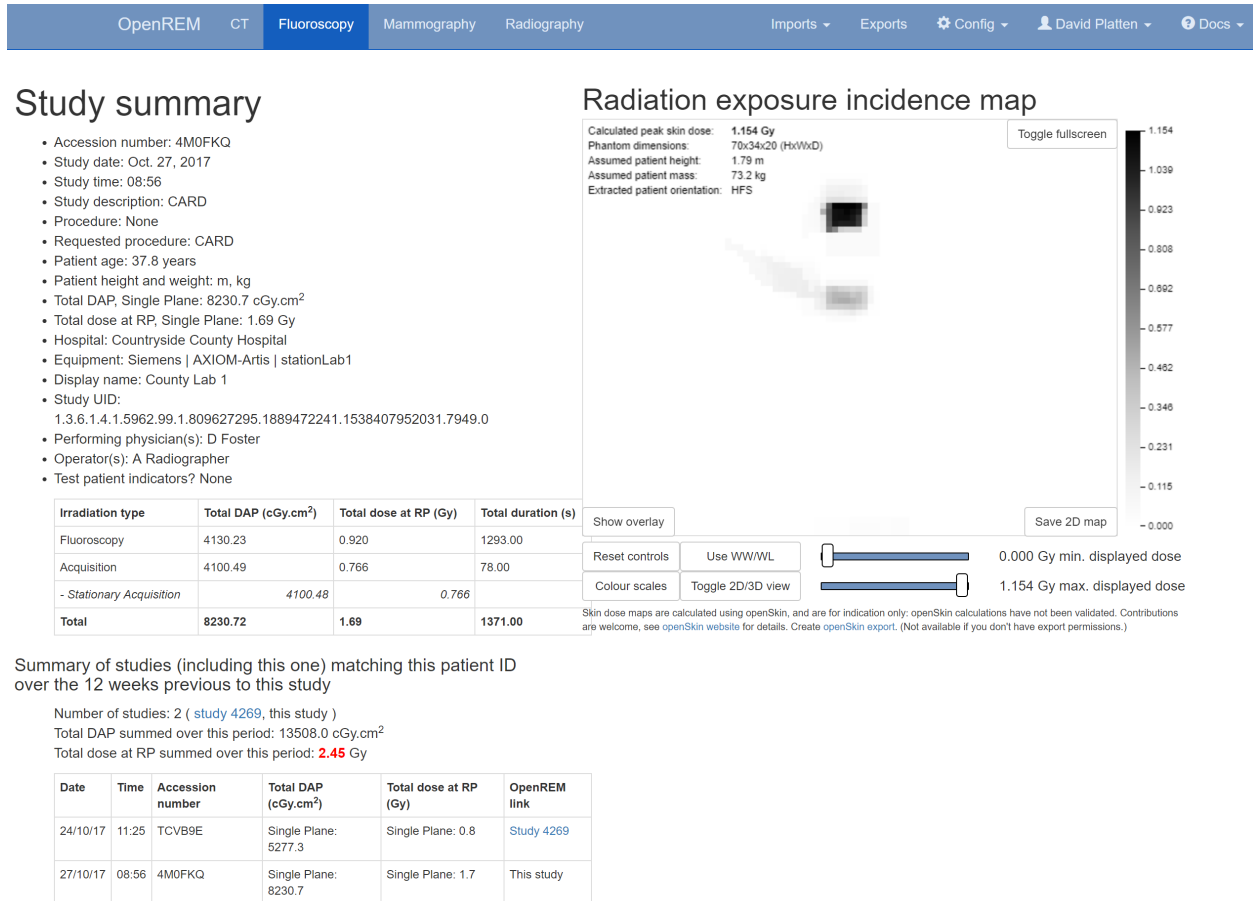
Abb. 19: Figure 5: Detailed view showing associated studies over a time period

Abb. 20: Figure 6: Prompt to recalculate the summed dose values

Abb. 21: Figure 7: Prompt to recalculate the summed dose values

Abb. 22: Figure 8: Message on successful recalculation

It should be noted that any OpenREM user selected to receive high dose alerts must have an e-mail address entered in their user profile.

3.10 Task management

Inhalt

- *Task management*
  - *Viewing task and service statuses*
  - *Terminating running tasks*
  - *Configuring the size of task history*

User name	First name	Last name	E-mail address	Receive high dose alert e-mails?
david	David		david@work.com	☑
demo	Demo	Demo		☐
luuk	Luuk		luuk@work.com	☑
testingadmin				☐
tim	Tim		tim@work.com	☐

Check one or more boxes and click an Update button to select which users receive e-mail notification of fluoroscopy high dose alerts.

The table can be sorted by clicking on the column headers.

Update

OpenREM version 0.8.1 is © 2013-2018 The Royal Marsden NHS Foundation Trust

Abb. 23: Figure 9: E-mail user-notification of high-dose alerts

3.10.1 Viewing task and service statuses

Users who are logged in with admin rights can use the **Tasks** menu and choose **All tasks** to see the following:

- A list of the tasks currently being executed

- A list of previous tasks and their final status. If any errors occurred they will be displayed here.
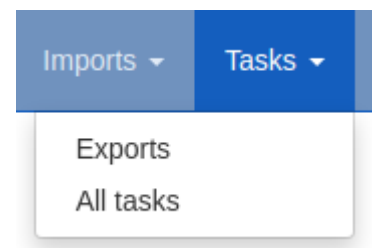
3.10.2 Terminating running tasks

It is possible to terminate any active tasks by clicking the red button. **There is no confirmation step**. Note that this immediately interrupts everything this process was doing so far, leading to things like partially imported studies. In general this should not be an issue (in case of aborted imports they should be completed when you start importing them again), but note that there is a certain risk in killing tasks and use this only as a last resort.

Abb. 24: Figure 1: The `Tasks` menu

A note on move: executing a move will create a task which then produces import tasks for all the studies it should import. This means if you intend to abort a move you should abort the task with Task type „move" and not the import tasks started by that process!

3.10.3 Configuring the size of task history

The status of 2000 active, recent and older tasks are stored in the OpenREM database. This limit can be altered by users who are logged in with admin rights by clicking on **Task settings** in the **Config** menu and changing the current value. If this limit is set to a very high value it can cause the web browser to run out of memory when trying to view the **Task** page due to the large number of rows in the tables.

Active tasks

UUID	Task type	Info	Error	Started	State	Abort Task
994f26d3-62a8-4a35-bcff-f70026adee55	move	None	None	9. Mai 2022 09:35	Running	Terminate task

Recent tasks

UUID	Task type	Info	Error	Started ▲	State
7d906035-369b-46f0-a7d4-9e2a3fb3d5ee	import_dx	UID: 1. 3. 6. 1. 4. 1. 5962. 99. 1. 2282339064. 1266597797. 1479751121656. 24. 0	Study 1. 3. 6. 1. 4. 1. 5962. 99. 1. 2282339064. 1266597797. 1479751121656. 24. 0 already in DB	9. Mai 2022 09:34	Failure
b8e28f0e-303c-4430-8058-58c537e86f54	query	None	None	9. Mai 2022 09:34	Success
8c289685-4f37-4a02-ab6b-9347abca0d80	import_dx	UID: 1. 2. 276. 0. 7230010. 3. 1. 2. 8323329. 28078. 1624372528. 66982	None	9. Mai 2022 09:34	Success
5fa15a01-77a6-4d78-9d66-87d36c482c16	import_dx	UID: 1. 2. 276. 0. 7230010. 3. 1. 2. 8323329. 11564. 1483691867. 34530	Study 1. 2. 276. 0. 7230010. 3. 1. 2. 8323329. 11564. 1483691867. 34530 already in DB	9. Mai 2022 09:34	Failure
4158d1ad-a299-439e-a1ac-b444aaf6b113	import_dx	UID: 1. 2. 276. 0. 7230010. 3. 1. 2. 8323329. 11564. 1483691867. 34530	Study 1. 2. 276. 0. 7230010. 3. 1. 2. 8323329. 11564. 1483691867. 34530 already in DB	9. Mai 2022 09:34	Failure
4e49620c-4844-480a-9bb0-3f239348e184	import_dx	UID: 1. 3. 6. 1. 4. 1. 5962. 99. 1. 2282339064. 1266597797. 1479751121656. 24. 0	Study 1. 3. 6. 1. 4. 1. 5962. 99. 1. 2282339064. 1266597797. 1479751121656. 24. 0 already in DB	9. Mai 2022 09:33	Failure
6afa3966-1bb4-44e3-9fb5-744d81a3211e	import_dx	UID: 1. 3. 6. 1. 4. 1. 5962. 99. 1. 2282339064. 1266597797. 1479751121656. 24. 0	Study 1. 3. 6. 1. 4. 1. 5962. 99. 1. 2282339064. 1266597797. 1479751121656. 24. 0 already in DB	9. Mai 2022 09:33	Failure
06028fb6-e200-4fa0-983d-f71339d87b6f	import_mam	UID: 1. 3. 6. 1. 4. 1. 5962. 99. 1. 1270844358. 1571783457. 1525984267206. 3. 0	None	9. Mai 2022 09:33	Success

Abb. 25: Figure 2: The task administration page

Importing data to OpenREM

4.1 From local DICOM files

If you have RDSRs or RRDSRs, DX, MG, PET or NM images or Philips CT Dose Info images, you can import them directly into OpenREM:

4.1.1 Importing from DICOM files

If you are using linux, or for Windows if you have put `C:\Python27\;C:\Python27\Lib\site-packages;C:\Python27\Scripts` onto your system path, you should be able to import from the command line:

Radiation Dose Structured Reports

```
openrem_rdsr.py filename.dcm
```

You can use wildcards to process a number of files at once, ie:

```
openrem_rdsr.py *.dcm
```

Cumulative and continued study RDSRs

Background

Cumulative RDSRs

Some modalities are configured to send an RDSR after every exposure, with each new RDSR containing a complete record of the examination up to that point. For example, this is what the current version of the Siemens CT scanner software does.

Continued study RDSRs

On most systems the RDSR is sent when the study is completed. If the study is then restarted, the system must create a new RDSR. On a Siemens CT system, this new RDSR will have the same Study Instance UID and the same accession number, but the content will only refer to the continued study, not the original study.

Pre-0.8.0 OpenREM behaviour

Prior to release 0.8.0, OpenREM would check the Study Instance UID on import and check the value against the existing studies in the database. If a match was found, then the new RDSR was rejected on the basis that it must be a duplicate.

This would therefore ignore both cumulative and continued study RDSRs which means your database might be filled with single event studies, and you won't have details of any continued studies.

Current OpenREM behaviour

New imports

On import of the first RDSR in a study, the SOP Instance UID of the RDSR is recorded with the study. This is an ID that is unique to that RDSR object - any further RDSRs might have the same Study Instance UID, but will always have a different SOP Instance UID.

When the second RDSR is imported, the duplicate StudyInstanceUID will trigger OpenREM to check the SOP Instance UID of the new RDSR against the one(s) stored with that study. If there is a match, the new RDSR is ignored as it has already been processed. If it does not match, then the Irradiation Event UID of each exposure in the new RDSR is compared to the Irradiation Event UIDs already in the database for that study, to establish if the new RDSR carries new information that should be imported.

In the case of a cumulative RDSR that is sent after each event, the original study is deleted from the database and is replaced by the newer one if it has additional events.

In the case of a continued study RDSR which has a completely different set of events, the new RDSR is imported alongside the existing one.

Existing studies imported before 0.8.0

RDSRs imported before upgrading to 0.8.0 will not have the SOP Instance UID recorded in the database and so the new RDSR will be compared at event level with the existing study before making an import decision, as with new studies.

Fixing existing studies

Importing from file

If you are have a store of the RDSRs that were previously rejected, import them all again and this time they should be processed properly.

For example on my system, using linux, each scanner started sending per-exposure RDSRs from the date they were upgraded. I found the RDSRs from that date to the date I upgraded OpenREM and imported them:

```
touch --date "2018-01-06" tmpdate20180106
touch --date "2018-02-07" tmpdate20180207
find RDSRs/ -newer tmpdate20180106 ! -newer tmpdate20180207 -name *.dcm -exec openrem_
↪rdsr.py {} \;
```

Importing via query-retrieve

The query-retrieve duplicates processing has been updated to compare SOP Instance UIDs returned by the remote node (the PACS) with the SOP Instance UIDs stored with each study in OpenREM. Therefore, after an initial import of each RDSR in your search, any subsequent query should drop any RDSRs that have previously been processed and not move them a second time.

Radiopharmaceutical Radiation Dose Structured Reports

You can use the same import script as for Radiation Dose Structured Reports.

For mammography DICOM images

```
openrem_mg.py filename.dcm
```

The facility for extracting dose information from mammography DICOM images has been designed and tested with images created with the GE Senographe DS. It has now also been used with the images generated by the following systems:

- GE Senographe Essential
- Hologic Selenia
- Siemens Inspiration

For radiographic DICOM images

```
openrem_dx.py filename.dcm
```

For PET/NM DICOM images

```
openrem_nm.py filename.dcm
```

Note that more complete information can be loaded from the RRDSRs if available. For PET images the PET series information can be added to the RRDSR data.

For CT dose summary files from Philips CT scanners

```
openrem_ctphilips.py filename.dcm
```

This extractor makes use of the information stored in the header data of the Philips Secondary Capture object with a series description of ‚Dose Info'. The value inserted into ‚Study description' in the OpenREM database is actually taken from the Protocol field. The value in Study description is inserted into the study level comment field in the database, along with the protocol file name and any ‚comments on radiation dose'.

For CT dose summary files from older Toshiba CT scanners

```
openrem_cttoshiba.py path_to_files
```

This extractor is designed to create a DICOM radiation dose structured report from the information contained in secondary capture dose summary images, supplemented by data stored in image tags. It requires a folder of DICOM objects as input (suitable data can be retrieved from a DICOM node using the `qrscu.py` command with the `-toshiba` switch - see *Query-retrieve using the command line interface*). It creates an initial RDSR from the secondary capture dose summary, and then tries to enrich this with additional information contained in image tags. The routine attempts to extract the following information from the image tags and insert it into the initial RDSR:

Study-level information

- Study description
- Requested procedure description
- Software versions
- Device serial number

Series-level information

- Protocol name
- Exposure time (per rotation)
- kVp
- Spiral pitch factor
- Nominal total collimation width
- Nominal single collimation width
- Exposure modulation type

The routine was developed for older Toshiba CT scanners that cannot create RDSR objects themselves. It is known to work with:

- Toshiba CX, software version V4.40ER011
- Toshiba CXL, software version V4.51ER014
- Toshiba CXL, software version V4.86ER008 (this software version can produce RDSR objects directly, but may not populate some fields, such as requested procedure name and study description)

This extractor has also been used successfully on images from a GE LightSpeed Plus scanner, although in this case no supplementary data is extracted from image tags.

If you want some examples, you can find the DICOM files that we use for the automated testing in the `openrem/ remapp/tests/test_files` folder in your OpenREM installation.

4.2 Direct from modalities

For production use, you will either need the modalities to send the RDSR or images directly to your OpenREM server using DICOM, or you will need to use query-retrieve to fetch the DICOM objects from the PACS or the modalities. In either of these situations, you will need to run a DICOM Store service on your OpenREM server.

4.2.1 DICOM Network Configuration

Configuring DICOM store nodes in OpenREM

You need to configure details of the DICOM store node to enable the query-retrieve functionality. You will also need to have installed Orthanc or an alternative:

- Orthanc enabled by default in Docker

- Installed in the Linux instructions and configured at *DICOM Store SCP*

- Installed in the Windows instructions and configured at *to be written*

To configure a DICOM Store SCP, on the `Config` menu select `DICOM networking`, then click `Add new Store` and fill in the details (see figure 1):

- Name of local store node: This is the *friendly name*, such as `OpenREM store`

- Application Entity Title of the node: This is the DICOM name for the store, and must be letters or numbers only, no spaces, and a maximum of 16 characters

- Port for store node: Port 104 is the reserved DICOM port, but it is common to use *high* ports such as 8104, partly because ports up to 1024 usually need more privileges than for the high ports. However, if there is a firewall between the remote nodes (modalities, PACS) and the OpenREM server, then you need to make sure that the firewall is configured to allow the port you choose here

Status of DICOM Store SCP nodes

DICOM Store SCP advanced configuration

DICOM Store SCP nodes that have been configured are listed in the left column of the DICOM network configuration page. For each server, the basic details are displayed, including the Database ID which is required for command line/scripted use of the query-retrieve function.

In the title row of the Store SCP config panel, the status will be reported either as ‚Server is alive' or ‚Error: Association fail - server not running¿ - see figure 3

Abb. 1: Figure 1: DICOM Store SCP configuration
Abb. 2: Figure 3: DICOM Store SCP status - Alive and Association failed

Query retrieve of third-party system, such as a PACS or modality

To Query-Retrieve a remote host, you will need to configure both a local Store SCP and the remote host.

To configure a remote query retrieve SCP, on the `Config` menu select `DICOM networking`, then click `Add new QR Node` and fill in the details:

- Name of QR node: This is the *friendly name*, such as `PACS QR`

- AE Title of the remote node: This is the DICOM name of the remote node, 16 or fewer letters and numbers, no spaces

- AE Title this server: This is the DICOM name that the query (DICOM C-Find) will come from. This may be important if the remote node filters access based on *calling aet*. Normal rules of 16 or fewer letters and numbers, no spaces

- Remote port: Enter the port the remote node is using (eg 104)

- Remote IP address: The IP address of the remote node, for example `192.168.1.100`

- Remote hostname: Alternatively, if your network has a DNS server that can resolve the hostnames, you can enter the hostname instead. If the hostname is entered, it will be used in preference to the IP address, so only enter it if you know it will be resolved.

Now go to the *DICOM Query Retrieve Service* documentation to learn how to use it.

Troubleshooting: openrem_store.log

If the default logging settings haven't been changed then there will be a log files to refer to. The default location is within your `MEDIAROOT` folder:

This file contains information about each echo and association that is made against the store node, and any objects that are sent to it.

The following is an example of the log for a Philips *dose info* image being received:

```
[21/Feb/2016 21:13:43] INFO [remapp.netdicom.storescp:310] Starting AE...␣
→AET:MYSTOREAE01, port:8104
[21/Feb/2016 21:13:43] INFO [remapp.netdicom.storescp:314] Started AE... AET:MYSTOREAE01,
→ port:8104
[21/Feb/2016 21:13:43] INFO [remapp.netdicom.storescp:46] Store SCP: association␣
→requested
[21/Feb/2016 21:13:44] INFO [remapp.netdicom.storescp:54] Store SCP: Echo received
[21/Feb/2016 21:13:46] INFO [remapp.netdicom.storescp:46] Store SCP: association␣
→requested
[21/Feb/2016 21:13:46] INFO [remapp.netdicom.storescp:54] Store SCP: Echo received
[21/Feb/2016 21:13:49] INFO [remapp.netdicom.storescp:46] Store SCP: association␣
→requested
[21/Feb/2016 21:13:49] INFO [remapp.netdicom.storescp:54] Store SCP: Echo received
[21/Feb/2016 21:13:50] INFO [remapp.netdicom.storescp:46] Store SCP: association␣
→requested
[21/Feb/2016 21:13:50] INFO [remapp.netdicom.storescp:54] Store SCP: Echo received
[21/Feb/2016 21:13:51] INFO [remapp.netdicom.storescp:46] Store SCP: association␣
→requested
[21/Feb/2016 21:13:51] INFO [remapp.netdicom.storescp:54] Store SCP: Echo received
[21/Feb/2016 21:14:39] INFO [remapp.netdicom.storescp:46] Store SCP: association␣
→requested
[21/Feb/2016 21:14:39] INFO [remapp.netdicom.storescp:78] Received C-Store. Stn name NM-
```

```
→54316, Modality CT,
SOPClassUID Secondary Capture Image Storage, Study UID 1.2.840.113564.9.1.2843752344.47.
→2.5000947881 and Instance
UID 1.2.840.113704.7.1.1.4188.1234134540.349
[21/Feb/2016 21:14:39] INFO [remapp.netdicom.storescp:232] File
/var/openrem/media/dicom_in/1.2.840.113704.7.1.1.4188.1453134540.349.dcm written
[21/Feb/2016 21:14:39] INFO [remapp.netdicom.storescp:263] Processing as Philips Dose␣
→Info series
...etc
```

4.2.2 DICOM Store

The Orthanc DICOM server is recommended; another store can be used instead but documentation is not provided. Docker installs have the Orthanc server build-in. For non-Docker installs, instructions are included in the main installation documentation:

- Linux: *DICOM Store SCP*
- Windows: *to be written*

4.3 Query-retrieve from a PACS or similar

Before you can query-retrieve objects from a remote PACS, you need to do the following:

- Create a DICOM Store service to receive the DICOM objects - see *Direct from modalities* above.
- Configure OpenREM with the settings for the remote query-retrieve server:

4.3.1 Configuration required for query-retrieve

You need a DICOM store service set up - see *Importing data to OpenREM* for details.

If you are using a third party DICOM Store server, then you will need to add the details as per *DICOM Network Configuration* but do not use the ‚advanced‘ section.

To configure a remote query retrieve SCP, on the `Config` menu select `DICOM networking`, then click `Add new QR Node` and fill in the details:

  – Name of QR node: This is the *friendly name*, such as `PACS QR`

  – AE Title of the remote node: This is the DICOM name of the remote node, 16 or fewer letters and numbers, no spaces

  – AE Title this server: This is the DICOM name that the query (DICOM C-Find) will come from. This may be important if the remote node filters access based on *calling aet*. Normal rules of 16 or fewer letters and numbers, no spaces

  – Remote port: Enter the port the remote node is using (eg 104)

  – Remote IP address: The IP address of the remote node, for example `192.168.1.100`

  – Remote hostname: Alternatively, if your network has a DNS server that can resolve the hostnames, you can enter the hostname instead. If the hostname is entered, it will be used in preference to the IP address, so only enter it if you know it will be resolved.

– Use Modality in Study Query: Some PACS systems (like Impax 6.6) need modality at study level for correct filtering. if this option is checked, the modality tag is inserted in the study level request.

> **Warnung:** Modality is not a valid tag in a study level request (Modalities In Study is available instead). However, some PACS systems require it for proper function, others will ignore it, and some will return zero results if the tag is present.

- Configure the settings of your DICOM store service on the PACS

- Learn how to use it:

4.3.2 DICOM Query Retrieve Service

To query retrieve dose related objects from a remote server, you need to review the *DICOM Network Configuration* documents first to make sure you have created a DICOM Store node installed and configured which will import objects to OpenREM.

You will also need to set up the remote server to allow you to query-retrieve using it - the remote server will need to be configured with details of the store node that you have configured.

Query-retrieve using the web interface

– On the Imports menu, select `Query remote server` - see figure 1. If the menu isn't there, you need to check your user permissions – see *Configure the settings* for details.

– Each configured query-retrieve node and each local store node is automatically tested to make sure they respond to a DICOM echo - the results are presented at the top of the page. See figure 2 for an example.
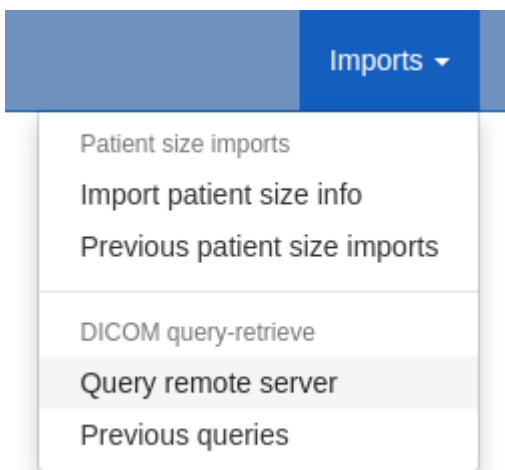
Abb. 3: Figure 1: Import Query-Retrieve menu

– Select the desired **remote host**, ie the PACS or modality you wish to query.

– Select the local **store node** you want to retrieve to.

– Select **which modalities** you want to query for - at least one must be ticked.

- Select a **date range** - the wider this is, the more stress the query will place on the remote server, and the higher the likelyhood of the query being returned with zero results (a common configuration on the remote host to prevent large database queries affecting other services). Defaults to ‚from yesterday‘.

- If you wish to **exclude studies** based on their study description, enter the text here. Add several terms by separating them with a comma. One example would be to exclude any studies with `imported` in the study description, if your institution modifies this field on import. The matching is case-insensitive.

- Alternatively, you might want to only **keep studies** with particular terms in the study description. If so, enter them in the next box, comma separated.

- You can also **exclude studies by station name**, or only keep them if they match the station name. This is only effective if the remote system (the PACS) supports sending back station name information. By default, this is only checked against series level responses *(changed in OpenREM 1.0)*.

Advanced query options

- **Attempt to get Toshiba dose images** *default not ticked*: If you have done the extra installation and configuration required for creating RDSRs from older Toshiba scanners, then you can tick this box for *CT* searches to get the images needed for this process. See the logic description below for details.

- **Ignore studies already in the database** *default ticked*: By default OpenREM will attempt to avoid downloading any DICOM objects (RDSRs or images) that have already been imported into the database. Untick this box to override that behaviour and download all suitable objects. See the logic description below for details.

- **Include SR only studies** *default not ticked*: If you have a DICOM store with only the radiation dose structured reports (RDSR) in, or a mix of whole studies and RDSRs without the corresponding study, then tick this box. Any studies with images and RDSRS will be ignored (they can be found without this option). If this box is ticked any modality choices will be ignored.

- **Get SR series that return nothing at image level query** *default not ticked*: If you have a DICOM store with SR series that you know contain RDSR objects, but when queried your store says they are empty, then check this box. If this behaviour is found, a message will be added to the `openrem_qr.log` at `INFO` level with the phrase `Try '-emptysr' option`?. With the box checked the query will assume any SR series found contains an RDSR. Warning: with this behavior, any non-RDSR structured report series (such as a radiologists report encoded as a structured report) will be retrieved instead of images that could actually be used (for example with mammography and digital radiographs). Therefore this option should be used with caution!

- **Check station name include/exclude at study level** *default not ticked*: Change this setting to enable checking of station name include/exclude at study level instead of series level. This addresses issue #772 as some studies will have different station name information at study level than at series level - if both levels are checked it is impossible to get the desired response.

When you have finished the query parameters, click `Submit`

Review and retrieve

Once all the responses have been purged of unwanted modalities, study descriptions or study UIDs, the number of studies of each type will be displayed and a button appears. Click `Retrieve` to request the remote server send the selected objects to your selected Store node. This will be based on your original selection - changing the node on the left hand side at this stage will have no effect.

The progress of the retrieve is displayed in the same place until the retrieve is complete. You can also see the query and start the `Retrieve` in the *DICOM query summary*.

Query-retrieve using the command line interface

Running the command in different environments

Docker: In a command window/shell, navigate to the folder containing `docker-compose.yml` etc. Then precede the command with `docker-compose exec openrem`:

```
$ docker-compose exec openrem openrem_qr.py -h
```

Linux: Activate the virtualenv - assuming default Ubuntu install:

```
$ . /var/dose/veopenrem3/bin/activate
$ openrem_qr.py -h
```

Windows: Activate the virtualenv - **docs to be written** - and command might need the full path?:

```
> C:\OpenREM\veopenrem3\Scripts\activate.bat
> C:\OpenREM\veopenrem3\Scripts\openrem_qr.py -h
```

```
usage: openrem_qr.py [-h] [-ct] [-mg] [-fl] [-dx] [-nm]
                     [-f yyyy-mm-dd] [-t yyyy-mm-dd] [-sd yyyy-mm-dd]
                     [-tf hhmm] [-tt hhmm]
                     [-e string] [-i string]
                     [-sne string] [-sni string] [--stationname_study_level]
                     [-toshiba] [-sr] [-dup] [-emptysr]
                     qr_id store_id

Query remote server and retrieve to OpenREM

positional arguments:
  qr_id                 Database ID of the remote QR node
  store_id              Database ID of the local store node

options:
  -h, --help            show this help message and exit
  -ct                   Query for CT studies. Cannot be used with -sr
  -mg                   Query for mammography studies. Cannot be used with -sr
  -fl                   Query for fluoroscopy studies. Cannot be used with -sr
  -dx                   Query for planar X-ray studies (includes panoramic X-ray␣
→studies). Cannot be used with -sr
  -nm                   Query for nuclear medicine studies. Cannot be used with -sr
```

```
 -f yyyy-mm-dd, --dfrom yyyy-mm-dd
                         Date from, format yyyy-mm-dd. Cannot be used with --single_
↪date
 -t yyyy-mm-dd, --duntil yyyy-mm-dd
                         Date until, format yyyy-mm-dd. Cannot be used with --single_
↪date
 -sd yyyy-mm-dd, --single_date yyyy-mm-dd
                         Date, format yyy-mm-dd. Cannot be used with --dfrom or --
↪duntil
 -tf hhmm, --tfrom hhmm
                         Time from, format hhmm. Requires --single_date.
 -tt hhmm, --tuntil hhmm
                         Time until, format hhmm. Requires --single_date.
 -e string, --desc_exclude string
                         Terms to exclude in study description, comma separated,
↪quote whole string
 -i string, --desc_include string
                         Terms that must be included in study description, comma
↪separated, quote whole string
 -sne string, --stationname_exclude string
                         Terms to exclude in station name, comma separated, quote
↪whole string
 -sni string, --stationname_include string
                         Terms to include in station name, comma separated, quote
↪whole string
 --stationname_study_level
                         Advanced: Filter station name at Study level, instead of at
↪Series level
 -toshiba                Advanced: Attempt to retrieve CT dose summary objects and
↪one image from each series
 -sr                     Advanced: Use if store has RDSRs only, no images. Cannot be
↪used with -ct, -mg, -fl, -dx
 -dup                    Advanced: Retrieve duplicates (objects that have been
↪processed before)
 -emptysr                Advanced: Get SR series that return nothing at image level
↪query
```

As an example, if you wanted to query the PACS for DX images on the 5th and 6th April 2010 with any study descriptions containing `imported` excluded, first you need to know the database IDs of the remote node and the local node you want the images sent to. To find these, go to the *DICOM Network Configuration* page where the database ID is listed among the other details for each node.

Assuming the PACS database ID is 2, and the store node ID is 1, the command would look something like:

```
$ docker-compose exec openrem openrem_qr.py 2 1 -dx -f 2010-04-05 -t 2010-04-06 -e
↪"imported"
```

If you want to do this regularly to catch new studies, you might like to use a script something like this on Linux - make sure you comment out or delete one of the options, and amend as necessary!

```
#!/bin/bash

ONEHOURAGO=$(date -d "1 hour ago" "+%Y-%m-%d")
```

```
# Docker on Linux
/usr/local/bin/docker-compose -f /path/to/docker-compose.yml exec -T openrem␣
↪openrem_qr.py 2 1 -dx -f $ONEHOURAGO -t $ONEHOURAGO  -e "Imported"
# Linux
/var/dose/veopenrem3/bin/python /var/dose/veopenrem3/bin/openrem_qr.py 2 1 -dx -f
↪$ONEHOURAGO -t $ONEHOURAGO  -e "Imported"
```

This script could be run once an hour using a cron job. By asking for the date an hour ago, you shouldn't miss exams taking place in the last hour of the day. As the script won't run from the folder containing `docker-compose.yml` the location of that file needs to be passed to `docker-compose` with the `-f` option. You can check the path to `docker-compose` on your system using `which docker-compose`.

A similar script could be created as a batch file or PowerShell script on Windows and run using the scheduler. An example PowerShell script is shown below:

```
# Script to obtain all CT studies from a DICOM node on the day prior to the
# date the script is run and import them into OpenREM.
# Get yesterday's date

$dateString = "{0:yyyy-MM-dd}" -f (get-date).AddDays(-1)

# Run the openrem_qr.py script with yesterday's date as the to and from date

# Docker on Windows
docker-compose -f C:\Path\To\docker-compose.yml exec -T openrem openrem_qr.py 2 1 -
↪ct -f $dateString -t $dateString
# Windows
C:\OpenREM\veopenrem3\Scripts\python.exe C:\OpenREM\veopenrem3\Scripts\openrem_qr.
↪py 2 1 -dx -f $ONEHOURAGO -t $ONEHOURAGO  -e "Imported"
```

The above PowerShell script could be run on a regular basis by adding a task to the Windows `Task Scheduler` that executes the `powershell` program with an argument of `-file C:\path\to\script.ps1`.

Querying with time range

It is now possible to query for studies in a time window when using query-retrieve from the command line (web interface version will be introduced later). This can be particularly useful where PACS query responses are limited or null if the query matches too many studies.

Using the `--tfrom`/`-tf` and/or the `--tuntil`/`-tt` arguments are only allowed if `--single_date`/`-sd` argument is used.

Note: `-sd 2018-03-19` is the same as using `-f 2018-03-19 -t 2018-03-19`, and can be used without the time arguments.

- `-tf` used without `-tt` will search from `tf` until 23.59 that day.

- `-tt` used without `-tf` will search from 00.00 to `tt` that day.

- `-tf` and `-tt` used together will search from `tf` to `tt`.

For example, to search for CT from 12 noon to 3pm on 19th March 2018, using remote QR node database ID 2 and local store database ID 1:

```
$ # Using Docker on Linux
$ docker-compose exec openrem openrem_qr.py 2 1 -ct -sd 2018-03-19 -tf 1200 -tt 1500
```

Query filtering logic

Study level query response processing

– First we query for each modality chosen in turn to get matching responses at study level.

– If the optional `ModalitiesInStudy` has been populated in the response, and if you have ticked `Include SR only studies`, then any studies with anything other than just SR studies is removed from the response list.

– If any study description filters have been added, and if the `StudyDescription` tags are returned by the remote server, the study response list is filtered accordingly. The same applies to the station name filter *if* the option to check station names at study level has been selected.

– For the remaining study level responses, each series is queried.

– If `ModalitiesInStudy` was not returned, it is now built from the series level responses.

– If the remote server returned everything rather than just the modalities we asked for, the study level responses are now filtered against the modalities selected.

Series level query processing

– If station name filters have been added, and if the `StationName` tags are returned by the remote server, the series list is filtered accordingly — unless the option to check station names at study level has been selected.

If **mammography** exams were requested, and a study has `MG` in:

– If one of the series is of type SR, an image level query is done to see if it is an RDSR. If it is, all the other series responses are deleted (i.e. when the move request/'retrieve' is sent only the RDSR is requested not the images.

– Otherwise the SR series responses are deleted and all the image series are requested.

If **planar radiographic** exams were requested, and a study has `DX` or `CR` in:

– Any SR series are checked at 'image' level to see if they are RDSRs. If they are, the other series level responses for that study are deleted.

– Otherwise the SR series responses are deleted and all the image series are requested.

If **fluoroscopy** exams were requested, and a study has `RF` or `XA` in:

– Any SR series are checked at 'image' level to see if they are RDSRs or ESRs (Enhanced Structured Reports - not currently used but will be in the future). Any other SR series responses are deleted.

– All non-SR series responses are deleted.

If **CT** exams were requested, and a study has `CT` in:

– Any SR series are checked at 'image' level to see if they are RDSRs. If they are, all other SR and image series responses are deleted. Otherwise, if it has an ESR series, again all other SR and image series responses are deleted.

– If there are no RDSR or ESR series, the other series are checked to see if they are Philips 'Dose info' series. If there are, other series responses are deleted.

- If there are no RDSR, ESR or ‚Dose info' series and the option to get Toshiba images has been selected, then an image level query is performed for the first image in each series. If the image is not a secondary capture, all but the first image are deleted from the image level responses and the image_level_move flag is set. If the image is a secondary capture, the whole series response is kept.

- If there are no RDSR or ESR, series descriptions aren't returned and the Toshiba option has been set, the image level query is performed as per the previous point. This process will keep the responses that might have Philips ‚Dose info' series.

- If there are no RDSR, ESR, series descriptions aren't returned and the Toshiba option has not been set, each series with more than five images in is deleted from the series response list - the remaining ones might be Philips ‚Dose info' series.

If **SR only studies** were requested:

- Each series response is checked at ‚image' level to see which type of SR it is. If is not RDSR or ESR, the study response is deleted.

If **Get SR series that return nothing at image level query** were requested:

- It is assumed that any SR series that appears to be empty actually contains an RDSR, and the other series are dealt with as above for when an RDSR is found. If at the image level query the full data requested is returned, then the series will be processed the same whether this option is selected or not.

Duplicates processing

For each remaining study in the query response, the Study Instance UID is checked against the studies already in the OpenREM database.

If there is a match and the series level modality is **SR** (from a CT, or RF etc):

- The image level response will have the SOP Instance UID - this is checked against the SOP Instance UIDs recorded with the matching study. If a match is found, the ‚image' level response is deleted.

If there is a match and the series level modality is **MG**, **DX** or **CR**:

- An image level query is made which will populate the image level responses with SOP Instance UIDs

- Each image level response is then processed and the SOP Instance UID is checked against the SOP Instance UIDs recorded with the matching study. If a match is found, the ‚image' level response is deleted.

Once each series level response is processed:

- If the series no longer has any image level responses the series level response is deleted.

- If the study no longer has any series level responses the study level response is deleted.

DICOM query summary

Current DICOM SCP statuses

Remote QR nodes
Conquest	✔ responding to DICOM echo
Test PACS node	❶ not responding to DICOM echo

Local Store nodes
Test Node	❶ not responding to DICOM echo
Test AutoStart Node	✔ responding to DICOM echo
conquest	✔ responding to DICOM echo

Query retrieve dialogue

Remote host field*	Store scp field*
Conquest ▾	Test AutoStart Node ▾

Abb. 4: Figure 2: Local and remote QR statuses

Either by clicking on the „Go to query details page" when executing a query or by going to Config > DICOM query summary you can review the current and older queries, check which files were found on the remote, which studies/files were ignored and why, and review the result of importing files which were retrieved.

Abb. 6: Figure 4: The query details page

By clicking on the studies of a query you can review the discovered DICOM series as well as to some extent the individual files that are part of those series. If no import tasks are shown, even though the study is marked for downloading, that probably means that the query has not been retrieved, i.e. was aborted before completion. In the example below the query was run with the setting to not ignore duplicates, therefore the study was still downloaded but then thrown away by the import.

Troubleshooting: openrem_qr.log

Note that if a query does not work as expected the first location to check should be the *DICOM query summary* and the *Task management*. However if that does not clarify the issue looking at the logs will be a good idea.

If the default logging settings haven't been changed then there will be a log files to refer to. The default location is within your `logs` folder:

This file contains information about the query, the status of the remote node, the C-Find response, the analysis of the response, and the individual C-Move requests.

The following is an example of the start of the log for the following query which is run once an hour (ie some responses will already have been imported):

| OpenREM | CT | Mammography | Radiography | | Imports ▾ | Exports | ⚙ Config ▾ |

The first table below shows all the series that the remote PACS/DICOM node sent and where considered for downloading (moving). Click on any to see which files (usually images or structured reports) where considered for downloading.

If the study was already ignored (for example because of a filter), maybe the series got never asked for and hence are not listed here even if present on the remote.

The second table displays all import tasks that were run for this study. At the moment displaying Toshiba and weight/size imports is not supported.

You are viewing series of the study study 1.2.840.113619.6.95.31.0.3.4.1.4400.13.8620675.

Series UID	Series description	Modality	Downloaded?
1.3.12.2.1107.5.1.4.11090.30000022022409220062600002116	dose report	SR	Downloaded
1.3.12.2.1107.5.1.4.11090.30000022022409484529300000025	pet dose report sr	SR	RDSR present, ignoring all non-RDSR SR
1.3.12.2.1107.5.1.4.11090.30000022022409254338300006536	pet nac	PT	RDSR present, all not SR series ignored
1.3.12.2.1107.5.1.4.11090.30000022022409254338300006535	pet ac	PT	RDSR present, all not SR series ignored
1.3.12.2.1107.5.1.4.11090.30000022022409234631300010883	ac_ct_body	CT	RDSR present, all not SR series ignored
1.3.12.2.1107.5.1.4.11090.30000022022409220062600001748	topogram 0.6 tr60	CT	RDSR present, all not SR series ignored

Task UUID	Task type	Error	Status
a6476929-f2b6-4b9c-b2c2-f55072820417	import_rdsr	Already in db	Error

Abb. 7: Figure 5: The query study details page

```
openrem_qr.py 2 1 -dx -f 2016-05-04 -t 2016-05-04 -e "imported"
```

```
[04/May/2016 11:30:02] INFO [remapp.netdicom.qrscu:580] qrscu script called
[04/May/2016 11:30:02] INFO [remapp.netdicom.qrscu:595] Modalities are ['DX']
[04/May/2016 11:30:02] INFO [remapp.netdicom.qrscu:601] Date from: 2016-05-04
[04/May/2016 11:30:02] INFO [remapp.netdicom.qrscu:604] Date until: 2016-05-04
[04/May/2016 11:30:02] INFO [remapp.netdicom.qrscu:610] Study description exclude␣
→terms are ['imported']
[04/May/2016 11:30:03] INFO [remapp.netdicom.qrscu:267] Request association with␣
→Hospital PACS PACSAET01 (PACSEAT01 104 DICOM_QR_SCP)
[04/May/2016 11:30:03] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:03] INFO [remapp.netdicom.qrscu:277] assoc is ...
→<Association(Thread-7208, started daemon 140538998306560)>
[04/May/2016 11:30:03] INFO [remapp.netdicom.qrscu:280] DICOM Echo ...
[04/May/2016 11:30:03] INFO [remapp.netdicom.qrscu:282] done with status Success
[04/May/2016 11:30:03] INFO [remapp.netdicom.qrscu:284] DICOM FindSCU ...
[04/May/2016 11:30:03] INFO [remapp.netdicom.qrscu:311] Currently querying for DX␣
→studies...
[04/May/2016 11:30:03] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:04] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:04] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:04] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:05] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:05] INFO [remapp.netdicom.qrscu:311] Currently querying for CR␣
→studies...
[04/May/2016 11:30:05] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:05] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:06] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:06] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:06] INFO [remapp.netdicom.qrscu:33] Association response received
```

(Fortsetzung auf der nächsten Seite)

```
[04/May/2016 11:30:07] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:10] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:10] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:11] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:11] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:12] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:12] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:12] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:12] INFO [remapp.netdicom.qrscu:339] Checking to see if any of␣
↪the 16 studies are already in the OpenREM database
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:343] Now have 11 studies
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:349] Deleting studies we didn't␣
↪ask for
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:358] mod is DX, mod_set is ["CR"]
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:358] mod is CR, mod_set is ["CR"]
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:358] mod is DX, mod_set is ["PR",
↪ "DX"]
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:358] mod is CR, mod_set is ["PR",
↪ "DX"]
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:358] mod is DX, mod_set is ["DX"]
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:358] mod is CR, mod_set is ["DX"]
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:358] mod is DX, mod_set is ["PR",
↪ "CR"]
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:358] mod is CR, mod_set is ["PR",
↪ "CR"]
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:367] Now have 11 studies
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:372] Deleting series we can't use
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:408] Now have 11 studies
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:413] Deleting any studies that␣
↪match the exclude criteria
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:422] Now have 6 studies after␣
↪deleting any containing any of [u'imported']
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:438] Release association
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:499] Preparing to start move␣
↪request
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:504] Requesting move of 6 studies
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:509] Mv: study_no 1
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:515] Mv: study no 1 series no 1
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:528] Requesting move: modality␣
↪DX, study 1 (of 6) series 1 (of 1). Series contains 1 objects
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:44] Move association requested
[04/May/2016 11:30:18] INFO [remapp.netdicom.qrscu:53] Move association released
[04/May/2016 11:30:18] INFO [remapp.netdicom.qrscu:532] _move_req launched
[04/May/2016 11:30:18] INFO [remapp.netdicom.qrscu:509] Mv: study_no 2
[04/May/2016 11:30:18] INFO [remapp.netdicom.qrscu:515] Mv: study no 2 series no 1
[04/May/2016 11:30:18] INFO [remapp.netdicom.qrscu:528] Requesting move: modality␣
↪DX, study 2 (of 6) series 1 (of 1). Series contains 2 objects
[04/May/2016 11:30:18] INFO [remapp.netdicom.qrscu:33] Association response received
[04/May/2016 11:30:19] INFO [remapp.netdicom.qrscu:44] Move association requested
[04/May/2016 11:30:29] INFO [remapp.netdicom.qrscu:48] gg is Pending
[04/May/2016 11:30:30] INFO [remapp.netdicom.qrscu:53] Move association released
```

```
...etc
```
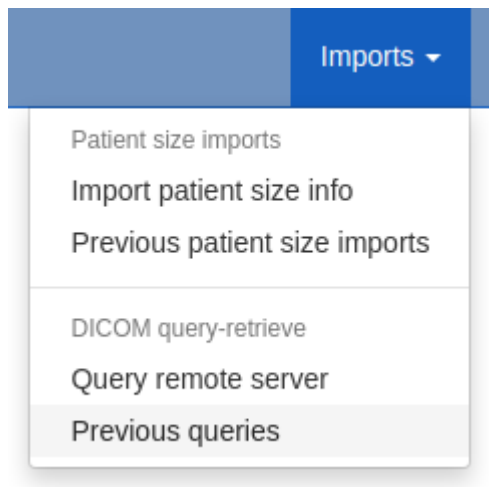

Abb. 5: Figure 3: Go to query summary

Navigating, filtering and study details

5.1 Navigating the OpenREM web interface

Depending on your web server setup, your web interface to OpenREM will usually be at http://yourserver/openrem or if you are using the test web server then it might be at http://localhost:8000/openrem.

The home page for OpenREM should look something like this when it is populated with studies:

By selecting the links in the navigation bar at the top, you can view all of the CT, fluoroscopy, mammography, radiographic or nuclear medicine studies. Alternatively, click on any row to filter by that system.

The modality tables can be sorted by any of the columns by clicking on the column header that you wish to sort by.

If you are not logged in, clicking any of the links will bring up the log in page.

5.2 Filtering for specific studies

This image shows the CT studies view, available to any logged in user, filtered by entering terms in the boxes on the right hand side to show just the studies where the modality manufacturer name includes the term ‚Siemens‘:

The search fields can all be used on their own or together, and they are all case insensitive ‚contains‘ searches. The exception is the date field, where both from and to have to be filled in (if either are), and the format must be yyyy-mm-dd. There currently isn't any more complex filtering available, but it does exist as issue 17 for a future release.

The last box below the filtering search boxes is the ordering preference.

OpenREM database browser and export

There are 30 studies in this database. Page last refreshed on 12. April 2022 14:06 Chart plotting off.

CT	Fluoroscopy	Mammography	Radiography	Nuclear Medicine
11	10	2	4	2

CT summary table

System name	Total number of studies	Latest study
REMOVED REMOVED	1	1 Monat, 2 Wochen her
NM SPECT Clinic SD_CT	1	1 Jahr, 6 Monate her
OpenREM Cancer Centre HOST-ABCD1234	1	2 Jahre, 10 Monate her
OpenREM CONTINUED	2	3 Jahre, 11 Monate her
OpenREM Clinic CTAWP12345	1	4 Jahre, 3 Monate her
OpenREM View Hospital AquilionOne	1	4 Jahre, 3 Monate her
OpenREM Clinic ToshibaDoseChk	1	4 Jahre, 4 Monate her
Oxbridge County Hospital CTTOSHIBA1	1	5 Jahre, 4 Monate her
Gnats Bottom Hospital CTAWP91919	1	8 Jahre, 10 Monate her
OpenREM centre médical rt16	1	9 Jahre, 1 Monat her

Fluoroscopy summary table

System name	Total number of studies	Latest study
The Royal Marsden	1	1 Jahr, 2 Monate her
Blank	2	2 Jahre, 3 Monate her
OpenREM GE Surgery GESURGIFPD	1	3 Jahre her
OpenREM Clinic GEMiniView	1	3 Jahre, 1 Monat her

Abb. 1: OpenREM homepage screenshot

Abb. 2: Filtering CT studies

5.2.1 CT: specifying number of event types

It is possible to filter for studies that have specific numbers of each acquisition type, or to only include studies have at least some (>0), or to include only studies that have zero acquisitions of a specific type.

For example, if the standard `CT Abdomen` on a particular scanner has two localisers and one spiral scan, then to filter for all the studies that followed this without deviation (an extra localiser or an extra series) the filters might be set to the particular Display Name and Requested Procedure, and Num. spiral events set to one and Num. localisers set to two. This can be useful for exporting a clean set of data to process for a dose audit.

5.3 Setting the number of studies displayed per page

The number of studies displayed per page can be controlled by changing the value selected in the `Items per page` drop down box, located beneath the chart options:

Abb. 3: Setting the number of studies per page

5.4 Viewing study details

By clicking on the study description link (in blue), you can see more details for an individual study:

Detail list of events

- Accession number: ab462362354
- Study date: 23 Jan 2013
- Study time: 1:17 p.m.
- Study description: Dual Energy^DE_TAP_IV (Adult)
- Requested procedure: CT Thorax abdomen and pelvis with contrast
- Patient age: 52.8
- Patient height and weight: 190 cm, 86 kg
- Hospital: Clinic B
- Scanner: SIEMENS | SOMATOM Definition Flash | CTAWP73491
- Study UID: 1.2.840.113564.9.1.27282345238.69.2.508347462734
- Comment:
- Test patient indicators? None

Acquisition protocol	Type	CTDIvol mGy	DLP mGy.cm	Scanning length (mm)	kVp	mA	Max mA	Exposure time per rotation (s)	Pitch	Exposure time (s)	Slice thickness (mm)	Collimation (mm)	X-ray modulation type
Topogram	Constant Angle Acquisition	0.14	11.26	803	120	35	35		None	8.190	0.600	3.60	OFF
Comment Internal technical scan parameters: Organ Characteristic = Thorax, Body Size = Adult, Body Region = Body, X-ray Modulation Type = OFF													
Topogram	Constant Angle Acquisition	0.14	11.54	824	120	35	35		None	8.400	0.600	3.60	OFF
Comment Internal technical scan parameters: Organ Characteristic = Thorax, Body Size = Adult, Body Region = Body, X-ray Modulation Type = OFF													
PreMonitoring	Stationary Acquisition	1.82	1.82	10	120	59	60	0.500	None	0.500	10.000	10.00	OFF
Comment Internal technical scan parameters: Organ Characteristic = Abdomen, Body Size = Adult, Body Region = Body, X-ray Modulation Type = OFF													
Monitoring	Stationary Acquisition	7.27	7.27	10	120	59	60	0.500	None	2.000	10.000	10.00	OFF
Comment Internal technical scan parameters: Organ Characteristic = Abdomen, Body Size = Adult, Body Region = Body, X-ray Modulation Type = OFF													
DE_TAP	Spiral Acquisition	8.11	630.63	797	100	165	430	0.500	0.8000	26.050	0.600	19.20	XYZ_EC
					140	131	307	0.500					
Comment Internal technical scan parameters: Organ Characteristic = Abdomen, Body Size = Adult, Body Region = Body, X-ray Modulation Type = XYZ_EC, Sn Filter (Tube B) = yes													

Abb. 4: Individual CT study

Not all the details stored for any one study are displayed, just those thought to be most useful. If there are others you'd like to see, add an issue to the tracker.

The final field in the summary at the top is called ‚Test patient indicators¿ When studies are imported the ID and patient name fields are both ignored, but they are parsed to check if they have ‚phy', ‚test' or ‚qa' in them to help exclude them from the data analysis. If they do, then this information is added to the field and is displayed both in the web interface as a Test patient indicator and in the Excel export. The name and ID themselves are not reproduced, simply the presence of one of the key words. Therefore a patient named ‚Phyliss' would trigger this, but only ‚Phy' would be reproduced in this field. Other fields will also help to confirm whether a study is for a real patient such as the lack of an Accession Number and an unusual patient age.

Bemerkung: For fluoroscopy the table showing details of each exposure can be sorted by clicking on the table headings.

5.4.1 A note on time data for fluoroscopy studies

On the page showing a specific fluoroscopy study there is a table that shows the details of each irradiation event in the study. This table includes a column labelled as:

```
Duration (ms)
Exposure time (ms)
```

The `Duration` value is the amount of time that the exposure switch or pedal was pressed (technically, this should be the time from the loading of the first x-ray pulse to the time of the trailing edge of the final pulse for that irradiation event). The `Exposure time` value is different: this is the total time that the x-ray beam was actually switched on for during the irradiation event. So for pulsed fluoroscopy the `Exposure time` will be (much) shorter than the `Duration`.

Near the top of each fluoroscopy study in the detail view is a table summarising the DAP, dose at reference point and duration for each irradiation type used in the study. Totals are also shown. The `Total duration` values in this table show the amount of time that the exposure switch or pedal was pressed.

KAPITEL 6

Charts

From OpenREM version 1.0.0+ charts use the Plotly open source Python library.

6.1 Chart types

The charts below are examples of the types of chart included in OpenREM version 1.0.0+. The examples are fully interactive in the same way as the charts included in a live OpenREM system. The data contained in the example charts is synthetic.

Single-clicking on a legend entry toggles the display of that series. Double-clicking on a legend entry hides all but that series; double-click again to show all series.

Hovering the cursor over a chart causes the chart menu to appear along the top right corner. From the menu you can:

- save a bitmap version of the chart

- set zoom, pan and selection options

- autoscale the chart

- reset the axes, and also reset the regions

- toggle spike lines to graphically illustrate x- and y-axis data values on hover

- choose whether to show the closest data when hovering, or to compare data on hover

6.1.1 Bar chart of average values across categories

These can be configured to show mean or median data. The example below shows the median DAP for each requested procedure name containing the word „knee" across eight x-ray rooms.

When viewing a chart of this type in OpenREM, the chart data can be downloaded as a csv file by clicking the *Download csv* button displayed below the chart. This feature is not available for the example charts in this document.

Hovering the cursor over a bar shows the:

- x-ray room name

- requested procedure name

- median DAP for that room and procedure

- number of requests for that room and procedure

6.1.2 Boxplot of values across a number of categories

The example below shows the same data as for the bar chart above, but presented as a box plot.

Hovering the cursor over an outlier shows the:

- x-ray room name

- requested procedure name

- DAP of the data point

Hovering the cursor over the box shows the:

- maximum value

- minimum value

- median

- 1st and 3rd quartiles

- lower fence: 1rd quartile - (1.5 x interquartile range)

- upper fence: 3rd quartile + (1.5 x interquartile range)

6.1.3 Histogram bar chart of values across categories

The example below shows the distribution of DAP values for the knee data used in the box and bar plots above. The number of bins used in the histograms can be configured in the *Additional chart options on the Config page*.

Hovering the cursor over a bar shows the:

- requested procedure name

- x-ray room name

- bin DAP range

- bin DAP mid-point value

- bin frequency

6.1.4 Bar chart of category frequency

The example below shows the frequency of the „knee" requested procedures for each x-ray room. The height of each bar is the total frequency for that requested procedure. Each bar is sub-divided into sections representing the number of requests for each x-ray room.

When viewing a chart of this type in OpenREM, the chart data can be downloaded as a csv file by clicking the *Download csv* button displayed below the chart. This feature is not available for the example charts in this document.

Hovering the cursor over a bar section shows the:

- x-ray room name

- requested procedure name

- requested procedure frequency

Setting *Grouping choice* to **System names** in the *Chart options on the modality pages* groups the data by x-ray system name rather than requested procedure name, as shown below:

6.1.5 Scatter chart of x vs y values

The example below shows the average glandular dose plotted against compressed breast thickness for „MAMMO-GRAM" acquisitions made on two x-ray systems.

Hovering the cursor over a bar section shows the:

- x-ray room name

- acquisition protocol name

- compressed breast thickness

- average glandular dose

6.1.6 Bar chart of workload

These show the number of studies taking place per weekday, sub-divided into hours of the day.

There is a bar per weekday. The total height of this bar is the number of studies carried out on that weekday. Each bar is sub-divided into sections representing the number of studies carried out in each of the 24 hours of that day. Each section is colour-coded according to how many studies it represents.

Hovering the cursor over a section shows you the:

- x-ray room name

- day of the week that the section represents

- hour of the day that the section represents

- number of studies that took place in that hour on that weekday in that x-ray room

6.1.7 Line chart of average value over time

These can be configured to show mean or median data. Each datapoint represents the average over a user-specified time period. This can be a day, week, month, quarter or year.

The example below shows the median DAP for „Head" requests made in four CT scanners over the course of five years.

With *Grouping choice* set to **Series item names** in the *Chart options on the modality pages* a sub-plot is created for each requested procedure name, each with a series per x-ray system as shown below. The *Number of charts per row* in the *Additional chart options on the Config page* was set to 2 for these example charts.

Hovering the cursor over a section shows you the:

- scanner name

- requested procedure name

- date

- average DLP value

- number of requests included in the sample

Setting *Grouping choice* to **System names** in the *Chart options on the modality pages* changes the grouping. Now a sub-plot is created for each x-ray system, each with a series per requested procedure name, as shown below:

6.1.8 Bar chart of average value against another binned value across categories

These can be configured to show mean or median data. The example below shows the median average glandular dose from „MAMMOGRAM" protocol exposures plotted against compressed breast thickness bins. The data is from two x-ray systems.

Hovering the cursor over a section shows you the:

- x-ray room name

- acquisition protocol

- average AGD value

- number of acquisitions included in the sample

- compressed breast thickness bin range
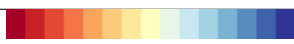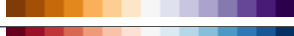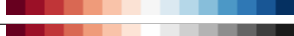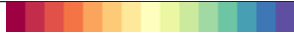
6.2 Chart options on the modality pages

Name	Configuration options	Notes
Average plots	Any combination of **mean**, **median** or **boxplot**	
Time period	One of **day**, **week**, **month**, **quarter** or **year**	Applies to over-time charts
Grouping choice	**System names** or **Series item names**	**System names** groups by x-ray system **Series item names** groups by each category
Plot a series per system	**On** or **off**	Splits the data by x-ray system
Calculate histogram data	**On** or **off**	Calculate histograms for average bar charts
Chart sorting	One of **name**, **frequency** or **value**	Sort the chart data according to the selected choice
Sorting direction	**Ascending** or **descending**	Sets the sort direction
Split plots by physician	**On** or **off**	Calculate a series per physician (*some fluoroscopy charts only*)

6.3 Additional chart options on the Config page

Name	Configuration options	Notes
Number of histogram bins	Value in the range 2 - 40	Default is 10
Fixed histogram bins across sub-plots	**On** or **off**	Forces all histogram sub-plots to use the same bins
Case-insensitive categories	**On** or **off**	Category names forced to lowercase For example, „Chest PA" becomes „chest pa"
Remove category whitespace padding	**On** or **off**	Removes spaces at beginning and end of category names, replaces multiple spaces with single spaces For example, " Chest PA " becomes „Chest PA"
Colour map choice	One of the available matplotlib colour maps	See the *available colourmaps* section
Chart theme	One of **Plotly**, **Plotly white**, **Plotly dark**, **presentation**, **ggplot2**, **Seaborn** or **simple white**	Set the Plotly theme to use for the charts. Some *example themed charts* are provided below. Examples of all themes on the Plotly themes page (external link).
Number of charts per row	Value in the range 1 - 10	Sets the number of sub-plots in each row

6.3.1 Available colourmaps

Name	Swatch
Red yellow blue	
Spectral	
Pink yellow green	
Purple green	
Brown green	
Purple orange	
Red blue	
Red grey	
Yellow green blue	
Yellow orange brown	
Hot	
Inferno	
Magma	
Plasma	
Viridis	
Cividis	

6.3.2 Some available themes

The example *Chart types* at the top of this document use the default Plotly theme. Below are some examples of other available themes.

Plotly dark

Presentation

Simple white

6.4 Available CT charts

Chart name	Chart type
Acquisition frequency	Bar chart of acquisition protocol frequency
Acquisition DLP	Bar chart of average DLP per acquisition protocol Boxplot with data point per acquisition protocol Histograms also plotted if *Calculate histogram data* **on**
Acquisition CTDI$_{vol}$	Bar chart of average CTDI$_{vol}$ per acquisition protocol Boxplot with data point per acquisition protocol Histograms also plotted if *Calculate histogram data* **on**
Acquisition DLP over time	Line chart of average DLP over time for each acquisition protocol
Acquisition CTDI$_{vol}$ over time	Line chart of average CTDI$_{vol}$ over time for each acquisition protocol
Acquisition DLP vs mass	Scatter chart of DLP vs patient mass for each acquisition protocol
Acquisition CTDI$_{vol}$ vs mass	Scatter chart of CTDI$_{vol}$ vs patient mass for each acquisition protocol
Study frequency	Bar chart of study description frequency
Study DLP	Bar chart of average DLP per study description Boxplot with data point per study description Histograms also plotted if *Calculate histogram data* **on**
Study CTDI$_{vol}$	Bar chart of average CTDI$_{vol}$ per study description Boxplot with data point per study description Histograms also plotted if *Calculate histogram data* **on**
Study events	Bar chart of average number of radiation events per study description Boxplot with data point per study description Histograms also plotted if *Calculate histogram data* **on**
Study DLP over time	Line chart of average DLP over time for each study description
Study workload	Bar chart of number of studies carried out on each day of the week, with each bar sub-divided into hours of the day
Requested procedure frequency	Bar chart of requested procedure name frequency
Requested procedure DLP	Bar chart of average DLP per requested procedure name Boxplot with data point per study description Histograms also plotted if *Calculate histogram data* **on**
Requested procedure events	Bar chart of average number of radiation events per requested procedure name Boxplot with data point per study description Histograms also plotted if *Calculate histogram data* **on**
Requested procedure DLP over time	Line chart of average DLP over time for each study description

6.5 Available radiographic charts

Chart name	Chart type
Acquisition frequency	Bar chart of acquisition protocol frequency
Acquisition DAP	Bar chart of average DAP per acquisition protocol Boxplot with data point per acquisition protocol Histograms also plotted if *Calculate histogram data* **on**
Acquisition mAs	Bar chart of average mAs per acquisition protocol Boxplot with data point per acquisition protocol Histograms also plotted if *Calculate histogram data* **on**
Acquisition kVp	Bar chart of average kVp per acquisition protocol Boxplot with data point per acquisition protocol Histograms also plotted if *Calculate histogram data* **on**
Acquisition DAP over time	Line chart of average DAP over time for each acquisition protocol
Acquisition mAs over time	Line chart of average mAs over time for each acquisition protocol
Acquisition kVp over time	Line chart of average kVp over time for each acquisition protocol
Acquisition DAP vs mass	Scatter chart of DAP vs patient mass for each acquisition protocol
Study frequency	Bar chart of study description frequency
Study DAP	Bar chart of average DAP per study description Boxplot with data point per study description Histograms also plotted if *Calculate histogram data* **on**
Study DAP vs mass	Scatter chart of DAP vs patient mass for each study description
Study workload	Bar chart of number of studies carried out on each day of the week, with each bar sub-divided into hours of the day
Requested procedure frequency	Bar chart of requested procedure name frequency
Requested procedure DAP	Bar chart of average DAP per requested procedure name Boxplot with data point per study description Histograms also plotted if *Calculate histogram data* **on**
Requested procedure DAP vs mass	Scatter chart of DAP vs patient mass for each requested procedure name

6.6 Available fluoroscopy charts

Chart name	Chart type
Study frequency	Bar chart of study description frequency
Study DAP	Bar chart of average DAP per study description Boxplot with data point per study description Histograms also plotted if *Calculate histogram data* **on**
Study DAP over time	Line chart of average DAP over time for each study description
Study workload	Bar chart of number of studies carried out on each day of the week, with each bar sub-divided into hours of the day
Requested procedure frequency	Bar chart of requested procedure name frequency
Requested procedure DAP	Bar chart of average DAP per requested procedure name Boxplot with data point per study description Histograms also plotted if *Calculate histogram data* **on**
Requested procedure DAP over time	Line chart of average DAP over time for each study description

6.7 Available mammography charts

Chart name	Chart type
Acquisition frequency	Bar chart of acquisition protocol frequency
Acquisition AGD	Bar chart of average AGDP per acquisition protocol Boxplot with data point per acquisition protocol Histograms also plotted if *Calculate histogram data* **on**
Acquisition average AGD vs thickness	Bar chart of average AGD for each of the following nine compressed breast thickness bands: min x < 20; 20 x < 30; 30 x < 40; 40 x < 50; 50 x < 60; 60 x < 70; 70 x < 80; 80 x < 90; 90 x < max
Acquisition AGD over time	Line chart of average AGD over time for each acquisition protocol
Acquisition AGD vs thickness	Scatter chart of AGD vs compressed breast thickness for each acquisition protocol
Acquisition mAs vs thickness	Scatter chart of mAs vs compressed breast thickness for each acquisition protocol
Acquisition kVp vs thickness	Scatter chart of kVp vs compressed breast thickness for each acquisition protocol
Study workload	Bar chart of number of studies carried out on each day of the week, with each bar sub-divided into hours of the day

6.8 Available nuclear medicine charts

Chart name	Chart type
Study frequency	Bar chart of study description frequency
Study description workload	Bar chart of number of studies carried out on each day of the week, with each bar sub-divided into hours of the day
Injected dose per study	Bar chart or boxplot of average injected dose per study description. If calculate histogram is enabled creates a histogram of injected dose
Injected dose over time	Bar chart or boxplot of injected dose per study description over time
Injected dose over weight	Show a scatter plot of injected dose versus patient weight

Standard name mapping

7.1 Introduction

The same type of examination can appear in OpenREM under a range of different `requested procedure names`, `study descriptions` or `procedure names`. Individual acquistions of the same type may appear under a range of different `acquisition protocol names`. The standard name mapping feature allows a number of differently named exams or acquisitions to be grouped into a single standard name, allowing easier analysis of data for audits and diagnostic reference level (DRL) creation (see DRL definition on the IAEA website).

If the logged in user is an OpenREM adminstrator the `standard name settings` option will be available on the `Config` drop down menu (figure 1). Clicking on this will take the administrator to the standard name mapping settings page (figure 2). From this page the administrator can enable or disable standard name mapping for the OpenREM installation. Enabling standard name mapping causes an additional column to be displayed in each modality summary page where any standard study-level names associated with each entry in the exam table are displayed (figure 3), and when viewing the details of an individual study any matching standard acquisition names are displayed in the appropriate table row (figure 4). Enabling standard name mapping also makes the standard study name and standard acquisition name charts available. The administrator can also use the `standard name settings` page to access the standard name internal database refresh links. Note that the use of these links is not expected to be necessary in normal OpenREM use.

Selecting the `standard name mapping` option within the `Config` drop down menu (figure 1) will take the user to the standard name mapping summary page(figure 5). This page shows any existing standard name mappings, which an OpenREM administrator is able to `Modify` or `Delete` using the buttons on the right hand side of each entry. A normal OpenREM user can only view standard name mappings.
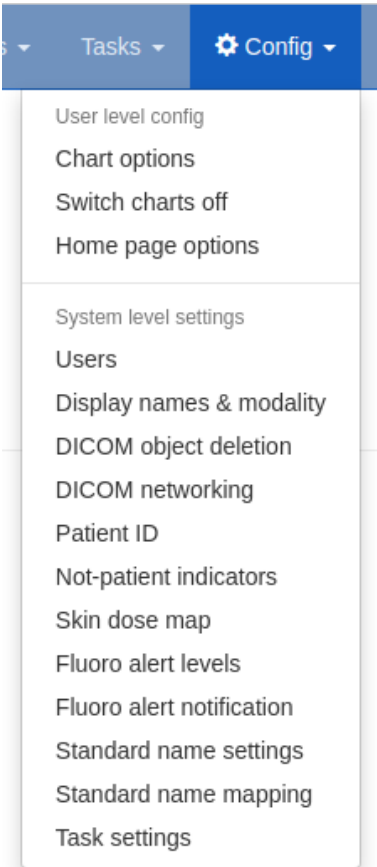
Abb. 1: Figure 1: The config menu
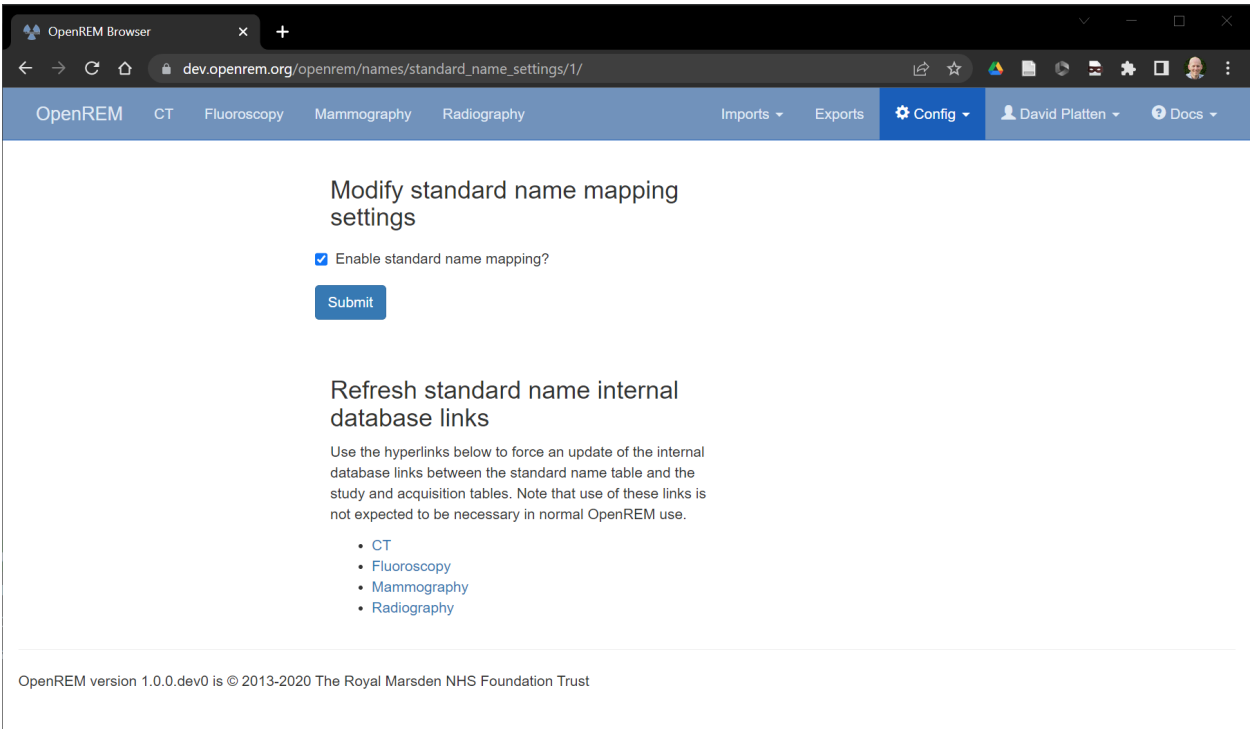
Abb. 2: Figure 2: The standard name mapping settings page
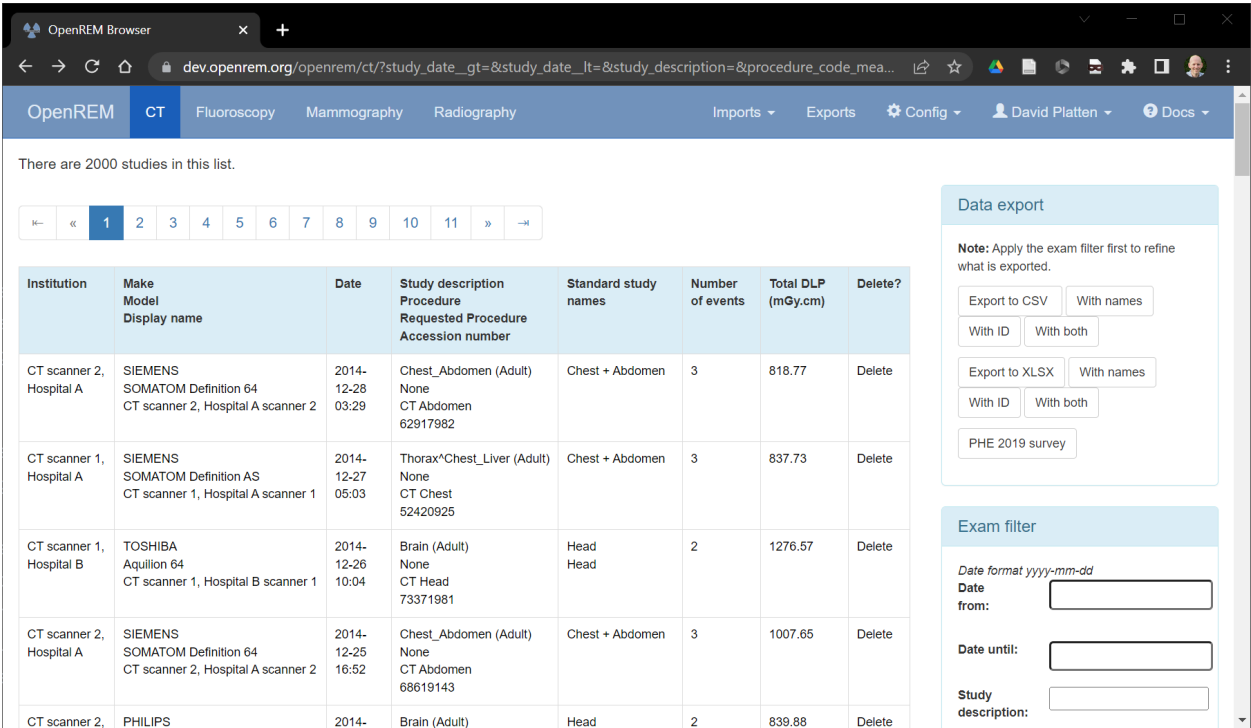
Abb. 3: Figure 3: Standard study names visible in the CT summary page

Abb. 4: Figure 4: Standard acquisition names visible in a study details page

Abb. 5: Figure 5: The standard name mapping summary page

7.2 Creating a new standard name mapping

A new standard name mapping can be created by an OpenREM administrator by clicking on the `Add new XX entry` button, where `XX` corresponds to a particular modality. This takes the administrator to a screen where the new standard name is set (figure 6), and where the administrator selects the `study descriptions`, `requested procedure names`, `procedure names` and `acquisition protocol names` that they want to be included in the new standard name definition. The available items are listed in the left-hand tables. The administrator can move a required item into the right-hand table by double-clicking on an entry, or selecting an entry and then clicking on the arrow pointing to the right.

The example in figure 7 shows that head-related `study descriptions` and `requested procedure names` have been chosen for a new standard name of `Head`.

Once all relevant items have been transferred to the right-hand tables the `Submit` button at the bottom of the page must be clicked to confirm the new entry. Once a `study description`, `requested procedure name`, `procedure name` or `acquisition protocol name` has been assigned to a standard name it cannot be added to another standard name, and disappears as an option in the left-hand tables when configuring future new standard name entries.

7.3 Modifying an existing standard name mapping

An existing standard name mapping can be modified by clicking the `Modify` button on the right-hand side of an entry in the standard name mapping summary page (figure 5). This takes the administrator to the same screen as shown in figure 7, where the chosen `study descriptions`, `requested procedure names`, `procedure names` and `acquisition protocol names` can be amended. Clicking `Submit` confirms the changes.

7.4 Charts

Charts of standard name data can be plotted in OpenREM. This can be helpful because at study-level it enables multiple `study descriptions`, `requested procedure names` and `procedure names` to be combined into a single data point. At acquisition level, multiple `acquisition protocol names` can be combined into a single data point. For example, figure 8 below shows the median DLP for a range of `study descriptions`. Three of the `study descriptions` relate to the head, and two of them relate to scans of the chest and abdomen. The three head-related descriptions have been mapped to a `Head` standard study name, and the two chest and abdomen descriptions have been mapped to a `Chest + Abdomen` standard study name, resulting in the chart shown in figure 9. The standard name mapping allows clearer visual comparison of the data per study for each hospital and piece of equipment.

Abb. 6: Figure 6: Adding or modifying a standard name

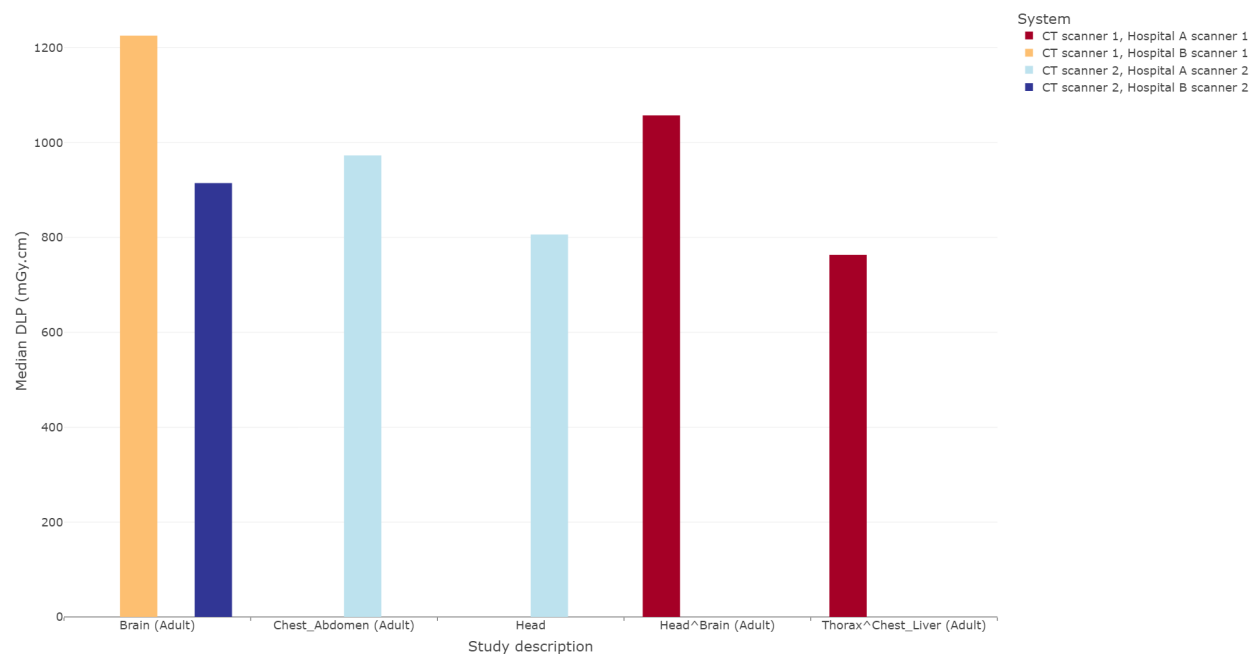Abb. 7: Figure 7: Adding or modifying a standard name mapping

Abb. 8: Figure 8: Chart of median DLP for each study description

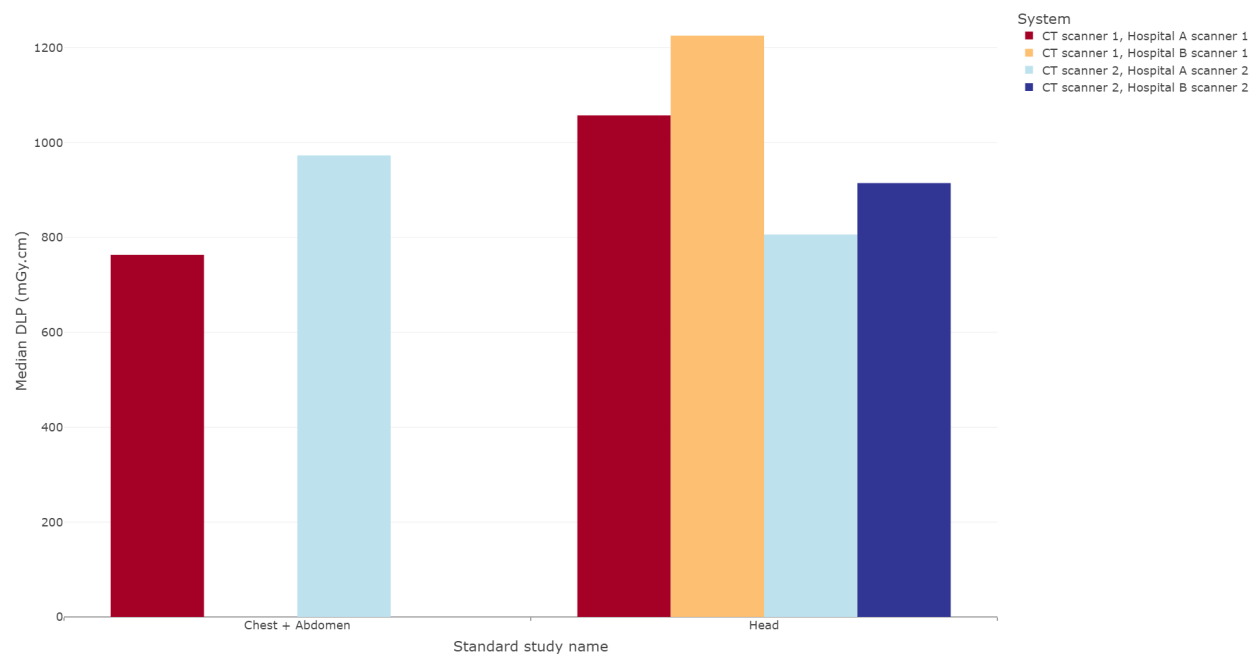

Abb. 9: Figure 9: Chart of median DLP for each standard study name

Skin dose maps

8.1 Functionality that is available

- Skin dose map data is calculated to the surface of a simple geometric phantom using the in-built openSkin routines (3D phantom)

- The calculated doses include kVp-dependent backscatter factors and account for any copper filters using data from this paper for each irradiation event; aluminium or other filters are not considered. Where more than one kVp is stored for an irradiation event the mean kVp is calculated, excluding any zero values.

- The phantom dimensions are calculated from the height and mass of the patient; defaults of 1.786 m and 73.2 kg are used when patient height and mass are not available

- Data can be calculated on import to OpenREM, or on demand when a study is viewed. Calculating the skin dose map can take several minutes.

- Data is recalculated automatically if the patient height or mass stored in the database differs from the values stored in the skin dose map data file. This is useful when patient size information has been imported in to OpenREM after the initial skin dose map data has been calculated

- 3D skin dose map data is shown graphically as a 2D image and a 3D model

- The 3D model can be manipulated using a touch screen

- The user can change the maximum and minimum displayed dose; alternatively, window level and width can be adjusted

- A colour dose scale is shown with a selection of colour schemes

- The skin dose map section can be displayed full-screen

- The calculated peak skin dose, phantom dimensions, patient height, mass and orientation used for the calculations are shown in the top left hand corner of the skin dose map

- Additionally the percentage of exposures that interact with the phantom and therefore contribute to the skin dose is shown as DAP percentage contributed, enhancing the interpretability of the skin dose calculation.

- If skin dose map display is disabled then fluoroscopy study data can be exported in a format suitable for the stand-alone openSkin routines

The phantom consists of a cuboid with one semi-cylinder on each side (see 3D phantom section of phantom design on the openSkin website for details).

8.1.1 2D visualisation of the 3D data

This is a 2D view of the whole surface of the 3D phantom, as though the phantom surface has been peeled off and laid out flat (figure 1). The 2D visualisation includes the following features:

- The skin dose at the mouse pointer is shown as a tool-tip

- Moving the mouse whilst holding down the left-hand mouse button changes the window level and width of the displayed skin dose map

- An overlay indicating the phantom regions and orientation can be toggled on and off. This indicates the phantom anterior, left, posterior and right sides, and also shows the superior and inferior ends (figure 2)

- The current view can be saved as a png file

Abb. 1: Figure 1: 2D visualisation of the 3D data

8.1.2 3D visualisation

This is a 3D view of the phantom that was used for the calculations, with the skin dose map overlaid onto the surface. The 3D visualisation includes the following features:

- Moving the mouse whilst holding down the left-hand mouse button rotates the 3D model

- Using the mouse wheel zooms in and out
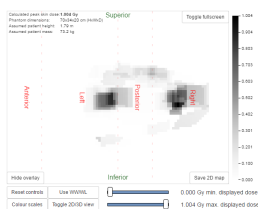
Abb. 2: Figure 2: Phantom region overlay

- A simple 3D model of a person is displayed in the bottom left corner. This is to enable the viewer to orientate themselves when viewing the 3D skin dose map

- The current view can be saved as a png file

8.2 Skin dose map settings

There are two skin dose map options that can be set by an OpenREM administrator via the `Skin dose map settings` option in the `Config menu`:

- Enable skin dose maps

- Calculate skin dose maps on import

- Ignore systems safelist

The first of these sets whether skin dose map data is calculated, and also switches the display of skin dose maps on or off. The second option controls whether the skin dose map data is calculated at the point when a new study is imported into OpenREM, or calculated when a user first views the details of that particular study in the OpenREM interface. The third option allows the user to enables skin dose maps for systems that are not validated for OpenSkin.

When skin dose maps are enabled:

- When a user views the details of a fluoroscopy study OpenREM looks for a skin dose map pickle file on the OpenREM server in the `skin_maps` subfolder of `MEDIA_ROOT` that corresponds to the study being viewed. If found, the skin dose map data in the pickle file is loaded and displayed. The `skin_maps` folder is created if it does not exist

- If a pickle file is not found then OpenREM calculates skin dose map data. These calculations can take some time. They are carried out in the background: an animated graphic is shown during the calculations. On successful calculation of the data the skin dose map is displayed. A pickle file containing the data is saved in the server's `skin_maps` subfolder of `MEDIA_ROOT`. The file name is of the form `skin_map_XXXX.p`, where `XXXX` is the database primary key of the study

- For subsequent views of the same study the data in the pickle file is loaded, rather than re-calculating the data, making the display of the skin dose map much quicker

When calculation on import is enabled:

- OpenREM calculates the skin dose map data for a fluoroscopy study as soon as it arrives in the system

- A pickle file containing the data is saved in the `skin_maps` subfolder of `MEDIA_ROOT`

- Users viewing the details of a study won't have to wait for the skin dose map data to be calculated

8.3 Exporting data to openSkin

If skin dose maps are disabled, and the user has export rights, the user is presented with the option of exporting the study data as a csv file that is formatted for use with a stand-alone installation of openSkin. The user must be in the detail view of the study they wish to create the exposure incidence map for, and then click on the link to create the OpenSkin export (figure 4).

8.4 Instructions for openSkin

Download the openSkin repository as a zip file from openSkin downloads. To use openSkin as a stand-alone application you need python 2.x and the pypng python library.

- Extract the contents of the zip file into a folder on your computer and run *python main.py* from a command line and answer each question.

- See phantom design for details of the 2D and 3D phantoms.

- When asked for the source csv file use the one exported from OpenREM

- Depending on the number of events in the export and the power of your computer the calculations can take a few minutes

Two files will be produced - a textfile called `skin_dose_results.txt` and a small image called `skin_dose_map.png`

8.4.1 Results text file

It should look something like this:

```
File created    : 04/21/15 17:42:45
Data file  ␣
→    : C:/Users/[...]/exports-2015-04-21-
→OpenSkinExport20150421-162805246134.csv
Phantom         : 90.0x70.0 3d phantom
Peak dose␣
→(Gy)  :                 0.50844405521
Cells >␣
→3 Gy     :                          0
Cells >␣
→5 Gy     :                          0
Cells >␣
→10 Gy    :                          0
```

The peak dose is the peak incident dose delivered to any one-cm-square area. If any of these 1 cm^2 areas (referred to as cells) are above 3 Gy, then the number of cells in this category, or the two higher dose categories, are listed in the table accordingly.

8.4.2 Incidence map image file

The image file will be a small 70x90 px PNG image if you used the 3D phantom, or 150 x 50 px PNG if you used the 2D phantom. With both, the head end of the table is on the left.

The image is scaled so that black is 0 Gy and white is 10 Gy. For most studies, this results in an incidence map that is largely black. However, if you use GIMP or ImageJ or similar to increase the contrast, you will find that the required map is there.

A native and 'colour equalised' version of the same export are shown below:
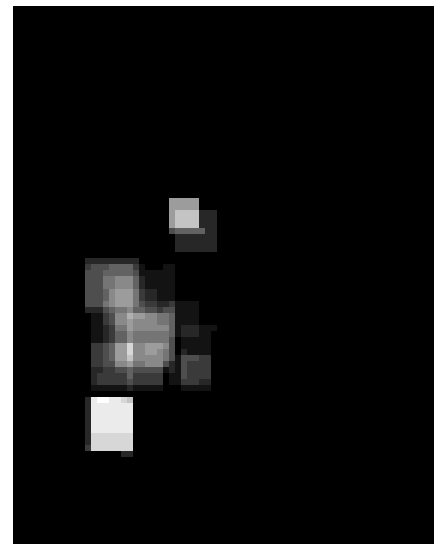
8.5 Limitations

Skin dose map calculations do not currently work for all systems. Siemens Artis Zee data is known to work. If skin dose maps do not work for your systems then please let us know via the OpenREM Google Group.

Monte Carlo validation has been carried out for a Siemens Artis zee system for the body section of the phantom: openSkin results were within ± 20% of the Monte Carlo results (*Platten 2023*). Skin dose values depend on the accuracy of the x-ray system's dose at reference point and DAP values. For other types of x-ray system the results are for indication only as they have not been validated.

If you would like to help with validation of openSkin please do go ahead and feed back your findings to Jonathan Cole at jacole and David Platten at dplatten.

8.6 References

Platten, D. „Monte Carlo simulation of skin dose distribution from interventional cardiology procedures", Doctor of Clinical Science thesis (2023), The University of Manchester, UK (external link to thesis).

Exporting study information

9.1 Exporting to csv and xlsx sheets

If you are logged in as a user in the `exportgroup` or the `admingroup`, the export links will be available near the top of the modality filter pages in the OpenREM interface.

For each modality you can export to a single-sheet csv file or a multi-sheet xlsx file. In addition, there is an export tailored to the *NHSBSP dose audits* requirements.

If you are logged in as a user in the `pidgroup` you will also have a choice of exporting with patient name and/or patient ID information included in the export (if any is recorded in the database). See *Patient identifiable data* for more details.

The xlsx export has multiple sheets. The first sheet contains a summary of all the study descriptions, requested procedures and series protocol names contained in the export:

This information is useful for seeing what data is in the spreadsheet, and can also be used to prioritise which studies or protocols to analyse based on frequency.

The second sheet of the exported file lists all the studies, with each study taking one line and each series in the study displayed in the columns to the right.

The remainder of the file has one sheet per series protocol name. Each series is listed one per line. If a single study has more than one series with the same protocol name, then the same study will appear on more than one line.

9.1.1 Fluoroscopy exports

Fluoroscopy csv exports only report study level information — this includes summary dose information for fluoroscopy exposures and acquisition exposures, but not information about individual exposures.

Fluoroscopy xlsx exports contain the following sheets:

- Summary sheet
- All data sheet with groups of exposures
- One sheet per acquisition protocol with one row per exposure including all the details of that exposure.

Exposures are considered similar and put in the same group if they are, relative to the first exposure of a group:

- same plane (for bi-plane systems)
- same protocol
- same field size (mag, not collimation)
- same pulse rate
- same filter material and thickness
- within 5° in both directions (primary and secondary)
- of the same ‚event type‘ (Fluoroscopy, Stationary Acquisition, Stepping Acquisition, Rotational Acquisition)

The minimum, maximum and mean of all the remaining factors are presented for each group along with the common factors. Where a factor is not available in the source RDSR, that factor is not considered.

The grouping process for the all data sheet takes a lot of time compared to the other exports. However, we hope that this is a useful way of comprehending the study. Other modalities have all the series for any one study detailed in full on one long row — this is not possible when one study might have 400 exposures!

The majority of systems report kV, mA and pulse width information as a mean value per exposure. Some systems report this information on a per pulse basis instead. In this circumstance, in the web interface you will see the list of pulses, but in the export the mean value (after excluding any zero values) is calculated first and this value is then used.

9.1.2 Nuclear medicine exports

Nuclear medicine does not have a concept like different irradiation events. Therefore in contrast to the other modalities it the summary dose information does not show protocols, and the excel export has individual sheets per study description instead of per protocol names.

9.1.3 Exports page

Clicking the link for an export redirects you to the Exports page, which you can also get to using the link at the top right of the navigation bar:

Whilst an export is being processed, it will be listed in the first table at the top. The current status is displayed to indicate export progress, and is updated every two seconds. You can stop an export early by using the abort button; you will not be able to download anything in this instance.

Once a study is complete a new table of recently completed exams is created and you will be able to download the file.

When the export is no longer needed, it can be deleted from the server by ticking the delete checkbox and clicking the delete button at the bottom:

9.2 Specific modality export information

9.2.1 NHSBSP dose audits

This export is specific to the UK NHS Breast Screening Programme and generates the source data in the format required for the dose audit database developed by the National Co-ordinating Centre for the Physics of Mammography.

It has been modified to clean up the data to remove exposures that are unlikely to be wanted in the submitted data, such as exposures with any of the following in the protocol name:

```
scout, postclip, prefire, biopsy, postfire, stereo, specimin, artefact
```

The view codes have been modified to match the NCCPM convention, i.e. medio-lateral oblique is recorded as `OB` instead of `MLO`. The other codes are mapped to the ACR MQCM 1999 Equivalent code.

Each patient is numbered from starting from 1. Each view for any one patient has a unique view code, so if a second cranio-caudal exposure is made to the left breast the view codes will be LCC and LCC2.

The survey number is left as 1. This needs to be modified as appropriate. The easiest way to do this in Excel is to change the first two or three rows, select those cells that have been changed, then double click on the bottom-right corner of the selection box to copy-down the value to all the remaining cells below.

The data can then be copied and pasted into the NCCPM database.

If there are a mixture of 2D and tomography exposures, providing you can separate them by virtue of the filter used, then you should further prepare the data as follows:

1. Copy the sheet to a new sheet

2. In the first sheet, filter for the target and filter combination used for used for the tomographic exposures and delete those rows.

3. In the second sheet, filter for the target and filter combinations used for 2D exposures and delete those rows.

4. Change the survey number on the 2D sheet and the the survey number on the tomographic sheet as appropriate, with the tomographic survey number bing one more than the 2D survey number.

Where patients have had both 2D and tomographic exposures in the same study, NCCPM will be able to match them up as they will have the same patient number in both surveys.

9.2.2 PHE 2019 CT survey and IPEM/PHE 2019 paediatric CT survey

This export is specific to the UK Public Health England (PHE) CT dose survey and exports the data in the correct format to copy and paste into the spreadsheet provided by PHE. More information about the survey and copies of the data collection spreadsheet can be found on the CT User Group (CTUG) website. The same export function is also suitable for the UK Paediatric CT Dose Survey launched by the IPEM paediatric optimisation working party in collaboration with PHE. The spreadsheet and instructions for this survey can also be found on the CTUG website.

The introduction and guidance tabs of the PHE data collection spreadsheet should be read and the ‚Your details‘ sheet completed. Then the ‚Patient and Protocol data 1‘ sheet should be copied and renamed appropriately for each protocol and scanner combination that you will be submitting. The first 142 rows of each sheet should be filled in manually with all the details for that protocol, though looking at study data in OpenREM may help to answer some of the questions.

The CT studies should then be filtered in OpenREM; by date (ideally previous 12 months, no older than 2017), by scanner (each scanner and protocol combination should be a new sheet), by age of (minimum of 16 for the PHE adult survey), and by study description (or combination of factors to specify a particular protocol).

Finally the studies should be filtered to have exactly the right number of each type of acquisition for that protocol. This might be one spiral, one localiser and two stationary (bolus tracking) acquisitions for example. Localisers do not appear in the export so are less important to specify, but more localisers than usual might indicate a deviation from the standard protocol.

The export can then be started and monitored in the normal way by clicking on the ‚PHE 2019 survey‘ button. The resulting export will be in xlsx format, with one header row. The data from row 2 onwards can be copied and pasted directly into row 150 (152 for paed survey) onwards of the Patient and Protocol sheet of the adult PHE data collection spreadsheet. The adult survey starts in column A, the paediatric survey starts in column B. Column AL (AM for paed survey) is for patient comments, and OpenREM uses this cell to record the series types that have been exported for each study. This can therefore be used to double check the data is as you expect it to be. If the protocol has more than four series excluding localisers, the data is continued in the same format from column AM (AN for paed) onwards.

9.2.3 PHE 2019 X-ray, fluoroscopy and interventional radiology survey

As with the PHE and IPEM CT surveys, the PHE exports on the radiography and fluoroscopy modality filter pages are designed to paste directly into templates provided by PHE via the Medical-Physics-Engineering JiscMail e-mail list or available directly from PHE.

The spreadsheets provided are:

Planar Radiography Survey:

- `PHE_Dose_PR_Patient_xxxxx.xlsx` Individual patient records - OpenREM export is designed for this

- `PHE_Dose_PR_Patient_X26_xxxxx.xlsx` Individual patient records, specifically for skeletal surveys

- `PHE_Dose_PR_System_xxxxx.xlsx` System mean and median - OpenREM does not support this summary format directly

IR and Fluoroscopy Survey:

- `PHE_Dose_IR_Fluoro_Patient_xxxxx.xlsx` Individual patient records - OpenREM export is designed for this

- `PHE_Dose_IR_Fluoro_System_xxxxx.xlsx` System mean and median - OpenREM does not support this summary format directly

As with the CT surveys, care should be taken to read and fill in the instructions and questions asked in the templates.

For the **radiography survey**, there are two types of studies asked for:

1. Single-projection studies - for example the sheet `Abdomen_AP_DAP_by_record`

2. Multi-projection studies - for example the sheet `Abdomen_exam_DAP_by_record`

For studies in the former category, select your data as appropriate on the Radiography filter page, making use of the ‚Num. events total' filter to ensure all selected studies have just one exposure. Then use the ‚PHE 2019 Survey: Projection' button to export the data. If any exams do have more than one exposure, the export will continue with a warning, and only include the first exposure of each study in the export.

For multi-projection studies, filter as before, setting the ‚Num. events total' filter if the exam normally has a specific number of views, and export using the ‚PHE 2019 Survey: Study' button. If any of the studies have more than six exposures, the export will automatically format the data to suit the template designed for skeletal studies. This allows for up to 20 exposures. However, you will need to request the ‚bespoke' template from PHE as it was not distributed in the original email!

The exported spreadsheet has a header row at the top. Copy from the second row onward and paste into the relevant sheet in the PHE template in row 7.

For The **fluoroscopy survey**, select the studies as appropriate and use the ‚PHE 2019 Survey' button to export the data. The resulting data should be copied into the `PHE_Dose_IR_Fluoro_Patient_xxxxx.xlsx` as follows:

1. Select the exported data from column `A` through to column `AQ`, row 1 though to the last row of the exported data

2. Select the cell `A1` and choose the full Paste Special menu - `Ctrl+Alt+V` or right-click Paste Special -> Paste Special. Then select Paste ‚Formulas' and ‚Skip Blanks'.

This will paste the correct DAP and time units into row 4 along with the exported data into row 7 onwards. The DAP units exported using the radiography exports (cGy·cm^2) and fluoroscopy (Gy·m^2) may not correspond to the units normally used on your system, but the exported values will be correct for the units stated.

9.3 Opening csv exports in Excel

If the export contains non-ASCII characters, then Microsoft Excel is unlikely to display them correctly by default. This issue does not occur with Libre Office which defaults to UTF-8 – behaviour with other applications will vary.

To correctly render characters in csv files with Excel, you will need to follow the following procedure:

1. Open Excel.

2. On the `Data` tab of the ribbon interface, select `From Text` in the `Get External Data` section.

3. Select your exported csv file and click `Import`

4. Ensure that Data Type `Delimited` is selected.

5. Change the `File origin` from to `65001 : Unicode (UTF-8)` – the easiest way to find it is to scroll right to the bottom of the list, then move up one.

6. Click `Next >`

7. Change the delimiter to just `Comma`

8. Either click `Finish` or `Next >` if you want to further customise the import.

Troubleshooting

10.1 General Docker troubleshooting

All commands should be run from the folder where `docker-compose.yml` is.

To list the active containers:

```
$ docker-compose ps
```

To list active containers running anywhere on the system:

```
$ docker ps
```

To start the containers and detach (so you get the command prompt back instead of seeing all the logging):

```
$ docker-compose up -d
```

To stop the containers:

```
$ docker-compose down
```

To see logs of all the containers in follow mode (`-f`) and with timestamps (`-t`):

```
$ docker-compose logs -ft
```

To see logs of just one container in follow mode - use the service name from the `docker-compose.yml` file, choose from `openrem`, `db` (PostgreSQL), `nginx` (web server), `orthanc_1` (DICOM server):

```
$ docker-compose logs -f orthanc_1
```

10.2 Other Docker errors

10.2.1 Errors at docker-compose up

Cannot start service nginx

Error message when running `docker-compose up -d` (example from Windows):

```
ERROR: for nginx Cannot start service nginx: driver failed programming external
→connectivity on endpoint
openrem-nginx (...): Error starting userland proxy: listen tcp 0.0.0.0:80: bind: An
→attempt was made to access a
socket in a way forbidden by its access permissions.
ERROR: Encountered errors while bringing up the project.
```

This error indicates port 80 is not available for Docker/OpenREM. Try:

- Shutting down other web servers, such as IIS

- Using an alternative port temporarily for OpenREM:

    - Edit the `docker-compose.yml` file

    - Find the section that includes

    ```
    nginx:
      ports:
        - 80:80
    ```

    - Change the external facing port to a high number, for example:

    ```
    nginx:
      ports:
        - 8080:80
    ```

Now stop and start the containers again:

```
$ docker-compose down
$ docker-compose up -d
```

If there are no errors, check that the containers are up and which ports are in use:

```
$ docker-compose ps
```

Connection was reset, Orthanc restarting

- After installation, browsing to the webservice reports „The connection was reset“.

- `docker-compose ps` reports:

```
openrem-orthanc-1                      /docker-entrypoint.sh /tmp ...   Restarting
```

- Orthanc Docker logs include:

```
openrem-orthanc-1 | E1208 12:51:29.599961 OrthancException.cpp:57] The specified path␣
→does not point to a regular file: The path does not point to a regular file: /etc/
→share/orthanc/scripts/openrem_orthanc_config_docker.lua
openrem-orthanc-1 | E1208 12:51:29.600051 ServerIndex.cpp:706] INTERNAL ERROR:␣
→ServerIndex::Stop() should be invoked manually to avoid mess in the destruction order!
```

This might indicate that the bind mounts have not worked. This might be due to SELinux, particularly if you are using Red Hat or Fedora or related distributions.

See *Docker SELinux configuration*

10.3 OpenREM log files

Log file location, naming and verbosity were configured in the `.env.prod` configuration - see the *Docker env configuration* configuration docs for details.

The `openrem.log` has general logging information, the other two are specific to the DICOM store and DICOM query-retrieve functions if you are making use of them.

You can increase the verbosity of the log files by changing the log ‚level‘ to `DEBUG`, or you can decrease the verbosity to `WARNING`, `ERROR`, or `CRITICAL`. The default is `INFO`.

To list the OpenREM log folder (with details, sorted with newest at the bottom, ‚human‘ file sizes):

```
$ docker-compose exec openrem ls -rlth /logs
```

To review the `openrem.log` file for example:

```
$ docker-compose exec openrem more /logs/openrem.log
```

10.4 Older stuff

10.4.1 Server 500 errors

Turn on debug mode

This will render a debug report in the browser - usually revealing the problem.

Docker installs

Edit the `.env.prod` file. Find the following line and change it from `0` to `1`:

```
DEBUG=1
```

Restart the containers using a command line in the folder containing your installation. This might be enough:

```
docker-compose up -d
```

If the webserver fails, then restart all the containers:

```
docker-compose down
docker-compose up -d
```

Non-Docker installs

Locate and edit your local_settings file

```
nano /var/dose/veopenrem3/lib/python3.8/site-packages/openrem/local_settings.py
```

Find the following line and make it active:

```
DEBUG = True
```

Restart the web service:

```
sudo systemctl reload openrem-gunicorn.service
```

Returning to normal mode

You should always disable debug mode when you have fixed the error. If you leave debug mode in place, the system is likely to run out of memory as database queries are cached in this mode.

Docker:

- Edit `.env.prod` to set `DEBUG=0`

- Restart `docker-compose`

Non-docker:

- Edit `local_settings.py` again to comment out the DEBUG line (add a # to the start) or set it to `False`

- Reload the web service

10.4.2 Fixing accumulated AGD and laterality for Hologic DBT

The code for extracting dose related information from Hologic digital breast tomosynthesis proprietary projection images object used an incorrect tag to extract the laterality of the image in releases before 0.8.0 in June 2018. As a result the accumulated AGD code didn't work, so the accumulated AGD cell on the mammography summary sheets remained blank.

Releases between 0.8.0 and 0.10.0 had instructions on how to rectify this for existing studies in the database, but these instructions are not suitable for version 1.0 and later and therefore these instructions have been removed.

If you have a modality where every study has one event (usually CT), review

If planar X-ray studies are appearing in fluoroscopy or vice-versa, review

- *Display names and user-defined modalities*

For DICOM networking:

- *Troubleshooting: openrem_qr.log* for query retrieve

- *Troubleshooting: openrem_store.log* for DICOM store

For task management:

- *Task management*

10.5 Log files

10.6 Starting again!

If for any reason you want to start again with the database, then this is how you might do it:

10.6.1 SLQite3 database

- Delete or rename your existing database file (location will be described in your `local_settings.py` file)
- *? ref to database creation here, if SQLite3 features anywhere?*

10.6.2 Any database

These instructions will also allow you to keep any user settings if you use an SQLite3 database.

In a shell/command window, move into the openrem folder:

- Ubuntu linux: `cd /usr/local/lib/python2.7/dist-packages/openrem/`
- Other linux: `cd /usr/lib/python2.7/site-packages/openrem/`
- Linux virtualenv: `cd virtualenvfolder/lib/python2.7/site-packages/openrem/`
- Windows: `cd C:\Python27\Lib\site-packages\openrem\`
- Windows virtualenv: `cd virtualenvfolder\Lib\site-packages\openrem\`

Run the django python shell:

```
$ python manage.py shell
```

```
>>> from remapp.models import GeneralStudyModuleAttr
>>> a = GeneralStudyModuleAttr.objects.all()
>>> a.count()  # Just to see that we are doing something!
53423
```

And if you are sure you want to delete all the studies…

```
>>> a.delete()
>>> a.count()
0

>>> exit()
```

Developers

Inhalt:

11.1 Creating a development environment

Install Python 3.6+, preferably Python 3.8: check your Linux distribution docs to see how to install a particular version; for Windows go to https://www.python.org/downloads/. Check „Add Python 3.8 to PATH" during installation.

Install git: `sudo apt install git` or equivalent on Linux; for Windows go to https://git-scm.com/download/win

Recommended - install an integrated development environment such as PyCharm or Visual Studio Code (many others are available).

Recommended - install PostgreSQL database

11.1.1 Check out git repo

Either clone the main OpenREM repository, or fork it first and then clone that (adapt the command accordingly):

```
$ git clone https://bitbucket.org/openrem/openrem.git
```

The OpenREM source code will now be in a folder called openrem. If you wish to specify the folder, you could do this by adding the folder name to the clone command.

11.1.2 Create Python virtual environment

Linux - install the Python package `venv` using `pip` (Windows users, `venv` should have been installed with Python automatically):

```
$ sudo apt install python3-venv
```

Then create the Python virtual environment in a folder called `openrem-venv` (change as required):

Linux:

```
$ python3.8 -m venv openrem-venv
$ . openrem-venv/bin/activate
```

Windows PowerShell (for `cmd.exe` substitute `Activate.ps1` with `activate.bat`)

```
PS C:\Path\To\Coding Folder> C:\Python38\python -m venv openrem-venv
PS C:\Path\To\Coding Folder> .\openrem-venv\Scripts\Activate.ps1
```

For users of VS Code, it can be useful to create the virtual environment in a folder called `.venv` within your project folder (where you checked out the git repo), then VS Code will find it automatically. If you are using PyCharm you can click on the Python interpreter at the bottom right and click ‚Add Interpreter'.

11.1.3 Install the Python libraries

Assumes:

- git repository is in a sub-folder called `openrem` - change as necessary
- venv is activated

```
$ pip install -e openrem/
```

11.1.4 Setup OpenREM

You'll need a basic configuration of OpenREM to run any code locally - copy the `openremproject/local_settings.py.example` to `openremproject/local_settings.py` and set a path for a SQLite database etc.

To use PosgreSQL instead of SQLite3, set up a user in pgAdmin 4 on Windows, and an empty database with the same user as owner, or use the *Konfiguration von Datenbank und OpenREM* instructions on Linux.

11.1.5 Run test webserver

To see the changes you have made with the web interface, you can use the built-in Django webserver:

```
python manage.py runserver --insecure
```

In a web browser on the same computer, go to http://localhost:8000/ - you should now see the message about creating users.

11.1.6 Get coding

Create a branch in the git repository, and start making your changes, adding your features etc!

When you are done, push it back to Bitbucket and send in a pull request! Ideally, try and use the `refs #123` syntax in commit messages to reference the issue on Bitbucket you are working on.

11.2 Running the test suite

TODO: Update for Python 3, OpenREM 1.0

11.2.1 Code formatting and tests

Steps before pushing to Bitbucket. Commands assume you are in the root directory of the git repository, at the same level as README.rst and requirements.txt etc, and that you have activated a virtualenv with the project requirements installed (`pip install -e .`) plus Black (`pip install black`)

Run black against the code:

```
$ black --exclude stuff/ .
```

Check the changes made, edit where necessary. Black is an opinionated Python formatter and in general OpenREM code should be subjected to it. The flake8 tests are tuned to agree with Black.

Run the Django tests:

```
$ python openrem/manage.py test remapp --parallel
```

old stuff to be updated

11.2.2 Preparation

Install the dependencies and OpenREM

OpenREM is a Django application, and therefore we use Django's test-execution framework to test OpenREM.

The first thing to do is to create a local copy of the git repository, then install all of OpenREM's dependencies in a virtualenv.

You will need `python`, `pip`, `git` and `virtualenv` installed - see the links on the : doc : *install-prep* docs for the latter, but you might try `pip install virtualenv`.

```
mkdir openremrepo
git clone https://bitbucket.org/openrem/openrem.git openremrepo
```

Now create the virtualenv:

```
mkdir veOpenREM
virtualenv veOpenREM
. veOpenREM/bin/activate  # Linux
veOpenREM\Scripts\activate  # Windows
```

At this stage there should be a (`veOpenREM`) prefix to our prompt telling us the virtualenv is activated.

Now install the dependencies:

```
pip install -e openremrepo/
pip install https://bitbucket.org/edmcdonagh/pynetdicom/get/default.tar.gz
→#egg=pynetdicom-0.8.2b2
```

In the future it might be necessary to install numpy too for testing.

OpenREM konfigurieren

Rename and configure `openremproject/local_settings.py.example` and `openremproject/wsgi.py.example` as per the : doc :*install* docs.

Create a database following the same : doc :*install* instructions.

11.2.3 Run the tests!

Making sure the virtualenv is activated, move to `openremrepo/openrem` and run:

```
python manage.py test remapp
```

All the tests that exit in `openrem/remapp/tests/` will now be run.

11.2.4 Related tools

Enabling django-debug-toolbar

See *Enabling debug toolbar*

11.2.5 Creating test versions of production systems

If you wish to create a duplicate install to test upgrades etc, refer to *Database restore* and the preceding text regarding making backups.

11.3 Translating OpenREM strings

OpenREM's primary language is British English (en_GB). Users and developers with knowledge of other languages can create translations of the interface strings, export file strings and the documentation. These will then be exposed when the web browser language is set to match the new translation language (OpenREM interface) or when the language is selected for the documentation.

A web-based service for managing translations has kindly been provided to OpenREM by Weblate. Their hosting is free to OpenREM, and they welcome donations.

11.3.1 Translators

- Create an account at https://hosted.weblate.org

- The OpenREM project is at https://hosted.weblate.org/projects/openrem/

- Each page in the Read The Docs documentation (https://docs.openrem.org) is a separate ‚component' in Weblate, and they have been named ‚RTD document name'. The web interface strings are all in one ‚component'.

- Choose a component, and on the next page you can select one of the existing translations which you can review, edit and propose new translation strings.

- Once approved, they will be merged in by developers

Creating new language translations

At the component level, you will see an option to create a new translation. This might need to be done for each component individually.

Code syntax in strings

Be careful not to edit code syntax within strings. For example, Python code might be:

```
Writing study {row} of {numrows} to All data sheet and individual protocol sheets
```

This is translated into Norwegian Bokmål as:

```
Skriver studie av {row} av {numrows} til alle datablad og individuelle protokollblader
```

Notice that the {} and their contents is unchanged - but may be moved around within the sentence to produce the correct grammar for the language being used.

Similarly with Django HTML template strings:

```
Number in last %(day_delta)s days
```

becomes:

```
Antall de siste %(day_delta)s dagene
```

It is essential that the `%()s` as well as the string inside the brackets stay intact.

For the RTD translations, there will be Sphinx codes that should be left untranslated, for example:

```
:ref:`genindex`
```

11.3.2 Developers

Install pre-requisites

gettext

Linux: `sudo apt install gettext` or equivalent for your distribution. For Windows: download a precompiled binary installer

sphinx-intl

Activate development environment - see *Creating a development environment* for details - and add the sphinx packages:

```
$ pip install sphinx
$ pip install sphinx-intl
$ pip install sphinx-argparse
$ pip install sphinx_issues
$ pip install sphinx_copybutton
```

Update .pot and .po files

Activate the development environment and move to the root of the OpenREM repository - with the `docs` folder and `openrem` folder etc:

```
$ cd docs/
$ mkdir _static
$ sphinx-build -b gettext . _build/gettext/
$ sphinx-intl update -p _build/gettext/
$ cd ../openrem/
$ django-admin makemessages -a --keep-pot
```

Adding new interface strings for translation

Please refer to https://docs.djangoproject.com/en/2.2/topics/i18n/translation/ for instructions.

In brief, the following will help get you started, but does not cover lazy translations, plurals and many other things!

All the Sphinx/Read The Docs strings are translatable - if a page does not appear in Weblate that is because it has not been configured as a component there yet.

Python code

First, import `gettext` from Django:

```
from django.utils.translation import gettext as _
```

Then wrap strings to be translated with _() so

```
query.stage = "Checking to see if any response studies are already in the OpenREM
↪database"
```

becomes

```
query.stage = _(
    "Checking to see if any response studies are already in the OpenREM database"
)
```

The same is done for strings that contain variables. Unfortunately `gettext` cannot work with f-strings so we are stuck with `.format()` instead. It is easier to understand how to translate the text though if we use named variables rather than position based ones, like this:

```
query.stage = _("Filter at {level} level on {filter_name} that {filter_type} {filter_
↪list}".format(
    level=level, filter_name=filter_name, filter_type=filter_type, filter_list=filter_
↪list
))
```

Remember we cannot assume the grammar of the translated string so try and pass the whole sentence or paragraph to be translated.

Template code

Add the following at the top of the template file, just after any `extends` code:

```
{% load i18n %}
```

This can be done with *inline* translations and *block* translations. For inline,

```
<th style="width:25%">System name</th>
```

becomes

```
<th style="width:25%">{% trans "System name" %}</th>
```

If there are variables, a block translation is required, for example:

```
{% if home_config.display_workload_stats %}
    <th style="width:12.5%">{% blocktrans with home_config.day_delta_a as day_delta␣
↪trimmed %}
        Number in last {{ day_delta }} days{% endblocktrans %}</th>
    <th style="width:12.5%">{% blocktrans with home_config.day_delta_b as day_delta␣
↪trimmed %}
        Number in last {{ day_delta }} days{% endblocktrans %}</th>
{% endif %}
```

Comments can be added to aid translators, for example:

```
{# Translators: Number of studies in DB listed above home-page table. No final full-stop␣
↪in English due to a.m./p.m. #}
{% now "DATETIME_FORMAT" as current_time %}
{% blocktrans with total_studies=homedata.total trimmed%}
    There are {{ total_studies }} studies in this database. Page last refreshed on {{␣
↪current_time }}
{% endblocktrans %}
```

Making use of updated strings on local system

Specify the language to build for Sphinx docs, eg for German:

```
$ sphinx-build -b html -D language=de . _build/html/de
```

For Django strings:

```
$ django-admin compilemessages
```

11.3.3 Incorporating translations into main repo

In the git repository:

```
$ git remote add weblate https://hosted.weblate.org/git/openrem/web-interface/
```

- Checkout the `weblate\develop` branch as a new local branch
- Push the branch to Bitbucket
- Create a pull request to develop

11.4 Enabling debug toolbar

Django Debug Toolbar can be very useful when troubleshooting or optimising the web interface, showing all the queries that have been run, the timings and lots more.

More information about Django Debug Toolbar can be found at https://django-debug-toolbar.readthedocs.io

11.4.1 Installation

- Activate the virtualenv (assuming you are using one. . .)
- Install from pip:

```
pip install django-debug-toolbar
```

11.4.2 Configuration

- Open `openremproject/local_settings.py` and add the lines:

```
MIDDLEWARE += ['debug_toolbar.middleware.DebugToolbarMiddleware',]
INSTALLED_APPS += ('debug_toolbar',)
INTERNAL_IPS = ['127.0.0.1']
```

If you wish to make use of the debug toolbar on machines other than the one the code is running on, change the INTERNAL_IPS address list to include your client machine.

11.4.3 Using Django Debug Toolbar

When `DEBUG = True` in `openremproject/local_settings.py` the toolbar should appear.

11.5 DICOM import modules

11.5.1 RDSR module

Ultimately this should be the only module required as it deals with all Radiation Dose Structured Reports. This is used for CT, fluoroscopy, mammography, digital radiography and nuclear medicine.

openrem.remapp.extractors.rdsr.**rdsr**(*rdsr_file*)

> Extract radiation dose related data from DICOM Radiation SR objects.
>
> > **Parameter**
> > **rdsr_file** (*str.*) – relative or absolute path to Radiation Dose Structured Report.

11.5.2 Mammography module

Mammography is interesting in that all the information required for dose audit is contained in the image header, including patient ‚size', ie thickness. However the disadvantage over an RSDR is the requirement to process each individual image rather than a single report for the study, which would also capture any rejected images.

openrem.remapp.extractors.mam.**mam**(*mg_file*)

> Extract radiation dose structured report related data from mammography images
>
> > **Parameter**
> > **mg_file** (*str.*) – relative or absolute path to mammography DICOM image file.

11.5.3 CR and DR module

In practice this is only useful for DR modalities, but most of them use the CR IOD instead of the DX one, so both are catered for. This module makes use of the image headers much like the mammography module.

openrem.remapp.extractors.dx.**dx**(*dig_file*)

> Extract radiation dose structured report related data from DX radiographic images
>
> > **Parameter**
> > **filename** (*str.*) – relative or absolute path to DICOM DX radiographic image file.

11.5.4 NM Image module

This has the abilty to read information from the DICOM Headers of PET and NM images. In contrast to the other import modules this may actually complement the data read from an RRDSR, because not all relevant data is included there.

openrem.remapp.extractors.nm_image.**nm_image**(*filename: str*)

> Extract radiation dose related data from DICOM PET/NM-Image.
>
> > **Parameter**
> > **filename** – relative or absolute path to PET/NM-Image.

11.5.5 CT non-standard modules

Philips CT dose info reports

These have all the information that could be derived from the images also held in the DICOM header information, making harvesting relatively easy. Used where RDSR is not available from older Philips systems.

openrem.remapp.extractors.ct_philips.**ct_philips**(*philips_file*)

> Extract radiation dose structured report related data from Philips CT dose report images

> > **Parameter**
> > **filename** (`str.`) – relative or absolute path to Philips CT dose report DICOM image file.

> **Tested with:**

> > • Philips Gemini TF PET-CT v2.3.0
> >
> > • Brilliance BigBore v3.5.4.17001.

Toshiba dose summary and images

OpenREM can harvest information from older Toshiba CT systems that create dose summary images but cannot create RDSR objects by using a combination of tools to create an RDSR that can then be imported in the normal manner. This extractor requires that the Offis DICOM toolkit, java.exe and pixelmed.jar are available to the system.

openrem.remapp.extractors.ct_toshiba.**ct_toshiba**(*folder_name*)

> Function to create radiation dose structured reports from a folder of dose images.

> > **Parameter**
> > **folder_name** – Path to folder containing Toshiba DICOM objects - dose summary and images

11.6 Non-DICOM import modules

11.6.1 Patient height and weight csv import module

This module enables a csv file to be parsed and the height and weight information extracted and added to existing studies in the OpenREM database. An example may be a csv extract from a RIS or EPR system.

There needs to be a common unique identifier for the exam - currently this is limited to accession number or study instance UID.

openrem.remapp.extractors.ptsizecsv2db.**csv2db**()

> Import patient height and weight data from csv RIS exports. Called from `openrem_ptsizecsv.py` script

> > **Parameter**
> > **args** – sys.argv from the command line call

> Example:

```
openrem_ptsizecsv.py -s MyRISExport.csv StudyInstanceUID height weight
```

openrem.remapp.extractors.ptsizecsv2db.**websizeimport**(*csv_pk=None*)

> Task to import patient size data from the OpenREM web interface.

Parameter
> **csv_pk** – Database index key for the import record, containing the path to the import csv file and the field header details.

11.7 Export from database

11.7.1 Multi-sheet Microsoft Excel XLSX exports

This export has a summary sheet of all the requested and performed protocols and the series protocols. The next sheet has all studies on, one study per line, with the series stretching off to the right. The remaining sheets are specific to each series protocol, in alphabetical order, with one series per line. If one study has three series with the same protocol name, each one has a line of its own. For NM this is not true as protocol names do not exist, sheets are generated by study description instead.

remapp.exports.rf_export.**rfxlsx**(*filterdict*, *pid=False*, *name=None*, *patid=None*, *user=None*)

> Export filtered RF database data to multi-sheet Microsoft XSLX files.

> **Parameter**
> - **filterdict** – Queryset of studies to export
> - **pid** – does the user have patient identifiable data permission
> - **name** – has patient name been selected for export
> - **patid** – has patient ID been selected for export
> - **user** – User that has started the export

> **Rückgabe**
> Saves xlsx file into Media directory for user to download

remapp.exports.ct_export.**ctxlsx**(*filterdict*, *pid=False*, *name=None*, *patid=None*, *user=None*)

> Export filtered CT database data to multi-sheet Microsoft XSLX files

> **Parameter**
> - **filterdict** – Queryset of studies to export
> - **pid** – does the user have patient identifiable data permission
> - **name** – has patient name been selected for export
> - **patid** – has patient ID been selected for export
> - **user** – User that has started the export

> **Rückgabe**
> Saves xlsx file into Media directory for user to download

remapp.exports.dx_export.**dxxlsx**(*filterdict*, *pid=False*, *name=None*, *patid=None*, *user=None*)

> Export filtered DX and CR database data to multi-sheet Microsoft XSLX files.

> **Parameter**
> - **filterdict** – Queryset of studies to export
> - **pid** – does the user have patient identifiable data permission
> - **name** – has patient name been selected for export
> - **patid** – has patient ID been selected for export

• **user** – User that has started the export

Rückgabe
Saves xlsx file into Media directory for user to download

remapp.exports.mg_export.**exportMG2excel**(*filterdict*, *pid=False*, *name=None*, *patid=None*, *user=None*, *xlsx=False*)

Export filtered mammography database data to a single-sheet CSV file or a multi sheet xlsx file.

Parameter

• **filterdict** – Queryset of studies to export

• **pid** – does the user have patient identifiable data permission

• **name** – has patient name been selected for export

• **patid** – has patient ID been selected for export

• **user** – User that has started the export

• **xlsx** – Whether to export a single sheet csv or a multi sheet xlsx

Rückgabe
Saves csv file into Media directory for user to download

remapp.exports.nm_export.**exportNM2excel**(*filterdict*, *pid=False*, *name=None*, *patid=None*, *user=None*)

Parameter

• **filterdict** – Queryset of studies to export

• **pid** – does the user have patient identifiable data permission

• **name** – has patient name been selected for export

• **patid** – has patient ID been selected for export

• **user** – User that has started the export

Rückgabe
Saves xlsx file into Media directory for user to download

11.7.2 Single sheet CSV exports

remapp.exports.rf_export.**exportFL2excel**(*filterdict*, *pid=False*, *name=None*, *patid=None*, *user=None*)

Export filtered fluoro database data to a single-sheet CSV file.

Parameter

• **filterdict** – Queryset of studies to export

• **pid** – does the user have patient identifiable data permission

• **name** – has patient name been selected for export

• **patid** – has patient ID been selected for export

• **user** – User that has started the export

Rückgabe
Saves csv file into Media directory for user to download

remapp.exports.ct_export.**ct_csv**(*filterdict*, *pid=False*, *name=None*, *patid=None*, *user=None*)

Export filtered CT database data to a single-sheet CSV file.

> **Parameter**
>> - **filterdict** – Queryset of studies to export
>> - **pid** – does the user have patient identifiable data permission
>> - **name** – has patient name been selected for export
>> - **patid** – has patient ID been selected for export
>> - **user** – User that has started the export
>
> **Rückgabe**
>> Saves csv file into Media directory for user to download

remapp.exports.dx_export.**exportDX2excel**(*filterdict*, *pid=False*, *name=None*, *patid=None*, *user=None*)

Export filtered DX database data to a single-sheet CSV file.

> **Parameter**
>> - **filterdict** – Queryset of studies to export
>> - **pid** – does the user have patient identifiable data permission
>> - **name** – has patient name been selected for export
>> - **patid** – has patient ID been selected for export
>> - **user** – User that has started the export
>
> **Rückgabe**
>> Saves csv file into Media directory for user to download

remapp.exports.nm_export.**exportNM2csv**(*filterdict*, *pid=False*, *name=None*, *patid=None*, *user=None*)

> **Parameter**
>> - **filterdict** – Queryset of studies to export
>> - **pid** – does the user have patient identifiable data permission
>> - **name** – has patient name been selected for export
>> - **patid** – has patient ID been selected for export
>> - **user** – User that has started the export
>
> **Rückgabe**
>> Saves csv file Media directory for user to download

Specialised csv exports - NHSBSP formatted mammography export

remapp.exports.mg_csv_nhsbsp.**mg_csv_nhsbsp**(*filterdict*, *user=None*)

Export filtered mammography database data to a NHSBSP formatted single-sheet CSV file.

> **Parameter**
>> **filterdict** (*dict*) – Dictionary of query parameters from the mammo filtered page URL.
>
> **Rückgabe**
>> None - file is saved to disk and location is stored in database

11.8 Tools and helper modules

11.8.1 Get values

Tiny modules to reduce repetition in the main code when extracting information from DICOM headers using pydicom.

openrem.remapp.tools.get_values.**get_keys_by_value**(*dict_of_elements*, *value_to_find*)

> Get a list of keys from a dictionary which have the given value :param dict_of_elements: a dictionary of elements :param value_to_find: the value to look for in the dictionary :return: list of key names matching the given value

openrem.remapp.tools.get_values.**get_or_create_cid**(*codevalue*, *codemeaning*)

> Create a code_value code_meaning pair entry in the ContextID table if it doesn't already exist.
>
> > **Parameter**
> >
> > - **codevalue** (*int.*) – Code value as defined in the DICOM standard part 16
> >
> > - **codemeaning** – Code meaning as defined in the DICOM standard part 16
> >
> > **Rückgabe**
> > ContextID entry for code value passed

openrem.remapp.tools.get_values.**get_seq_code_meaning**(*sequence*, *dataset*)

> From a DICOM sequence, get the code meaning.
>
> > **Parameter**
> >
> > - **sequence** (*DICOM keyword, no spaces or plural as per dictionary.*) – DICOM sequence name.
> >
> > - **dataset** (*DICOM dataset*) – The DICOM dataset containing the sequence.
> >
> > **Rückgabe**
> > str. – code meaning

openrem.remapp.tools.get_values.**get_seq_code_value**(*sequence*, *dataset*)

> From a DICOM sequence, get the code value.
>
> > **Parameter**
> >
> > - **sequence** (*DICOM keyword, no spaces or plural as per dictionary.*) – DICOM sequence name.
> >
> > - **dataset** (*DICOM dataset*) – The DICOM dataset containing the sequence.
> >
> > **Rückgabe**
> > int. – code value

openrem.remapp.tools.get_values.**get_value_kw**(*tag*, *dataset*)

> Get DICOM value by keyword reference.
>
> > **Parameter**
> >
> > - **tag** (*str.*) – DICOM keyword, no spaces or plural as per dictionary.
> >
> > - **dataset** (*dataset*) – The DICOM dataset containing the tag.
> >
> > **Rückgabe**
> > str. – value

openrem.remapp.tools.get_values.**get_value_num**(*tag*, *dataset*)

> Get DICOM value by tag group and element number.
>
> Always use get_value_kw by preference for readability. This module can be required when reading private elements.
>
> > **Parameter**
> >
> > - **tag** (*hex*) – DICOM group and element number as a single hexadecimal number (prefix 0x).
> >
> > - **dataset** (*dataset*) – The DICOM dataset containing the tag.
> >
> > **Rückgabe**
> > str. – value

openrem.remapp.tools.get_values.**list_to_string**(*dicom_value*)

> Turn multivalue names into a single string for correct encoding and pretty reproduction :param dicom_value: returned DICOM value, usually a name field. Might be single (string) or multivalue (list) :return: string of name(s)

openrem.remapp.tools.get_values.**return_for_export**(*model*, *field*)

> Prevent errors due to missing data in models :param model: database table :param field: database field :return: value or None

openrem.remapp.tools.get_values.**test_numeric_value**(*string_number*)

> Tests if string can be converted to a float. If it can, return it :param string_number: string to test if is a number :return: string if number, nothing otherwise

openrem.remapp.tools.get_values.**to_decimal_value**(*string_number*)

> Tests if string can be converted to a float. If yes returns it as decimal. :param string_number: string to test if a number :return: Decimal if convertable, None otherwise

11.8.2 Check if UID exists

Small module to check if UID already exists in the database.

openrem.remapp.tools.check_uid.**check_uid**(*uid*, *level='Study'*)

> Check if UID already exists in database.
>
> > **Parameter**
> > **uid** (*str.*) – Study UID.
> >
> > **Rückgabe**
> > 1 if it does exist, 0 otherwise

openrem.remapp.tools.check_uid.**record_sop_instance_uid**(*study*, *sop_instance_uid*)

> Record the object's SOP Instance UID so we can ignore it next time. If an object does need to be imported again, the original one needs to be deleted first.
>
> > **Parameter**
> >
> > - **study** – GeneralStudyModuleAttr database object
> >
> > - **sop_instance_uid** – SOP Instance UID of object being imported
> >
> > **Rückgabe**

11.8.3 DICOM time and date values

Module to convert betweeen DICOM and Python dates and times.

openrem.remapp.tools.dcmdatetime.**get_date**(*tag*, *dataset*)

> Get DICOM date string and return Python date.
>
> > **Parameter**
> >
> > - **tag** (`str.`) – DICOM keyword, no spaces or plural as per dictionary.
> >
> > - **dataset** (`dataset`) – The DICOM dataset containing the tag.
> >
> > **Rückgabe**
> > Python date value

openrem.remapp.tools.dcmdatetime.**get_date_time**(*tag*, *dataset*)

> Get DICOM date time string and return Python date time.
>
> > **Parameter**
> >
> > - **tag** (`str.`) – DICOM keyword, no spaces or plural as per dictionary.
> >
> > - **dataset** (`dataset`) – The DICOM dataset containing the tag.
> >
> > **Rückgabe**
> > Python date time value

openrem.remapp.tools.dcmdatetime.**get_time**(*tag*, *dataset*)

> Get DICOM time string and return Python time.
>
> > **Parameter**
> >
> > - **tag** (`str.`) – DICOM keyword, no spaces or plural as per dictionary.
> >
> > - **dataset** (`dataset`) – The DICOM dataset containing the tag.
> >
> > **Rückgabe**
> > python time value

openrem.remapp.tools.dcmdatetime.**make_date**(*dicomdate*)

> Given a DICOM date, return a Python date.
>
> > **Parameter**
> > **dicomdate** (`str.`) – DICOM style date.
> >
> > **Rückgabe**
> > Python date value

openrem.remapp.tools.dcmdatetime.**make_date_time**(*dicomdatetime*)

> Given a DICOM date time, return a Python date time.
>
> > **Parameter**
> > **dicomdate** (`str.`) – DICOM style date time.
> >
> > **Rückgabe**
> > Python date time value

openrem.remapp.tools.dcmdatetime.**make_dcm_date**(*pythondate*)

> Given a Python date, return a DICOM date :param pythondate: Date :type pythondate: Python date object :returns: DICOM date as string

openrem.remapp.tools.dcmdatetime.**make_dcm_date_range**(*date1=None*, *date2=None*, *single_date=False*)

Given one or two dates of the form yyyy-mm-dd, return a DICOM date range.

> **Parameter**
>
>> - **date1** – Date from, string, yyyy-mm-dd, 1900-01-01 if None or badly formatted
>>
>> - **date2** – Date until, string, yyyy-mm-dd, today if None or badly formatted
>>
>> - **single_date** – Single date range, bool, default False
>
> **Rückgabe**
>> DICOM formatted date range or single date

openrem.remapp.tools.dcmdatetime.**make_dcm_time**(*python_time*)

Return DICOM formatted time without seconds from python time

> **Parameter**
>> **python_time** – Python datetime.time object
>
> **Rückgabe**
>> string, %H%M

openrem.remapp.tools.dcmdatetime.**make_dcm_time_range**(*time1=None*, *time2=None*)

Given one or two times of the format 0123, return DICOM formatted time range (without seconds)

> **Parameter**
>
>> - **time1** – time, format 0123, 0000 if None
>>
>> - **time2** – time, format 0123, 2359 if None
>
> **Rückgabe**
>> time range, string, format 0123-1234

openrem.remapp.tools.dcmdatetime.**make_time**(*dicomtime*)

Given a DICOM time, return a Python time.

> **Parameter**
>> **dicomdate** (*str.*) – DICOM style time.
>
> **Rückgabe**
>> Python time value

11.8.4 Test for QA or other non-patient related studies

openrem.remapp.tools.not_patient_indicators.**get_not_pt**(*dataset*)

Looks for indications that a study might be a test or QA study.

Some values that might indicate a study was for QA or similar purposes are not recorded in the database, for example patient name. Therefore this module attempts to find such indications and creates an xml style string that can be recorded in the database on study import.

> **Parameter**
>> **dataset** (*dataset*) – The DICOM dataset.
>
> **Rückgabe**
>> str. – xml style string if any trigger values are found.

11.9 Models

class remapp.models.**AccumCassetteBsdProjRadiogDose**(**args*, ***kwargs*)

Accumulated Cassette-based Projection Radiography Dose TID 10006

From DICOM Part 16 Correction Proposal CP-1077:
This template provides information on Projection Radiography dose values accumulated on Cassette- based systems over one or more irradiation events (typically a study or a performed procedure step) from the same equipment.

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**AccumIntegratedProjRadiogDose**(**args*, ***kwargs*)

Accumulated Integrated Projection Radiography Dose TID 10007

From DICOM Part 16 Correction Proposal CP-1077:
This template provides information on Projection Radiography dose values accumulated on Integrated systems over one or more irradiation events (typically a study or a performed procedure step) from the same equipment.

exception DoesNotExist

exception MultipleObjectsReturned

convert_gym2_to_cgycm2()
Converts Gy.m2 to cGy.cm2 for display in web interface

total_dap_delta_gym2_to_cgycm2()
Converts total DAP over delta days from Gy.m2 to cGy.cm2 for display in web interface

class remapp.models.**AccumMammographyXRayDose**(**args*, ***kwargs*)

Accumulated Mammography X-Ray Dose TID 10005

From DICOM Part 16:
This modality specific template provides detailed information on mammography X-Ray dose value accumulations over several irradiation events from the same equipment (typically a study or a performed procedure step).

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**AccumProjXRayDose**(**args*, ***kwargs*)

Accumulated Fluoroscopy and Acquisition Projection X-Ray Dose TID 10004

From DICOM Part 16:
This general template provides detailed information on projection X-Ray dose value accumulations over several irradiation events from the same equipment (typically a study or a performed procedure step).

exception DoesNotExist

exception MultipleObjectsReturned

acq_gym2_to_cgycm2()
Converts acquisition DAP total from Gy.m2 to cGy.cm2 for display in web interface

fluoro_gym2_to_cgycm2()

> Converts fluoroscopy DAP total from Gy.m2 to cGy.cm2 for display in web interface

class remapp.models.**AccumXRayDose**(*args*, **kwargs*)

Accumulated X-Ray Dose TID 10002

From DICOM Part 16:

> This general template provides detailed information on projection X-Ray dose value accumulations over several irradiation events from the same equipment (typically a study or a performed procedure step).

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**AdminTaskQuestions**(*args*, **kwargs*)

Record if admin tasks have been dealt with

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**BackgroundTask**(*id*, *uuid*, *proc_id*, *task_type*, *info*, *error*, *completed_successfully*, *complete*, *started_at*)

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**BackgroundTaskMaximumRows**(*args*, **kwargs*)

Table to store the maximum number of rows allowed in the BackgroundTask table

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**BillingCode**(*id*, *radiopharmaceutical_administration_event_data*, *billing_code*)

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**Calibration**(*args*, **kwargs*)

Table to hold the calibration information

- Container in TID 10002 Accumulated X-ray dose

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**ContextID**(*args*, **kwargs*)

Table to hold all the context ID code values and code meanings.

- Could be prefilled from the tables in DICOM 3.16, but is actually populated as the codes occur. This assumes they are used correctly.

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**CtAccumulatedDoseData**(**args*, ***kwargs*)

    CT Accumulated Dose Data

    From DICOM Part 16:

        This general template provides detailed information on CT X-Ray dose value accumulations over several irradiation events from the same equipment and over the scope of accumulation specified for the report (typically a Study or a Performed Procedure Step).

    exception DoesNotExist

    exception MultipleObjectsReturned

class remapp.models.**CtDoseCheckDetails**(**args*, ***kwargs*)

    CT Dose Check Details TID 10015

    From DICOM Part 16:

        This template records details related to the use of the NEMA Dose Check Standard (NEMA XR-25-2010).

    exception DoesNotExist

    exception MultipleObjectsReturned

class remapp.models.**CtIrradiationEventData**(**args*, ***kwargs*)

    CT Irradiation Event Data TID 10013

    From DICOM Part 16:

        This template conveys the dose and equipment parameters of a single irradiation event.

    Additional to the template:

        • date_time_started

        • series_description

    exception DoesNotExist

    exception MultipleObjectsReturned

class remapp.models.**CtRadiationDose**(**args*, ***kwargs*)

    CT Radiation Dose TID 10011

    From DICOM Part 16:

        This template defines a container (the root) with subsidiary content items, each of which corresponds to a single CT X-Ray irradiation event entry. There is a defined recording observer (the system or person responsible for recording the log, generally the system). Accumulated values shall be kept for a whole Study or at least a part of a Study, if the Study is divided in the workflow of the examination, or a performed procedure step. Multiple CT Radiation Dose objects may be created for one Study.

    exception DoesNotExist

    exception MultipleObjectsReturned

class remapp.models.**CtReconstructionAlgorithm**(**args*, ***kwargs*)

    Container in TID 10013 to hold CT reconstruction methods

    exception DoesNotExist

    exception MultipleObjectsReturned

class remapp.models.**CtXRaySourceParameters**(**args*, ***kwargs*)

    Container in TID 10013 to hold CT x-ray source parameters

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**DeviceParticipant**(**args*, ***kwargs*)
  Device Participant TID 1021

  From DICOM Part 16:
    This template describes a device participating in an activity as other than an observer or subject. E.g. for a dose report documenting an irradiating procedure, participants include the irradiating device.

  exception DoesNotExist

  exception MultipleObjectsReturned

class remapp.models.**DicomDeleteSettings**(**args*, ***kwargs*)
  Table to store DICOM deletion settings

  exception DoesNotExist

  exception MultipleObjectsReturned

class remapp.models.**DicomQRRspImage**(*id*, *dicom_qr_rsp_series*, *query_id*, *sop_instance_uid*, *instance_number*, *sop_class_uid*, *deleted_flag*, *deleted_reason*)

  exception DoesNotExist

  exception MultipleObjectsReturned

class remapp.models.**DicomQRRspSeries**(*id*, *dicom_qr_rsp_study*, *query_id*, *series_instance_uid*, *series_number*, *series_time*, *modality*, *series_description*, *number_of_series_related_instances*, *station_name*, *sop_class_in_series*, *image_level_move*, *deleted_flag*, *deleted_reason*)

  exception DoesNotExist

  exception MultipleObjectsReturned

class remapp.models.**DicomQRRspStudy**(*id*, *dicom_query*, *query_id*, *study_instance_uid*, *modality*, *modalities_in_study*, *study_description*, *number_of_study_related_series*, *sop_classes_in_study*, *station_name*, *deleted_flag*, *deleted_reason*)

  exception DoesNotExist

  exception MultipleObjectsReturned

class remapp.models.**DicomQuery**(**args*, ***kwargs*)
  Table to store DICOM query settings

  exception DoesNotExist

  exception MultipleObjectsReturned

class remapp.models.**DicomRemoteQR**(**args*, ***kwargs*)
  Table to store DICOM remote QR settings

  exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**DicomStoreSCP**(**args*, ***kwargs*)

Table to store DICOM store settings

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**DoseRelatedDistanceMeasurements**(**args*, ***kwargs*)

Dose Related Distance Measurements Context ID 10008

Called from TID 10003c

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**DrugProductIdentifier**(*id*, *radiopharmaceutical_administration_event_data*, *drug_product_identifier*)

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**Exports**(**args*, ***kwargs*)

Table to hold the export status and filenames

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**Exposure**(**args*, ***kwargs*)

In TID 10003b. Code value 113736 (uA.s)

exception DoesNotExist

exception MultipleObjectsReturned

convert_uAs_to_mAs()

Converts uAs to mAs for display in web interface

class remapp.models.**GeneralEquipmentModuleAttr**(**args*, ***kwargs*)

General Equipment Module C.7.5.1

From DICOM Part 3: Information Object Definitions Table C.7-8:
Specifies the Attributes that identify and describe the piece of equipment that produced a Series of Composite Instances.

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**GeneralStudyModuleAttr**(**args*, ***kwargs*)

General Study Module C.7.2.1

Specifies the Attributes that describe and identify the Study performed upon the Patient. From DICOM Part 3: Information Object Definitions Table C.7-3

Additional to the module definition:

- performing_physician_name

- operator_name

- modality_type

- procedure_code_value_and_meaning

- requested_procedure_code_value_and_meaning

exception DoesNotExist

exception MultipleObjectsReturned

dap_a_cgycm2()

    Converts DAP A to cGy.cm2 from Gy.m2 for display or export

dap_b_cgycm2()

    Converts DAP B to cGy.cm2 from Gy.m2 for display or export

dap_delta_weeks_cgycm2()

    Converts DAP delta weeks to cGy.cm2 from Gy.m2 for display

dap_total_cgycm2()

    Converts DAP A+B to cGy.cm2 from Gy.m2 for display or export

class remapp.models.**GlomerularFiltrationRate**(*id*, *radiopharmaceuti-cal_administration_patient_characteristics*, *glomerular_filtration_rate*, *measurement_method*, *equivalent_meaning_of_concept_name*)

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**HighDoseMetricAlertRecipients**(**args*, ***kwargs*)

    Table to store whether users should receive high dose fluoroscopy alerts

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**HighDoseMetricAlertSettings**(**args*, ***kwargs*)

    Table to store high dose fluoroscopy alert settings

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**HomePageAdminSettings**(**args*, ***kwargs*)

    Table to store home page settings

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**ImageViewModifier**(**args*, ***kwargs*)

    Table to hold image view modifiers for the irradiation event x-ray data table

    From DICOM Part 16 Annex D DICOM controlled Terminology Definitions

- Code Value 111032

- Code Meaning Image View Modifier

- Code Definition Modifier for image view

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**IntravenousExtravasationSymptoms**(*id*, *radiopharmaceuti-cal_administration_event_data*, *intravenous_extravasation_symptoms*)

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**IrradEventXRayData**(**args*, ***kwargs*)

Irradiation Event X-Ray Data TID 10003

From DICOM part 16:
This template conveys the dose and equipment parameters of a single irradiation event.

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**IrradEventXRayDetectorData**(**args*, ***kwargs*)

Irradiation Event X-Ray Detector Data TID 10003a

From DICOM Part 16 Correction Proposal CP-1077:
This template contains data which is expected to be available to the X-ray detector or plate reader component of the equipment.

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**IrradEventXRayMechanicalData**(**args*, ***kwargs*)

Irradiation Event X-Ray Mechanical Data TID 10003c

From DICOM Part 16 Correction Proposal CP-1077:
This template contains data which is expected to be available to the gantry or mechanical component of the equipment.

Additional to the template:

- compression_force

- magnification_factor

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**IrradEventXRaySourceData**(**args*, ***kwargs*)

Irradiation Event X-Ray Source Data TID 10003b

From DICOM Part 16 Correction Proposal CP-1077:
This template contains data which is expected to be available to the X-ray source component of the equipment.

Additional to the template:

- ii_field_size

- exposure_control_mode

- grid information over and above grid type

exception DoesNotExist

exception MultipleObjectsReturned

convert_gy_to_mgy()
 Converts Gy to mGy for display in web interface

class remapp.models.**Kvp**(*args*, ***kwargs*)
 In TID 10003b. Code value 113733 (kV)

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**LanguageofContentItemandDescendants**(*id*, *radiopharmaceutical_radiation_dose*, *langua-ge_of_contentitem_and_descendants*, *country_of_language*)

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**MergeOnDeviceObserverUIDSettings**(*args*, ***kwargs*)
 Table to store setting(s) for autmoatic setting of Display Name and Modality type based on same Device observer UID

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**NotPatientIndicatorsID**(*args*, ***kwargs*)
 Table to record strings that indicate a patient ID is really a test or QA ID

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**NotPatientIndicatorsName**(*args*, ***kwargs*)
 Table to record strings that indicate a patient name is really a test or QA name

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**ObjectUIDsProcessed**(*args*, ***kwargs*)
 Table to hold the SOP Instance UIDs of the objects that have been processed against this study to enable duplicate sorting.

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**ObserverContext**(**args*, ***kwargs*)

> Observer Context TID 1002
>
> **From DICOM Part 16:**
>> The observer (person or device) that created the Content Items to which this context applies.
>
> **exception DoesNotExist**
>
> **exception MultipleObjectsReturned**

class remapp.models.**OpenSkinSafeList**(**args*, ***kwargs*)

> Table to store systems names and software versions that are suitable for OpenSkin
>
> **exception DoesNotExist**
>
> **exception MultipleObjectsReturned**

class remapp.models.**OrganDose**(*id*, *radiopharmaceutical_administration_event_data*, *finding_site*, *laterality*, *mass*, *measurement_method*, *organ_dose*, *reference_authority_code*, *reference_authority_text*, *type_of_detector_motion*)

> **exception DoesNotExist**
>
> **exception MultipleObjectsReturned**

class remapp.models.**PETSeries**(*id*, *radiopharmaceutical_radiation_dose*, *series_uid*, *series_datetime*, *number_of_rr_intervals*, *number_of_time_slots*, *number_of_time_slices*, *number_of_slices*, *reconstruction_method*, *coincidence_window_width*, *energy_window_lower_limit*, *energy_window_upper_limit*, *scan_progression_direction*)

> **exception DoesNotExist**
>
> **exception MultipleObjectsReturned**

class remapp.models.**PETSeriesCorrection**(*id*, *pet_series*, *corrected_image*)

> **exception DoesNotExist**
>
> **exception MultipleObjectsReturned**

class remapp.models.**PETSeriesType**(*id*, *pet_series*, *series_type*)

> **exception DoesNotExist**
>
> **exception MultipleObjectsReturned**

class remapp.models.**PKsForSummedRFDoseStudiesInDeltaWeeks**(**args*, ***kwargs*)

> Table to hold foreign keys of all studies that fall within the delta weeks of each RF study.
>
> **exception DoesNotExist**
>
> **exception MultipleObjectsReturned**

class remapp.models.**PatientIDSettings**(**args*, ***kwargs*)

> Table to store patient ID settings
>
> **exception DoesNotExist**

exception MultipleObjectsReturned

class remapp.models.**PatientModuleAttr**(*args*, **kwargs*)

Patient Module C.7.1.1

From DICOM Part 3: Information Object Definitions Table C.7-1:
Specifies the Attributes of the Patient that describe and identify the Patient who is the subject of a diagnostic Study. This Module contains Attributes of the patient that are needed for diagnostic interpretation of the Image and are common for all studies performed on the patient. It contains Attributes that are also included in the Patient Modules in Section C.2.

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**PatientState**(*id*, *radiopharmaceutical_administration_patient_characteristics*, *patient_state*)

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**PatientStudyModuleAttr**(*args*, **kwargs*)

Patient Study Module C.7.2.2

From DICOM Part 3: Information Object Definitions Table C.7-4a:
Defines Attributes that provide information about the Patient at the time the Study started.

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**PersonParticipant**(*args*, **kwargs*)

Person Participant TID 1020

From DICOM Part 16:
This template describes a person participating in an activity as other than an observer or subject. E.g. for a dose report documenting an irradiating procedure, participants include the person administering the irradiation and the person authorizing the irradiation.

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**ProjectionXRayRadiationDose**(*args*, **kwargs*)

Projection X-Ray Radiation Dose template TID 10001

From DICOM Part 16:
This template defines a container (the root) with subsidiary content items, each of which represents a single projection X-Ray irradiation event entry or plane-specific dose accumulations. There is a defined recording observer (the system or person responsible for recording the log, generally the system). A Biplane irradiation event will be recorded as two individual events, one for each plane. Accumulated values will be kept separate for each plane.

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**PulseWidth**(*args*, **kwargs*)

In TID 10003b. Code value 113793 (ms)

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**RadionuclideIdentifier**(*id, radiopharmaceutical_administration_event_data, radionuclide_identifier*)

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**RadiopharmaceuticalAdministrationEventData**(*id, radiopharmaceutical_radiation_dose, radiopharmaceutical_agent, radiopharmaceutical_agent_string, radionuclide, radionuclide_half_life, radiopharmaceutical_specific_activity, radiopharmaceutical_administration_event_uid, estimated_extravasation_activity, radiopharmaceutical_start_datetime, radiopharmaceutical_stop_datetime, administered_activity, effective_dose, radiopharmaceutical_volume, pre_administration_measured_activity, pre_activity_measurement_device, post_administration_measured_activity, post_activity_measurement_device, route_of_administration, site_of, laterality, brand_name, radiopharmaceutical_dispense_unit_identifier, prescription_identifier, comment*)

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**RadiopharmaceuticalAdministrationPatientCharacteristics**(*id*, *radiopharmaceutical_radiation_dose*, *subject_age*, *subject_sex*, *patient_height*, *patient_weight*, *body_surface_area*, *body_surface_area_formula*, *body_mass_index*, *equation*, *glucose*, *fasting_duration*, *hydration_volume*, *recent_physical_activity*, *serum_creatinine*)

> **exception DoesNotExist**

> **exception MultipleObjectsReturned**

class remapp.models.**RadiopharmaceuticalLotIdentifier**(*id*, *radiopharmaceutical_administration_event_data*, *radiopharmaceutical_lot_identifier*)

> **exception DoesNotExist**

> **exception MultipleObjectsReturned**

class remapp.models.**RadiopharmaceuticalRadiationDose**(**args*, ***kwargs*)

Radiopharmaceutical Radiation Dose TID 10021

From DICOM Part 16:
> This Template defines a container (the root) with subsidiary Content Items, each of which corresponds to a single Radiopharmaceutical Administration Dose event entry. There is a defined recording observer (the system and/or person responsible for recording the assay of the radiopharmaceutical, and the person administered the radiopharmaceutical). Multiple Radiopharmaceutical Radiation Dose objects may be created for one study. Radiopharmaceutical Start DateTime in TID 10022 "Radiopharmaceutical Administration Event Data" will convey the order of administrations.

> **exception DoesNotExist**

> **exception MultipleObjectsReturned**

class remapp.models.**ReagentVialIdentifier**(*id*, *radiopharmaceutical_administration_event_data*, *reagent_vial_identifier*)

> **exception DoesNotExist**

> **exception MultipleObjectsReturned**

class remapp.models.**ScanningLength**(*args*, ***kwargs*)

    Scanning Length TID 10014

    From DICOM Part 16:
        No description

    exception DoesNotExist

    exception MultipleObjectsReturned

class remapp.models.**SizeSpecificDoseEstimation**(*args*, ***kwargs*)

    Container in TID 10013 to hold size specific dose estimation details

    exception DoesNotExist

    exception MultipleObjectsReturned

class remapp.models.**SizeUpload**(*args*, ***kwargs*)

    Table to store patient size information

    exception DoesNotExist

    exception MultipleObjectsReturned

class remapp.models.**SkinDoseMapCalcSettings**(*args*, ***kwargs*)

    Table to store skin dose map calculation settings

    exception DoesNotExist

    exception MultipleObjectsReturned

class remapp.models.**SkinDoseMapResults**(*args*, ***kwargs*)

    Table to hold the results from OpenSkin

    exception DoesNotExist

    exception MultipleObjectsReturned

class remapp.models.**SourceOfCTDoseInformation**(*args*, ***kwargs*)

    Source of CT Dose Information

    exception DoesNotExist

    exception MultipleObjectsReturned

class remapp.models.**StandardNameSettings**(*args*, ***kwargs*)

    Table to store standard name mapping settings

    exception DoesNotExist

    exception MultipleObjectsReturned

class remapp.models.**StandardNames**(*args*, ***kwargs*)

    Table to store standard study description, requested procedure, procedure or acquisition names

    exception DoesNotExist

    exception MultipleObjectsReturned

class remapp.models.**SummaryFields**(**args*, ***kwargs*)

Status and progress of populating the summary fields in GeneralStudyModuleAttr

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**UniqueEquipmentNames**(**args*, ***kwargs*)

Table to unique equipment name information

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**UpgradeStatus**(**args*, ***kwargs*)

Record upgrade status activity

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**UserProfile**(**args*, ***kwargs*)

Table to store user profile settings

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**WEDSeriesOrInstances**(**args*, ***kwargs*)

From TID 10013 Series or Instance used for Water Equivalent Diameter estimation

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**XrayFilters**(**args*, ***kwargs*)

Container in TID 10003b. Code value 113771

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**XrayGrid**(**args*, ***kwargs*)

Content ID 10017 X-Ray Grid

From DICOM Part 16

exception DoesNotExist

exception MultipleObjectsReturned

class remapp.models.**XrayTubeCurrent**(**args*, ***kwargs*)

In TID 10003b. Code value 113734 (mA)

exception DoesNotExist

exception MultipleObjectsReturned

remapp.models.**create_or_save_high_dose_metric_alert_recipient_setting**(*sender*, *instance*,
**kwargs)

> Function to create or save fluoroscopy high dose alert recipient settings

remapp.models.**limit_background_task_table_rows**(*sender*, *instance*, ***kwargs*)

> Method to limit the number of rows in the BackgroundTask table. This method is triggered by a post_save signal associated with the BackgroundTask table.

11.10 Filtering code

class remapp.interface.mod_filters.**CTFilterPlusPid**(**args*, ***kwargs*)

> Adding patient name and ID to filter if permissions allow

class remapp.interface.mod_filters.**CTFilterPlusPidPlusStdNames**(**args*, ***kwargs*)

> Adding standard name fields

class remapp.interface.mod_filters.**CTFilterPlusStdNames**(*data=None*, *queryset=None*, ***,
request=None, *prefix=None*)

> Adding standard name fields

class remapp.interface.mod_filters.**CTSummaryListFilter**(*data=None*, *queryset=None*, ***,
request=None, *prefix=None*)

> Filter for CT studies to display in web interface.
>
> > **class Meta**
> >
> > > Lists fields and order-by information for django-filter filtering
> > >
> > > > **model**
> > > >
> > > > > alias of *GeneralStudyModuleAttr*

class remapp.interface.mod_filters.**DXFilterPlusPid**(**args*, ***kwargs*)

> Adding patient name and ID to filter if permissions allow

class remapp.interface.mod_filters.**DXFilterPlusPidPlusStdNames**(**args*, ***kwargs*)

> Adding standard name fields

class remapp.interface.mod_filters.**DXFilterPlusStdNames**(*data=None*, *queryset=None*, ***,
request=None, *prefix=None*)

> Adding standard name fields

class remapp.interface.mod_filters.**DXSummaryListFilter**(*data=None*, *queryset=None*, ***,
request=None, *prefix=None*)

> Filter for DX studies to display in web interface.
>
> > **class Meta**
> >
> > > Lists fields and order-by information for django-filter filtering
> > >
> > > > **model**
> > > >
> > > > > alias of *GeneralStudyModuleAttr*

class remapp.interface.mod_filters.**DateTimeOrderingFilter**(**args*, ***kwargs*)

> Custom filter to order by date and time as they are two seperate fields

filter(*qs*, *value*)

Sets order_by to date then time and returns queryset

Parameter

- **qs** – queryset

- **value** – list containing ordering type as string

Rückgabe

ordered queryset

class remapp.interface.mod_filters.**MGFilterPlusPid**(**args*, ***kwargs*)

Adding patient name and ID to filter if permissions allow

class remapp.interface.mod_filters.**MGFilterPlusPidPlusStdNames**(**args*, ***kwargs*)

Adding standard name fields

class remapp.interface.mod_filters.**MGFilterPlusStdNames**(*data=None*, *queryset=None*, ***, *request=None*, *prefix=None*)

Adding standard name fields

class remapp.interface.mod_filters.**MGSummaryListFilter**(*data=None*, *queryset=None*, ***, *request=None*, *prefix=None*)

Filter for mammography studies to display in web interface.

class Meta

Lists fields and order-by information for django-filter filtering

model

alias of *GeneralStudyModuleAttr*

class remapp.interface.mod_filters.**NMFilterPlusPid**(**args*, ***kwargs*)

Adding patient name and ID to filter if permissions allow

class remapp.interface.mod_filters.**NMSummaryListFilter**(*data=None*, *queryset=None*, ***, *request=None*, *prefix=None*)

Filter for NM studies to display in web interface.

class Meta

Lists fields and order-by information for django-filter filtering

model

alias of *GeneralStudyModuleAttr*

class remapp.interface.mod_filters.**RFFilterPlusPid**(**args*, ***kwargs*)

Adding patient name and ID to filter if permissions allow

class remapp.interface.mod_filters.**RFFilterPlusPidPlusStdNames**(**args*, ***kwargs*)

Adding standard name fields

class remapp.interface.mod_filters.**RFFilterPlusStdNames**(*data=None*, *queryset=None*, ***, *request=None*, *prefix=None*)

Adding standard name fields

class remapp.interface.mod_filters.**RFSummaryListFilter**(*data=None*, *queryset=None*, ***, *request=None*, *prefix=None*)

Filter for fluoroscopy studies to display in web interface.

class Meta
> Lists fields and order-by information for django-filter filtering

> **model**
>> alias of *GeneralStudyModuleAttr*

remapp.interface.mod_filters.**custom_id_filter**(*queryset*, *name*, *value*)
> Search for ID as plain text and encrypted

remapp.interface.mod_filters.**custom_name_filter**(*queryset*, *name*, *value*)
> Search for name as plain text and encrypted

11.11 Views

remapp.views.**ct_detail_view**(*request*, *pk=None*)
> Detail view for a CT study.

remapp.views.**ct_summary_list_filter**(*request*)
> Obtain data for CT summary view.

remapp.views.**dx_detail_view**(*request*, *pk=None*)
> Detail view for a DX study.

remapp.views.**dx_summary_list_filter**(*request*)
> Obtain data for radiographic summary view.

remapp.views.**logout_page**(*request*)
> Log users out and re-direct them to the main page.

remapp.views.**mg_detail_view**(*request*, *pk=None*)
> Detail view for a CT study.

remapp.views.**mg_summary_list_filter**(*request*)
> Mammography data for summary view.

remapp.views.**multiply**(*value*, *arg*)
> Return multiplication within Django templates

> **Parameter**
>> - **value** – the value to multiply
>> - **arg** – the second value to multiply

> **Rückgabe**
>> the multiplication

remapp.views.**nm_detail_view**(*request*, *pk=None*)
> Detail view for a NM study.

remapp.views.**nm_summary_list_filter**(*request*)
> Obtain data for NM summary view.

remapp.views.**rf_detail_view**(*request*, *pk=None*)
> Detail view for an RF study.

remapp.views.**rf_detail_view_skin_map**(*request*, *pk=None*)

>   View to calculate a skin dose map. Currently just a copy of rf_detail_view.

remapp.views.**rf_summary_list_filter**(*request*)

>   Obtain data for radiographic summary view.

remapp.views.**standard_name_settings**()

>   Obtain the system-level enable_standard_names setting.

remapp.views.**update_latest_studies**(*request*)

>   AJAX function to calculate the latest studies for each display name for a particular modality.

>   >   **Parameter**
>   >   >   **request** – Request object

>   >   **Rückgabe**
>   >   >   HTML table of modalities

11.11.1 openSkin related views

openSkin related views.

class remapp.views_openskin.**SkinDoseMapCalcSettingsUpdate**(***kwargs*)

>   Update skin dose map calculation settings.

>   **form_class**

>   >   alias of *SkinDoseMapCalcSettingsForm*

>   **form_valid**(*form*)

>   >   If the form is valid, save the associated model.

>   **get_context_data**(***context*)

>   >   Insert the form into the context dict.

>   **model**

>   >   alias of *SkinDoseMapCalcSettings*

class remapp.views_openskin.**SkinSafeListCreate**(***kwargs*)

>   Enable skin map calculations by adding model, or model and software version to *OpenSkinSafeList*.

>   **form_class**

>   >   alias of *SkinSafeListForm*

>   **form_valid**(*form*)

>   >   If the form is valid, save the associated model.

>   **get_context_data**(***context*)

>   >   Insert the form into the context dict.

>   **model**

>   >   alias of *OpenSkinSafeList*

class remapp.views_openskin.**SkinSafeListDelete**(***kwargs*)

>   Disable skin map calculations for particular model or model and software version.

>   **delete**(*request*, **args*, ***kwargs*)

>   >   Call the delete() method on the fetched object and then redirect to the success URL.

get_context_data(***context*)

Insert the single object into the context dict.

model

alias of *OpenSkinSafeList*

class remapp.views_openskin.**SkinSafeListUpdate**(***kwargs*)

Add or remove the software version restriction.

form_class

alias of *SkinSafeListForm*

form_valid(*form*)

If the form is valid, save the associated model.

get_context_data(***context*)

Insert the form into the context dict.

model

alias of *OpenSkinSafeList*

remapp.views_openskin.**check_skin_safe_model**(*skin_safe_models*)

Check if device matches on manufacturer and model without version restriction.

openSkin safe list *OpenSkinSafeList* is checked against manufacturer and model. This function is then used to check if there are any entries on the list where *software_version* is blank.

Parameter

skin_safe_models (`OpenSkinSafeList queryset`) – Queryset of safe list entries matching manufacturer and model

Rückgabe

- **safe_list_model_pk** (*int or None*) – Primary key of entry if match found, `None` otherwise

- **model_enabled** (*bool*) – True if match found with blank *software_version*, otherwise `False`

remapp.views_openskin.**display_name_skin_enabled**(*request*)

AJAX view to display if skin map calculations are enabled and links to change the configuration.

remapp.views_openskin.**get_matching_equipment_names**(*manufacturer*, *model_name*)

Get queryset of unique equipment names that match the manufacturer and model name being reviewed.

Filters the *UniqueEquipmentNames* table for fluoroscopy entries (or dual fluoro + radiography) that match the manufacturer and model that has been selected.

Parameter

- **manufacturer** (`str`) – Name of manufacturer from *UniqueEquipmentNames* table

- **model_name** (`str`) – Model name from *UniqueEquipmentNames* table

Rückgabe

Queryset filtered for fluoro systems matching the manufacturer and model name

Rückgabetyp

UniqueEquipmentNames queryset

11.12 Export Views

remapp.exports.exportviews.**ct_xlsx_phe2019**(*request*)

> View to launch task to export CT studies to xlsx file in PHE 2019 CT survey format
>
> > **Parameter**
> > > **request** – Contains the database filtering parameters and user details.

remapp.exports.exportviews.**ctcsv1**(*request*, *name=None*, *pat_id=None*)

> View to launch task to export CT studies to csv file
>
> > **Parameter**
> > > - **request** (*GET*) – Contains the database filtering parameters. Also used to get user group.
> > > - **name** – string, 0 or 1 from URL indicating if names should be exported
> > > - **pat_id** – string, 0 or 1 from URL indicating if patient ID should be exported

remapp.exports.exportviews.**ctxlsx1**(*request*, *name=None*, *pat_id=None*)

> View to launch task to export CT studies to xlsx file
>
> > **Parameter**
> > > - **request** (*GET*) – Contains the database filtering parameters. Also used to get user group.
> > > - **name** – string, 0 or 1 from URL indicating if names should be exported
> > > - **pat_id** – string, 0 or 1 from URL indicating if patient ID should be exported

remapp.exports.exportviews.**deletefile**(*request*)

> View to delete export files from the server
>
> > **Parameter**
> > > **request** (*POST*) – Contains the task ID

remapp.exports.exportviews.**download**(*request*, *task_id*)

> View to handle downloads of files from the server
>
> Originally used for download of the export spreadsheets, now also used for downloading the patient size import logfiles.
>
> > **Parameter**
> > > - **request** – Used to get user group.
> > > - **task_id** – ID of the export or logfile

remapp.exports.exportviews.**dx_xlsx_phe2019**(*request*, *export_type=None*)

> View to launch task to export DX studies to xlsx file in PHE 2019 DX survey format
>
> > **Parameter**
> > > - **request** – Contains the database filtering parameters and user details.
> > > - **export_type** – string, ‚projection' or ‚exam'

remapp.exports.exportviews.**dxcsv1**(*request*, *name=None*, *pat_id=None*)

> View to launch task to export DX and CR studies to csv file
>
> > **Parameter**
> > > - **request** (*GET*) – Contains the database filtering parameters. Also used to get user group.
> > > - **name** – string, 0 or 1 from URL indicating if names should be exported

> • `pat_id` – string, 0 or 1 from URL indicating if patient ID should be exported

remapp.exports.exportviews.**dxxlsx1**(*request*, *name=None*, *pat_id=None*)

> View to launch task to export DX and CR studies to xlsx file
>
> > **Parameter**
> >
> > > • `request` (*GET*) – Contains the database filtering parameters. Also used to get user group.
> > >
> > > • `name` – string, 0 or 1 from URL indicating if names should be exported
> > >
> > > • `pat_id` – string, 0 or 1 from URL indicating if patient ID should be exported

remapp.exports.exportviews.**export**(*request*)

> View to list current and completed exports to track progress, download and delete
>
> > **Parameter**
> >
> > > `request` – Used to get user group.

remapp.exports.exportviews.**export_abort**(*request*, *pk*)

> View to abort current export job
>
> > **Parameter**
> >
> > > `request` (*POST*) – Contains the task primary key

remapp.exports.exportviews.**export_remove**(*request*, *task_id=None*)

> Function to remove export task from queue
>
> > **Parameter**
> >
> > > • `request` (*POST*) – Contains the task primary key
> > >
> > > • `task_id` – UUID of task in question

remapp.exports.exportviews.**flcsv1**(*request*, *name=None*, *pat_id=None*)

> View to launch task to export fluoroscopy studies to csv file
>
> > **Parameter**
> >
> > > • `request` (*GET*) – Contains the database filtering parameters. Also used to get user group.
> > >
> > > • `name` – string, 0 or 1 from URL indicating if names should be exported
> > >
> > > • `pat_id` – string, 0 or 1 from URL indicating if patient ID should be exported

remapp.exports.exportviews.**include_pid**(*request*, *name*, *pat_id*)

> Check if user is allowed to export PID, then check if they have asked to. :param request: request so we can determine the user and therefore groups :param name: string, 0 or 1 from URL indicating if names should be exported :param pat_id: string, 0 or 1 from URL indicating if patient ID should be exported :return: dict, with pidgroup, include_names and include_pat_id as bools

remapp.exports.exportviews.**mgcsv1**(*request*, *name=None*, *pat_id=None*)

> Launches export of mammo data to CSV :param request: Contains the database filtering parameters. Also used to get user group. :param name: string, 0 or 1 from URL indicating if names should be exported :param pat_id: string, 0 or 1 from URL indicating if patient ID should be exported :return:

remapp.exports.exportviews.**mgnhsbsp**(*request*)

> View to launch task to export mammography studies to csv file using a NHSBSP template
>
> > **Parameter**
> >
> > > `request` (*GET*) – Contains the database filtering parameters. Also used to get user group.

remapp.exports.exportviews.**mgxlsx1**(*request*, *name=None*, *pat_id=None*)

>   Launches export of mammo data to xlsx :param request: Contains the database filtering parameters. Also used to get user group. :param name: string, 0 or 1 from URL indicating if names should be exported :param pat_id: string, 0 or 1 from URL indicating if patient ID should be exported :return:

remapp.exports.exportviews.**nmcsv1**(*request*, *name=None*, *pat_id=None*)

>   View to launch task to export NM studies to csv file

>   >   **Parameter**

>   >   >   - **request** (*GET*) – Contains the database filtering parameters. Also used to get user group.

>   >   >   - **name** – string, 0 or 1 from URL indicating if names should be exported

>   >   >   - **pat_id** – string, 0 or 1 from URL indicating if patient ID should be exported

remapp.exports.exportviews.**nmxlsx1**(*request*, *name=None*, *pat_id=None*)

>   View to launch celery task to export NM studies to excel file

>   >   **Parameter**

>   >   >   - **request** (*GET*) – Contains the database filtering parameters. Also used to get user group.

>   >   >   - **name** – string, 0 or 1 from URL indicating if names should be exported

>   >   >   - **pat_id** – string, 0 or 1 from URL indicating if patient ID should be exported

remapp.exports.exportviews.**rf_xlsx_phe2019**(*request*)

>   View to launch task to export fluoro studies to xlsx file in PHE 2019 IR/fluoro survey format

>   >   **Parameter**

>   >   request – Contains the database filtering parameters and user details.

remapp.exports.exportviews.**rfopenskin**(*request*, *pk*)

>   Create csv export suitable for import to standalone openSkin :param request: request object :param pk: primary key of study in GeneralStudyModuleAttr table

remapp.exports.exportviews.**rfxlsx1**(*request*, *name=None*, *pat_id=None*)

>   View to launch task to export fluoroscopy studies to xlsx file

>   >   **Parameter**

>   >   >   - **request** (*GET*) – Contains the database filtering parameters. Also used to get user group.

>   >   >   - **name** – string, 0 or 1 from URL indicating if names should be exported

>   >   >   - **pat_id** – string, 0 or 1 from URL indicating if patient ID should be exported

remapp.exports.exportviews.**update_active**(*request*)

>   AJAX function to return active exports

>   >   **Parameter**

>   >   request – Request object

>   >   **Rückgabe**

>   >   HTML table of active exports

remapp.exports.exportviews.**update_complete**(*request*)

>   AJAX function to return recently completed exports

>   >   **Parameter**

>   >   request – Request object, including pk of latest complete export at initial page load

> **Rückgabe**
> HTML table of completed exports

remapp.exports.exportviews.**update_error**(*request*)

> AJAX function to return exports in error state
>
> > **Parameter**
> > **request** – Request object
> >
> > **Rückgabe**
> > HTML table of exports in error state

remapp.exports.exportviews.**update_queue**(*request*)

> AJAX function to return queued exports
>
> > **Parameter**
> > **request** – Request object
> >
> > **Rückgabe**
> > HTML table of active exports

11.13 Forms

class remapp.forms.**BackgroundTaskMaximumRowsForm**(**args*, ***kwargs*)

> Form for configuring the maximum number of rows in the BackgroundTask table
>
> **property media**
> > Return all media required to render the widgets on this form.

class remapp.forms.**CTChartOptionsDisplayForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

> Form for CT chart display options
>
> **property media**
> > Return all media required to render the widgets on this form.

class remapp.forms.**CTChartOptionsDisplayFormIncStandard**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

> **property media**
> > Return all media required to render the widgets on this form.

class remapp.forms.**CTChartOptionsForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

> Form for CT chart options

property media

> Return all media required to render the widgets on this form.

class remapp.forms.**CTChartOptionsFormIncStandard**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

property media

> Return all media required to render the widgets on this form.

class remapp.forms.**DXChartOptionsDisplayForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Form for DX chart display options

property media

> Return all media required to render the widgets on this form.

class remapp.forms.**DXChartOptionsDisplayFormIncStandard**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

property media

> Return all media required to render the widgets on this form.

class remapp.forms.**DXChartOptionsForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Form for DX chart options

property media

> Return all media required to render the widgets on this form.

class remapp.forms.**DXChartOptionsFormIncStandard**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

property media

> Return all media required to render the widgets on this form.

class remapp.forms.**DicomDeleteSettingsForm**(**args*, ***kwargs*)

Form for configuring whether DICOM objects are stored or deleted once processed

property media

Return all media required to render the widgets on this form.

class remapp.forms.**DicomQRForm**(**args*, ***kwargs*)

Form for configuring remote Query Retrieve nodes

property media

Return all media required to render the widgets on this form.

class remapp.forms.**DicomQueryForm**(**args*, ***kwargs*)

Form for launching DICOM Query

clean()

Validate the form data to clear modality selections if sr_only is selected. :return: Form with modalities _or_ sr_only selected

class date

date(year, month, day) –> date object

ctime()

Return ctime() style string.

fromisocalendar()

int, int, int -> Construct a date from the ISO year, week number and weekday.

This is the inverse of the date.isocalendar() function

fromisoformat()

str -> Construct a date from the output of date.isoformat()

fromordinal()

int -> date corresponding to a proleptic Gregorian ordinal.

fromtimestamp()

Create a date from a POSIX timestamp.

The timestamp is a number, e.g. created via time.time(), that is interpreted as local time.

isocalendar()

Return a 3-tuple containing ISO year, week number, and weekday.

isoformat()

Return string in ISO 8601 format, YYYY-MM-DD.

isoweekday()

Return the day of the week represented by the date. Monday == 1 … Sunday == 7

replace()

Return date with new specified fields.

strftime()

format -> strftime() style string.

timetuple()

Return time tuple, compatible with time.localtime().

today()

Current date or datetime: same as self.__class__.fromtimestamp(time.time()).

> > **toordinal()**
> > > Return proleptic Gregorian ordinal. January 1 of year 1 is day 1.
> >
> > **weekday()**
> > > Return the day of the week represented by the date. Monday == 0 … Sunday == 6
>
> **property media**
> > Return all media required to render the widgets on this form.

class remapp.forms.**DicomStoreForm**(**args*, ***kwargs*)
> Form for configuring local Store nodes
>
> **property media**
> > Return all media required to render the widgets on this form.

class remapp.forms.**GeneralChartOptionsDisplayForm**(*data=None*, *files=None*, *auto_id='id_%s'*,
> > > > > *prefix=None*, *initial=None*, *error_class=<class*
> > > > > *'django.forms.utils.ErrorList'>*, *label_suffix=None*,
> > > > > *empty_permitted=False*, *field_order=None*,
> > > > > *use_required_attribute=None*, *renderer=None*)
>
> Form for general chart display options
>
> **property media**
> > Return all media required to render the widgets on this form.

class remapp.forms.**HomepageOptionsForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*,
> > > > *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*,
> > > > *label_suffix=None*, *empty_permitted=False*, *field_order=None*,
> > > > *use_required_attribute=None*, *renderer=None*)
>
> Form for displaying and changing the home page options
>
> **property media**
> > Return all media required to render the widgets on this form.

class remapp.forms.**MGChartOptionsDisplayForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*,
> > > > *initial=None*, *error_class=<class*
> > > > *'django.forms.utils.ErrorList'>*, *label_suffix=None*,
> > > > *empty_permitted=False*, *field_order=None*,
> > > > *use_required_attribute=None*, *renderer=None*)
>
> Form for MG chart display options
>
> **property media**
> > Return all media required to render the widgets on this form.

class remapp.forms.**MGChartOptionsDisplayFormIncStandard**(*data=None*, *files=None*, *auto_id='id_%s'*,
> > > > > *prefix=None*, *initial=None*,
> > > > > *error_class=<class*
> > > > > *'django.forms.utils.ErrorList'>*,
> > > > > *label_suffix=None*,
> > > > > *empty_permitted=False*, *field_order=None*,
> > > > > *use_required_attribute=None*,
> > > > > *renderer=None*)
>
> **property media**
> > Return all media required to render the widgets on this form.

class remapp.forms.**MGChartOptionsForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

    Form for MG chart options

    property media

        Return all media required to render the widgets on this form.

class remapp.forms.**MGChartOptionsFormIncStandard**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

    property media

        Return all media required to render the widgets on this form.

class remapp.forms.**MergeOnDeviceObserverUIDForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

    Form for displaying and changing the option for merging on Device Observer UID

    property media

        Return all media required to render the widgets on this form.

class remapp.forms.**NMChartOptionsDisplayForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

    Form for NM chart display options

    property media

        Return all media required to render the widgets on this form.

class remapp.forms.**NMChartOptionsForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

    Form for NM chart options

    property media

        Return all media required to render the widgets on this form.

class remapp.forms.**NotPatientIDForm**(**args*, ***kwargs*)

    Form for configuring not-patient ID patterns

    property media

        Return all media required to render the widgets on this form.

class remapp.forms.**NotPatientNameForm**(**args*, ***kwargs*)

    Form for configuring not-patient name patterns

property media

        Return all media required to render the widgets on this form.

class remapp.forms.**RFChartOptionsDisplayForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Form for RF chart display options

property media

        Return all media required to render the widgets on this form.

class remapp.forms.**RFChartOptionsDisplayFormIncStandard**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

property media

        Return all media required to render the widgets on this form.

class remapp.forms.**RFChartOptionsForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Form for RF chart options

property media

        Return all media required to render the widgets on this form.

class remapp.forms.**RFChartOptionsFormIncStandard**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

property media

        Return all media required to render the widgets on this form.

class remapp.forms.**RFHighDoseFluoroAlertsForm**(**args*, ***kwargs*)

Form for displaying and changing fluoroscopy high dose alert settings

property media

        Return all media required to render the widgets on this form.

class remapp.forms.**SizeHeadersForm**(*my_choice=None*, ***kwargs*)

Form for csv column header patient size imports through the web interface

property media

        Return all media required to render the widgets on this form.

---

class remapp.forms.**SizeUploadForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Form for patient size csv file upload

property media

Return all media required to render the widgets on this form.

class remapp.forms.**SkinDoseMapCalcSettingsForm**(**args*, ***kwargs*)

Form for configuring whether skin dose maps are shown / calculated

property media

Return all media required to render the widgets on this form.

class remapp.forms.**SkinSafeListForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

Form for adding/updating/removing system from openSkin safe list

property media

Return all media required to render the widgets on this form.

class remapp.forms.**StandardNameFormBase**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

For configuring standard names for study description, requested procedure, procedure and acquisition name.

property media

Return all media required to render the widgets on this form.

class remapp.forms.**StandardNameFormCT**(**args*, ***kwargs*)

Form for configuring standard names for study description, requested procedure, procedure and acquisition name

property media

Return all media required to render the widgets on this form.

class remapp.forms.**StandardNameFormDX**(**args*, ***kwargs*)

Form for configuring standard names for study description, requested procedure, procedure and acquisition name

property media

Return all media required to render the widgets on this form.

class remapp.forms.**StandardNameFormMG**(**args*, ***kwargs*)

Form for configuring standard names for study description, requested procedure, procedure and acquisition name

property media

Return all media required to render the widgets on this form.

class remapp.forms.**StandardNameFormRF**(**args*, ***kwargs*)

    Form for configuring standard names for study description, requested procedure, procedure and acquisition name

    property media

        Return all media required to render the widgets on this form.

class remapp.forms.**StandardNameSettingsForm**(**args*, ***kwargs*)

    Form for configuring whether standard names are shown / used

    property media

        Return all media required to render the widgets on this form.

class remapp.forms.**UpdateDisplayNamesForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

    property media

        Return all media required to render the widgets on this form.

class remapp.forms.**itemsPerPageForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

    property media

        Return all media required to render the widgets on this form.

11.14 Charts

remapp.interface.chart_functions.**calc_facet_rows_and_height**(*df*, *facet_col_name*, *facet_col_wrap*)

    Calculates the required total chart height and the number of facet rows. Each row has a hard-coded height of 500 pixels.

    Parameter

        • **df** – Pandas DataFrame containing the data

        • **facet_col_name** – string containing the DataFrame column name containing the facet names

        • **facet_col_wrap** – int representing the number of subplots to have on each row

    Rückgabe

        two-element list containing the chart height in pixels (int) and the number of facet rows (int)

remapp.interface.chart_functions.**calc_histogram_bin_data**(*df*, *value_col_name*, *n_bins=10*)

    Calculates histogram bin label text, bin boundaries and bin mid-points

    Parameter

        • **df** – the Pandas DataFrame containing the data

        • **value_col_name** – (string)name of the DataFrame column that contains the values

        • **n_bins** – (int) the number of bins to use

Rückgabe

a three element list containing the bin labels, bin boundaries and bin mid-points

remapp.interface.chart_functions.**calculate_colour_sequence**(*scale_name='RdYlBu'*, *n_colours=10*)

Calculates a sequence of n_colours from the matplotlib colourmap scale_name

Parameter

- **scale_name** – string containing the name of the matplotlib colour scale to use

- **n_colours** – int representing the number of colours required

Rückgabe

list of hexadecimal colours from a matplotlib colormap

remapp.interface.chart_functions.**construct_over_time_charts**(*df*, *params*,
group_by_physician=None)

Construct a Plotly line chart of average values over time, optionally grouped by performing physician name. For „boxplot" a plotly boxplot of values over time is returned instead of an plotly line chart.

Parameter

- **df** – the Pandas DataFrame containing the data

- **params** – a dictionary of processing parameters

- **params["df_name_col"]** – (string) DataFrame column containing categories

- **params["name_title"]** – (string) name title

- **params["df_value_col"]** – (string) DataFrame column containing values

- **params["value_title"]** – (string) y-axis title

- **params["df_date_col"]** – (string) DataFrame column containing dates

- **params["date_title"]** – (string) date title

- **params["facet_title"]** – (string) subplot title

- **params["sorting_choice"]** – 2-element list. [0] sets sort direction, [1] used to determine which field to sort

- **params["average_choices"]** – list of strings containing required averages („mean", „median", „boxplot")

- **params["time_period"]** – string containing the time period to average over; „A" (years), „Q" (quarters), „M" (months), „W" (weeks), „D" (days)

- **params["grouping_choice"]** – (string) „series" or „system"

- **params["colourmap"]** – (string) colourmap to use

- **params["filename"]** – (string) default filename to use for plot bitmap export

- **params["facet_col_wrap"]** – (int) number of subplots per row

- **params["return_as_dict"]** – (boolean) flag to trigger return as a dictionary rather than a HTML DIV

- **group_by_physician** – boolean flag to set whether to group by physician name

Rückgabe

a dictionary containing a combination of [„mean"], [„median"] and [„boxplot"] entries, each of which contains a Plotly figure embedded in an HTML DIV; or Plotly figure as a dictionary (if

params[„return_as_dict"] is True); or an error message embedded in an HTML DIV if there was a ValueError when calculating the figure

remapp.interface.chart_functions.**create_dataframe**(*database_events*, *field_dict*, *data_point_name_lowercase=None*, *data_point_name_remove_whitespace_padding=None*, *data_point_value_multipliers=None*, *char_wrap=500*, *uid=None*)

Creates a Pandas DataFrame from the supplied database records. names fields are made categorical to save system memory Any missing (na) values in names fields are set to Blank

> **Parameter**
>
> - **database_events** – the database events
>
> - **field_dict** – a dictionary of lists, each containing database field names to include in the DataFrame. The dictionary should include „names", „values", „dates", „times" and optionally „system" items
>
> - **data_point_name_lowercase** – boolean flag to determine whether to make all „names" field values lower case
>
> - **data_point_name_remove_whitespace_padding** – boolean flag to determine whether to strip whitespace
>
> - **data_point_value_multipliers** – list of float valuse to multiply each „values" field value by
>
> - **char_wrap** – the maximum length of series and system names before characters are wrapped with
>
> - **uid** – string containing database field name which contains a unique identifier for each record
>
> **Rückgabe**
> a Pandas DataFrame with a column per required field

remapp.interface.chart_functions.**create_dataframe_aggregates**(*df*, *df_name_cols*, *df_agg_col*, *stats_to_use=None*)

Creates a Pandas DataFrame with the specified statistics (mean, median, count, for example) grouped by x-ray system name and by the list of provided df_name_cols.

> **Parameter**
>
> - **df** – Pandas DataFrame containing the raw data; it must have an „x_ray_system_name" column
>
> - **df_name_cols** – list of strings representing the DataFrame column names to group by
>
> - **df_agg_col** – string containing the DataFrame column over which to calculate the statistics
>
> - **stats_to_use** – list of strings containing the statistics to calculate, such as „mean", „median", „count"
>
> **Rückgabe**
> Pandas DataFrame containing the grouped aggregate data

remapp.interface.chart_functions.**create_dataframe_time_series**(*df*, *df_name_col*, *df_value_col*, *df_date_col='study_date'*, *time_period='M'*, *average_choices=None*, *group_by_physician=None*)

Creates a Pandas DataFrame time series of average values grouped by x_ray_system_name and df_name_col

Parameter

- **df** – the Pandas DataFrame containing the raw data
- **df_name_col** – string containing the DataFrame columnn name used to group the data
- **df_value_col** – string containing the DataFrame column containing the values to be averaged
- **df_date_col** – string containing the DataFrame column containing the dates
- **time_period** – string containing the time period to average over; „A" (years), „Q" (quarters), „M" (months), „W" (weeks), „D" (days)
- **average_choices** – list of strings containing one or both of „mean" and „median"
- **group_by_physician** – boolean flag to set whether to group by physician

Rückgabe
Pandas DataFrame containing the time series of average values grouped by system and name

remapp.interface.chart_functions.**create_dataframe_weekdays**(*df*, *df_name_col*,
df_date_col='study_date')

Creates a Pandas DataFrame of the number of events in each day of the week, and in hour of that day.

Parameter

- **df** – Pandas DataFrame containing the raw data; it must have a „study_time" and „x_ray_system_name" column
- **df_name_col** – string containing the df column name to group the results by
- **df_date_col** – string containing the df column name containing dates

Rückgabe
Pandas DataFrame containing the number of studies per day and hour grouped by name

remapp.interface.chart_functions.**create_freq_sorted_category_list**(*df*, *df_name_col*, *sorting*)

Create a sorted list of categories for frequency charts. Makes use of Pandas DataFrame sort_values (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.sort_values.html).

sorting[0] sets sort direction

sorting[1] used to determine field to sort on: „name" sorts by df_name_col; otherwise sorted by „x_ray_system_name"

Parameter

- **df** – Pandas DataFrame containing the data
- **df_name_col** – DataFrame column containing the category names
- **sorting** – 2-element list. [0] sets sort direction, [1] used to determine which field to sort on

Rückgabe
dictionary with key df_name_col and a list of sorted categories as the value

remapp.interface.chart_functions.**create_sorted_category_list**(*df*, *df_name_col*, *df_value_col*,
sorting)

Create a sorted list of categories for scatter and over-time charts. The data is grouped by df_name_col and the mean and count calculated for each. The grouped DataFrame is then sorted according to the provided sorting. Makes use of Pandas DataFrame sort_values (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.sort_values.html).

sorting[0] sets sort direction

sorting[1] used to determine sort order: „name" sorts by df_name_col; otherwise sorted by „x_ray_system_name"

> **Parameter**
>> • **df** – Pandas DataFrame containing the data
>>
>> • **df_name_col** – DataFrame column containing the category names. Used to group the data
>>
>> • **df_value_col** – DataFrame column containing values to count and calculate the mean
>>
>> • **sorting** – 2-element list. [0] sets sort direction, [1] used to determine which field to sort on
>
> **Rückgabe**
>> dictionary with key df_name_col and a list of sorted categories as the value

remapp.interface.chart_functions.**csv_data_barchart**(*fig*, *params*)

> Calculates a Pandas DataFrame containing chart data to be used for csv download
>
> **Parameter**
>> • **fig** – Plotly figure containing the data to extract
>>
>> • **params** – a dictionary of parameters
>>
>> • **params["df_name_col"]** – (string) DataFrame column containing categories
>>
>> • **params["name_axis_title"]** – (string) title for the name data
>>
>> • **params["value_axis_title"]** – (string) title for the value data
>>
>> • **params["facet_col"]** – (string) DataFrame column used to split data into subgroups
>
> **Rückgabe**
>> DataFrame containing the data for download

remapp.interface.chart_functions.**csv_data_frequency**(*fig*, *params*)

> Calculates a Pandas DataFrame containing chart data to be used for csv download
>
> **Parameter**
>> • **fig** – Plotly figure containing the data to extract
>>
>> • **params** – a dictionary of parameters; must include „x_axis_title"
>
> **Rückgabe**
>> DataFrame containing the data for download

remapp.interface.chart_functions.**download_link**(*object_to_download*, *download_filename*, *download_link_text='Download csv'*)

> Adapted from: https://discuss.streamlit.io/t/heres-a-download-function-that-works-for-dataframes-and-txt/4052
>
> Generates a link to download the given object_to_download.
>
> object_to_download (str, pd.DataFrame): The object to be downloaded. download_filename (str): filename and extension of file. e.g. mydata.csv, some_txt_output.txt download_link_text (str): Text to display for download link.
>
> Examples:
>
> download_link(YOUR_DF, 'YOUR_DF.csv', 'Click here to download data!')
>
> download_link(YOUR_STRING, 'YOUR_STRING.txt', 'Click here to download your text!')

remapp.interface.chart_functions.**empty_dataframe_msg**(*params=None*)

> Returns a string containing an HTML DIV with a message warning that the DataFrame is empty
>
> > **Parameter**
> > > **params** – parameters which may contain a custom_msg_line
> >
> > **Rückgabe**
> > > string containing an html div with the empty DataFrame message

remapp.interface.chart_functions.**failed_chart_message_div**(*custom_msg_line*, *e*)

> Returns a string containing an HTML DIV with a failed chart message
>
> > **Parameter**
> > > - **custom_msg_line** – string containing a custom line to add to the message
> > > - **e** – Python error object
> >
> > **Rückgabe**
> > > string containing the message in an HTML DIV

remapp.interface.chart_functions.**global_config**(*filename*, *height_multiplier=1.0*, *height=1080*, *width=1920*)

> Creates a Plotly global configuration dictionary. The parameters all relate to the chart bitmap that can be saved by the user.
>
> > **Parameter**
> > > - **filename** – string containing the file name to use if the user saves the chart as a graphic file
> > > - **height_multiplier** – floating point value used to scale the chart height
> > > - **height** – int value for the height of the chart graphic file
> > > - **width** – int value for the width of the chart graphic file
> >
> > **Rückgabe**
> > > a dictionary of Plotly options

remapp.interface.chart_functions.**plotly_barchart**(*df*, *params*, *csv_name='OpenREM chart data.csv'*)

> Create a plotly bar chart
>
> > **Parameter**
> > > - **df** – Pandas DataFrame containing the data
> > > - **params** – a dictionary of parameters
> > > - **params["average_choice"]** – (string) DataFrame column containing values („mean" or „median")
> > > - **params["value_axis_title"]** – (string) y-axis title
> > > - **params["df_name_col"]** – (string) DataFrame column containing categories
> > > - **params["name_axis_title"]** – (string) x-axis title
> > > - **params["facet_col"]** – (string) DataFrame column used to create subplots
> > > - **params["facet_col_wrap"]** – (int) number of subplots per row
> > > - **params["sorting_choice"]** – 2-element list. [0] sets sort direction, [1] used to determine which field to sort
> > > - **params["colourmap"]** – (string) colourmap to use

- **params["return_as_dict"]** – (boolean) flag to trigger return as a dictionary rather than a HTML DIV

- **params["filename"]** – (string) default filename to use for plot bitmap export

- **csv_name** – (string) default filename to use for plot csv export

Rückgabe

Plotly figure embedded in an HTML DIV; or Plotly figure as a dictionary (if params[„return_as_dict"] is True); or an error message embedded in an HTML DIV if there was a ValueError when calculating the figure

remapp.interface.chart_functions.**plotly_barchart_weekdays**(*df*, *df_name_col*, *df_value_col*, *name_axis_title=''*, *value_axis_title=''*, *colourmap='RdYlBu'*, *filename='OpenREM_workload_chart'*, *facet_col_wrap=3*, *sorting_choice=None*, *return_as_dict=False*)

Create a plotly bar chart of event workload

Parameter

- **df** – Pandas DataFrame containing the data

- **df_name_col** – (string) DataFrame column containing categories

- **df_value_col** – (string) DataFrame column containing values

- **name_axis_title** – (string) x-axis title

- **value_axis_title** – (string) y-axis title

- **colourmap** – (string) colourmap to use

- **filename** – (string) default filename to use for plot bitmap export

- **facet_col_wrap** – (int) number of subplots per row

- **sorting_choice** – 2-element list. [0] sets sort direction, [1] used to determine which field to sort

- **return_as_dict** – (boolean) flag to trigger return as a dictionary rather than a HTML DIV

Rückgabe

Plotly figure embedded in an HTML DIV; or Plotly figure as a dictionary (if „return_as_dict" is True); or an error message embedded in an HTML DIV if there was a ValueError when calculating the figure

remapp.interface.chart_functions.**plotly_binned_statistic_barchart**(*df*, *params*)

Create a plotly binned statistic bar chart

Parameter

- **df** – Pandas DataFrame containing the data

- **params** – a dictionary of parameters

- **params["df_category_col"]** – (string) DataFrame column containing categories

- **params["df_facet_col"]** – (string) DataFrame column used to create subplots

- **params["facet_title"]** – (string) Subplot title

- **params["facet_col_wrap"]** – (int) number of subplots per row

- `params["user_bins"]` – list of ints containing bin edges for binning

- `params["df_category_col"]` – (string) DataFrame column containing categories

- `params["df_x_value_col"]` – (string) DataFrame column containing x data

- `params["df_y_value_col"]` – (string) DataFrame column containing y data

- `params["x_axis_title"]` – (string) Title for x-axis

- `params["y_axis_title"]` – (string) Title for y-axis

- `params["stat_name"]` – (string) „mean" or „median"

- `params["sorting_choice"]` – 2-element list. [0] sets sort direction, [1] used to determine which field to sort

- `params["colourmap"]` – (string) colourmap to use

- `params["return_as_dict"]` – (boolean) flag to trigger return as a dictionary rather than a HTML DIV

- `params["filename"]` – (string) default filename to use for plot bitmap export

Rückgabe

Plotly figure embedded in an HTML DIV; or Plotly figure as a dictionary (if params[„return_as_dict"] is True); or an error message embedded in an HTML DIV if there was a ValueError when calculating the figure

`remapp.interface.chart_functions.`**`plotly_boxplot`**(*df*, *params*)

Produce a plotly boxplot

Parameter

- `df` – Pandas DataFrame containing the data

- `params` – a dictionary of parameters

- `params["df_value_col"]` – (string) DataFrame column containing values

- `params["value_axis_title"]` – (string) x-axis title

- `params["df_name_col"]` – (string) DataFrame column containing categories

- `params["name_axis_title"]` – (string) y-axis title

- `params["df_facet_col"]` – (string) DataFrame column used to create subplots

- `params["df_facet_col_wrap"]` – (int) number of subplots per row

- `params["sorting_choice"]` – 2-element list. [0] sets sort direction, [1] used to determine which field to sort

- `params["colourmap"]` – (string) colourmap to use

- `params["return_as_dict"]` – (boolean) flag to trigger return as a dictionary rather than a HTML DIV

Rückgabe

Plotly figure embedded in an HTML DIV; or Plotly figure as a dictionary (if params[„return_as_dict"] is True); or an error message embedded in an HTML DIV if there was a ValueError when calculating the figure

`remapp.interface.chart_functions.`**`plotly_frequency_barchart`**(*df*, *params*, *csv_name='OpenREM chart data.csv'*)

Create a plotly bar chart of event frequency

Parameter

- **df** – Pandas DataFrame containing the data
- **params** – a dictionary of parameters
- **params["df_x_axis_col"]** – (string) DataFrame column containing categories
- **params["x_axis_title"]** – (string) x-axis title
- **params["groupby_cols"]** – list of strings with DataFrame columns to group data by
- **params["grouping_choice"]** – (string) „series" or „system"
- **params["sorting_choice"]** – 2-element list. [0] sets sort direction, [1] used to determine which field to sort
- **params["legend_title"]** – (string) legend title
- **params["facet_col"]** – (string) DataFrame column used to create subplots
- **params["facet_col_wrap"]** – (int) number of subplots per row
- **params["return_as_dict"]** – (boolean) flag to trigger return as a dictionary rather than a HTML DIV
- **params["colourmap"]** – (string) colourmap to use
- **params["filename"]** – (string) default filename to use for plot bitmap export
- **csv_name** – (string) default filename to use for plot csv export

Rückgabe

Plotly figure embedded in an HTML DIV; or Plotly figure as a dictionary (if „return_as_dict" is True); or an error message embedded in an HTML DIV if there was a ValueError when calculating the figure

remapp.interface.chart_functions.**plotly_histogram_barchart**(*df*, *params*)

Create a plotly histogram bar chart

Parameter

- **df** – Pandas DataFrame containing the data
- **params** – a dictionary of parameters
- **params["df_value_col"]** – (string) DataFrame column containing values
- **params["value_axis_title"]** – (string) y-axis title
- **params["df_facet_col"]** – (string) DataFrame column used to create subplots
- **params["df_category_name_list"]** – string list of each category name
- **params["df_facet_col_wrap"]** – (int) number of subplots per row
- **params["n_bins"]** – (int) number of hisgogram bins to use
- **params["colourmap"]** – (string) colourmap to use
- **params["sorting_choice"]** – 2-element list. [0] sets sort direction, [1] used to determine which field to sort
- **params["global_max_min"]** – (boolean) flag to calculate global max and min or per-subplot max and min
- **params["legend_title"]** – (string) legend title

- **params["return_as_dict"]** – (boolean) flag to trigger return as a dictionary rather than a HTML DIV

- **params["filename"]** – (string) default filename to use for plot bitmap export

Rückgabe

Plotly figure embedded in an HTML DIV; or Plotly figure as a dictionary (if params[„return_as_dict"] is True); or an error message embedded in an HTML DIV if there was a ValueError when calculating the figure

remapp.interface.chart_functions.**plotly_scatter**(*df*, *params*)

Create a plotly scatter chart

Parameter

- **df** – Pandas DataFrame containing the data

- **params** – a dictionary of parameters

- **params["df_name_col"]** – (string) DataFrame column containing categories

- **params["df_x_col"]** – (string) DataFrame column containing x values

- **params["df_y_col"]** – (string) DataFrame column containing y values

- **params["sorting_choice"]** – 2-element list. [0] sets sort direction, [1] used to determine which field to sort

- **params["grouping_choice"]** – (string) „series" or „system"

- **params["legend_title"]** – (string) legend title

- **params["facet_col_wrap"]** – (int) number of subplots per row

- **params["colourmap"]** – (string) colourmap to use

- **params["x_axis_title"]** – (string) x-axis title

- **params["y_axis_title"]** – (string) y-axis title

- **params["filename"]** – (string) default filename to use for plot bitmap export

- **params["return_as_dict"]** – (boolean) flag to trigger return as a dictionary rather than a HTML DIV

Rückgabe

Plotly figure embedded in an HTML DIV; or Plotly figure as a dictionary (if „return_as_dict" is True); or an error message embedded in an HTML DIV if there was a ValueError when calculating the figure

remapp.interface.chart_functions.**plotly_set_default_theme**(*theme_name*)

A short method to set the plotly chart theme

Parameter

theme_name – the name of the theme

Rückgabe

remapp.interface.chart_functions.**plotly_timeseries_linechart**(*df*, *params*)

Create a plotly line chart of data over time

Parameter

- **df** – Pandas DataFrame containing the data

- **params** – a dictionary of parameters

- **params["df_facet_col"]** – (string) DataFrame column used to create subplots
- **params["df_facet_col_wrap"]** – (int) number of subplots per row
- **params["facet_title"]** – (string) subplot title
- **params["df_value_col"]** – (string) DataFrame column containing values
- **params["value_axis_title"]** – (string) y-axis title
- **params["colourmap"]** – (string) colourmap to use
- **params["colourmap"]** – (string) colourmap to use
- **params["df_date_col"]** – (string) DataFrame column containing dates
- **params["df_count_col"]** – (string) DataFrame column containing frequency data
- **params["df_name_col"]** – (string) DataFrame column containing categories
- **params["legend_title"]** – (string) legend title
- **params["name_axis_title"]** – (string) x-axis title
- **params["return_as_dict"]** – (boolean) flag to trigger return as a dictionary rather than a HTML DIV
- **params["filename"]** – (string) default filename to use for plot bitmap export

> **Rückgabe**
> Plotly figure embedded in an HTML DIV; or Plotly figure as a dictionary (if „return_as_dict" is True); or an error message embedded in an HTML DIV if there was a ValueError when calculating the figure

remapp.interface.chart_functions.**save_fig_as_html_div**(*fig*, *filename*, *active=False*)

Saves the Plotly figure as an HTML file containing a single DIV. The file is saved on the OpenREM server in MEDIA_ROOTchartsyyyymmdd. Viewing the saved file requires an active internet connection as the Plotly JavaScript library is not included in the file.

This method is not currently accessible to an OpenREM user or administrator - it is present to assist developers when producing example charts for the OpenREM documentation. It must be manually activated by setting active=True in the method definition.

> **Parameter**
> - **fig** – a Plotly figure
> - **filename** – (string)the filename to use
> - **active** – (boolean) to set whether to save the figure

11.15 DICOM networking modules

11.15.1 Query-retrieve module

Query function

openrem.remapp.netdicom.qrscu.**qrscu**(*qr_scp_pk=None*, *store_scp_pk=None*, *implicit=False*,
explicit=False, *move=False*, *query_id=None*, *date_from=None*,
date_until=None, *single_date=False*, *time_from=None*,
time_until=None, *modalities=None*, *inc_sr=False*,
remove_duplicates=True, *filters=None*, *get_toshiba_images=False*,
get_empty_sr=False)

Query retrieve service class user function

Queries a pre-configured remote query retrieve service class provider for dose metric related objects, making use
of the filter parameters provided. Can automatically trigger a c-move (retrieve) operation.

> **Parameter**
>
> - **qr_scp_pk** (`int, optional`) – Database ID/pk of the remote QR SCP (Default value =
>   None)
>
> - **store_scp_pk** (`int, optional`) – Database ID/pk of the local store SCP (Default value
>   = None)
>
> - **implicit** (`bool, optional`) – Prefer implicit transfer syntax (preference possibly not im-
>   plemented) (Default value = False)
>
> - **explicit** (`bool, optional`) – Prefer explicit transfer syntax (preference possibly not im-
>   plemented) (Default value = False)
>
> - **move** (`bool, optional`) – Automatically trigger move request when query is complete
>   (Default value = False)
>
> - **query_id** (`str, optional`) – UID of query if generated by web interface (Default value
>   = None)
>
> - **date_from** (`str, optional`) – Date to search from, format yyyy-mm-dd (Default value =
>   None)
>
> - **date_until** (`str, optional`) – Date to search until, format yyyy-mm-dd (Default value
>   = None)
>
> - **single_date** (`bool, optional`) – search only on date_from, allows ti-
>   me_from/time_until (Default value = False)
>
> - **time_from** (`str, optional`) – Time of day to search from, format hhmm 24 hour clock,
>   single date only (Default value = None)
>
> - **time_until** (`str, optional`) – Time of day to search until, format hhmm 24 hour clock,
>   single date only (Default value = None)
>
> - **modalities** (`list, optional`) – Modalities to search for, options are CT, MG, DX and
>   FL (Default value = None)
>
> - **inc_sr** (`bool, optional`) – Only include studies that only have structured reports in (un-
>   known modality) (Default value = False)
>
> - **remove_duplicates** (`bool, optional`) – If True, studies that already exist in the data-
>   base are removed from the query results (Default value = True)
>
> - **filters** (`dictionary list, optional`) – lowercase include and exclude lists for Sta-
>   tionName and StudyDescription (Default value = None)
>
> - **get_toshiba_images** (`bool, optional`) – Whether to try to get Toshiba dose summary
>   images
>
> - **get_empty_sr** (`bool, optional`) – Whether to get SR series that return nothing at image
>   level

> **Rückgabe**
>> Series Instance UIDs are stored as rows in the database to be used by a move request. Move request is optionally triggered automatically.

Move function

openrem.remapp.netdicom.qrscu.**movescu**(*query_id*)

> C-Move request element of query-retrieve service class user :param query_id: ID of query in the DicomQuery table :return: None

openrem_qr.py script

Query remote server and retrieve to OpenREM

```
usage: openrem_qr.py [-h] [-ct] [-mg] [-fl] [-dx] [-nm] [-f yyyy-mm-dd]
                     [-t yyyy-mm-dd] [-sd yyyy-mm-dd] [-tf hhmm] [-tt hhmm]
                     [-e string] [-i string] [-sne string] [-sni string]
                     [--stationname_study_level] [-toshiba] [-sr] [-dup]
                     [-emptysr]
                     qr_id store_id
```

Positional Arguments

qr_id	Database ID of the remote QR node
store_id	Database ID of the local store node

Named Arguments

-ct	Query for CT studies. Cannot be used with -sr
	Default: False
-mg	Query for mammography studies. Cannot be used with -sr
	Default: False
-fl	Query for fluoroscopy studies. Cannot be used with -sr
	Default: False
-dx	Query for planar X-ray studies (includes panoramic X-ray studies). Cannot be used with -sr
	Default: False
-nm	Query for nuclear medicine studies. Cannot be used with -sr
	Default: False
-f, --dfrom	Date from, format yyyy-mm-dd. Cannot be used with –single_date
-t, --duntil	Date until, format yyyy-mm-dd. Cannot be used with –single_date
-sd, --single_date	Date, format yyy-mm-dd. Cannot be used with –dfrom or –duntil

-tf, --tfrom	Time from, format hhmm. Requires –single_date.
-tt, --tuntil	Time until, format hhmm. Requires –single_date.
-e, --desc_exclude	Terms to exclude in study description, comma separated, quote whole string
-i, --desc_include	Terms that must be included in study description, comma separated, quote whole string
-sne, --stationname_exclude	Terms to exclude in station name, comma separated, quote whole string
-sni, --stationname_include	Terms to include in station name, comma separated, quote whole string
--stationname_study_level	Advanced: Filter station name at Study level, instead of at Series level
	Default: False
-toshiba	Advanced: Attempt to retrieve CT dose summary objects and one image from each series
	Default: False
-sr	Advanced: Use if store has RDSRs only, no images. Cannot be used with -ct, -mg, -fl, -dx
	Default: False
-dup	Advanced: Retrieve duplicates (objects that have been processed before)
	Default: False
-emptysr	Advanced: Get SR series that return nothing at image level query
	Default: False

11.15.2 NetDICOM common functions

openrem.remapp.netdicom.tools.**echoscu**(*scp_pk=None*, *store_scp=False*, *qr_scp=False*)

> Function to check if built-in Store SCP or remote Query-Retrieve SCP returns a DICOM echo :param scp_pk: Primary key if either Store or QR SCP in database :param store_scp: True if checking Store SCP :param qr_scp: True if checking QR SCP :return: ‚AssocFail', Success or ?

11.16 BackgroundTask

Contains functions for running tasks in the background.

remapp.tools.background.**get_current_task**()

> **Rückgabe**
> The associated BackgroundTask object when called in a task. If this is not executed in a background Task None will be returned.

remapp.tools.background.**get_or_generate_task_uuid**()

> **Rückgabe**
> If called from within a task the task id, else a generated uuid

remapp.tools.background.**get_queued_tasks**(*task_type=None*) → List[QueuedTask]

> Returns all task which are currently waiting for execution

>> **Parameter**
>>> **task_type** – Optionally filter by task type

>> **Rückgabe**
>>> List of queued tasks

remapp.tools.background.**record_task_error_exit**(*error_msg*)

> Small helper that checks if we are in a task and assuming we are records error_msg as well as setting the completed_successfully to false and completed to True. Note that get_current_task will return None after a call to this.

remapp.tools.background.**record_task_info**(*info_msg*)

> Small helper that checks if we are in a task and assuming we are records info_msg as info.

remapp.tools.background.**record_task_related_query**(*study_instance_uid*)

> Tries to find the related DicomQRRspStudy object given a study instance uid and if this is running in a task will record it to the query object. This is used to later find the import tasks that were run as part of a query. Since this actually just takes the latest query if the user runs imports manually via script it may in principle wrongly associate.

remapp.tools.background.**remove_task_from_queue**(*task_id: str*)

> Removes task from queue.

>> **Parameter**
>>> **task_id** – task id of the task in question

remapp.tools.background.**run_in_background**(*func*, *task_type*, **args*, ***kwargs*) → Result

> Syntactic sugar around run_in_background_with_limits.

> Arguments correspond to run_in_background_with_limits. Will always run the passed function as fast as possible. This function should always be used for functions triggered from the webinterface.

remapp.tools.background.**run_in_background_with_limits**(*func*, *task_type*, *num_proc*, *num_of_task_type*, **args*, ***kwargs*) → Result

> Runs func as background Process.

> This method will create a new task which will be scheduled to be run by the Huey consumer with the specified priority (defaults to 0). The priority can be passed via a keyword argument

> Internally the Huey consumer spawns a new process, which then creates a BackgroundTask object. This can be obtained via get_current_task() inside the calling process. Note that BackgroundTask objects will not be deleted onto completion - instead the complete flag will be set to True.

> num_proc and num_of_task_type can be used to give conditions on the start.

>> **Parameter**

>>> • **func** – The function to run. Note that you should set the status of the task yourself and mark as completed when exiting yourself e.g. via sys.exit(). Assuming the function returns normally on success or returns with an exception on error, the status of the BackgroundTask object will be set correctly.

>>> • **task_type** – One of the strings declared in BackgroundTask.task_type. Indicates which kind of background process this is supposed to be. (E.g. move, query, . . .)

>>> • **num_proc** – Will wait with execution until there are less than num_proc active tasks. If num_proc == 0 will run directly

- **num_of_task_type** – A dictionary from str to int, where key should be some used task_type. Will wait until there are less active tasks of task_type than specified in the value of the dict for that key.

- **args** – Positional arguments. Passed to func.

- **kwargs** – Keywords arguments. Passed to func.

> **Rückgabe**
> The BackgroundTask object.

remapp.tools.background.**terminate_background**(*task:* BackgroundTask)

> Terminate a background task by force. Sets complete=True on the task object.

remapp.tools.background.**wait_task**(*task: Result*)

> Wait until the task has completed

11.17 Adding new charts

To add a new chart several files need to be updated:

- `models.py`

- `forms.py`

- `views.py`

- `xxfiltered.html`

- `xxChartAjax.js`

- `displaychartoptions.html`

Where xx is one of `ct`, `dx`, `mg` or `rf`

The additions to the files add:

- database fields in the user profile to control whether the new charts are plotted (`models.py`)

- new options on the chart plotting forms (`forms.py`, `displaychartoptions.html`)

- extra code to calculate the data for the new charts if they are switched on (`views.py`)

- a section of html and JavaScript to contain the charts (`xxfiltered.html`)

- a section of JavaScript to pass the data calculated by `views.py` to `xxfiltered.html`

The process is probably best illustrated with an example. What follows is a description of how to add a new chart that displays study workload for fluoroscopy, and a pie chart of study description frequency.

11.17.1 Additions to `models.py`

A field per chart needs to be added to the `UserProfile` section in `models.py` to control whether the new charts should be plotted. There is a section of this file that looks like the following:

```
plotCTAcquisitionMeanDLP = models.BooleanField(default=True)
plotCTAcquisitionMeanCTDI = models.BooleanField(default=True)
plotCTAcquisitionFreq = models.BooleanField(default=False)
plotCTStudyMeanDLP = models.BooleanField(default=True)
plotCTStudyFreq = models.BooleanField(default=False)
```

```
plotCTRequestMeanDLP = models.BooleanField(default=False)
plotCTRequestFreq = models.BooleanField(default=False)
plotCTStudyPerDayAndHour = models.BooleanField(default=False)
plotCTStudyMeanDLPOverTime = models.BooleanField(default=False)
plotCTStudyMeanDLPOverTimePeriod = models.CharField(max_length=6,
                                                    choices=TIME_PERIOD,
                                                    default=MONTHS)
plotCTInitialSortingChoice = models.CharField(max_length=4,
                                              choices=SORTING_CHOICES_CT,
                                              default=FREQ)
```

Two new lines needs to be added to this section, using appropriate names such as:

```
plotRFStudyPerDayAndHour = models.BooleanField(default=False)
plotRFStudyFreq = models.BooleanField(default=False)
```

Adding new fields to `models.py` requires that a database migration is carried out to add the fields to the database. This is done via the command line:

```
python manage.py makemigrations remapp
python manage.py migrate remapp
```

The first command should result in a response similar to:

```
Migrations for 'remapp':
  0004_auto_20160424_1116.py:
    - Add field plotRFAcquisitionCTDIOverTime to userprofile
    - Add field plotRFStudyFreq to userprofile
```

The second command should result in a response similar to:

```
Operations to perform:
  Apply all migrations: remapp
Running migrations:
  Rendering model states... DONE
  Applying remapp.0004_auto_20160424_1116... OK
```

That's the end of the changes required in `models.py`

11.17.2 Additions to `forms.py`

Two additional lines need to be added to the `XXChartOptionsForm` and `XXChartOptionsDisplayForm` methods in `forms.py`, where `XX` is one of `CT`, `DX`, `MG` or `RF`.

For our new charts the following lines needs to be added to both `RFChartOptionsForm` and `RFChartOptionsDisplayForm`:

```
plotRFStudyPerDayAndHour = forms.BooleanField(label='Study workload', required=False)
plotRFStudyFreq = forms.BooleanField(label='Study frequency', required=False)
```

In addition, a new method needs to be added so that the RF chart options are shown when the user goes to Config -> Chart options:

```
class RFChartOptionsDisplayForm(forms.Form):
    plotRFStudyPerDayAndHour = forms.BooleanField(label='Study workload', required=False)
    plotRFStudyFreq = forms.BooleanField(label='Study frequency', required=False)
```

That's the end of the changes required in `forms.py`

11.17.3 Additions to `views.py`

Four methods in this file need to be updated.

`xx_summary_list_filter` additions

Some additions need to be made to the `xx_summary_list_filter` method in `views.py`, where `xx` is one of `ct`, `dx`, `mg` or `rf`. As we're adding new RF charts, we need to edit `rf_summary_list_filter`.

A section of this method examines the user's chart plotting preferences. Code must be added to include the new chart in these checks. An abbreviated version of the section is shown below.

```
# Obtain the chart options from the request
chart_options_form = RFChartOptionsForm(request.GET)
# Check whether the form data is valid
if chart_options_form.is_valid():
    # Use the form data if the user clicked on the submit button
    if "submit" in request.GET:
        # process the data in form.cleaned_data as required
        user_profile.plotCharts = chart_options_form.cleaned_data['plotCharts']
        if median_available:
            user_profile.plotAverageChoice = chart_options_form.cleaned_data[
→'plotMeanMedianOrBoth']
        user_profile.save()

    else:
        form_data = {'plotCharts': user_profile.plotCharts,
                     'plotMeanMedianOrBoth': user_profile.plotAverageChoice}
        chart_options_form = RFChartOptionsForm(form_data)
```

Two new lines needs to be inserted into the `if` and `else` sections for the new chart:

```
# Obtain the chart options from the request
chart_options_form = RFChartOptionsForm(request.GET)
# Check whether the form data is valid
if chart_options_form.is_valid():
    # Use the form data if the user clicked on the submit button
    if "submit" in request.GET:
        # process the data in form.cleaned_data as required
        user_profile.plotCharts = chart_options_form.cleaned_data['plotCharts']
        user_profile.plotRFStudyPerDayAndHour = chart_options_form.cleaned_data[
→'plotRFStudyPerDayAndHour']
        user_profile.plotRFStudyFreq = chart_options_form.cleaned_data['plotRFStudyFreq']
        if median_available:
            user_profile.plotAverageChoice = chart_options_form.cleaned_data[
→'plotMeanMedianOrBoth']
```

```
        user_profile.save()

    else:
        form_data = {'plotCharts': user_profile.plotCharts,
                      'plotRFStudyPerDayAndHour': user_profile.plotRFStudyPerDayAndHour,
                      'plotRFStudyFreq': user_profile.plotRFStudyFreq,
                      'plotMeanMedianOrBoth': user_profile.plotAverageChoice}
        chart_options_form = RFChartOptionsForm(form_data)
```

`xx_summary_chart_data` additions

The `return_structure` variable needs the new user_profile fields adding.

Before:

```
return_structure =\
    rf_plot_calculations(f, request_results, median_available, user_profile.
→plotAverageChoice,
                         user_profile.plotSeriesPerSystem, user_profile.
→plotHistogramBins,
                         user_profile.plotHistograms)
```

After:

```
return_structure =\
    rf_plot_calculations(f, request_results, median_available, user_profile.
→plotAverageChoice,
                         user_profile.plotSeriesPerSystem, user_profile.
→plotHistogramBins,
                         user_profile.plotRFStudyPerDayAndHour,  user_profile.
→plotRFStudyFreq,
                         user_profile.plotHistograms)
```

`xx_plot_calculations` additions

Two items needs to be added to this method's parameters.

Before:

```
def rf_plot_calculations(f, request_results, median_available, plot_average_choice, plot_
→series_per_systems,
                         plot_histogram_bins, plot_histograms):
```

After:

```
def rf_plot_calculations(f, request_results, median_available, plot_average_choice, plot_
→series_per_systems,
                         plot_histogram_bins, plot_study_per_day_and_hour, plot_study_
→freq, plot_histograms):
```

Our new charts makes use of `study_events` (rather than `acquisition_events` or `request_events`). We therefore need to ensure that `study_events` are available if the user has chosen to show the new chart.

After additions:

```
if plot_study_per_day_and_hour:
    study_events = f.qs
```

We now need to add code that will calculate the data for the new charts. This uses one of the methods in the `chart_functions.py` file, located in the `interface` folder of the OpenREM project.

```
if plot_study_per_day_and_hour:
    result = workload_chart_data(study_events)
    return_structure['studiesPerHourInWeekdays'] = result['workload']

if plot_study_freq:
    result = average_chart_inc_histogram_data(study_events,
                                              'generalequipmentmoduleattr__unique_
→equipment_name_id__display_name',
                                              'study_description',
                                              'projectionxrayradiationdose__
→accumxraydose__accumintegratedprojradiogdose__dose_area_product_total',
                                              1000000,
                                              plot_study_dap, plot_study_freq,
                                              plot_series_per_systems, plot_average_
→choice,
                                              median_available, plot_histogram_bins,
                                              calculate_histograms=plot_histograms)

    return_structure['studySystemList'] = result['system_list']
    return_structure['studyNameList'] = result['series_names']
    return_structure['studySummary'] = result['summary']
```

The data in `return_structure` will now be available to the browser via JavaScript, and can be used to populate the charts themselves.

`chart_options_view` additions

The RF options form need to be imported

Before:

```
from remapp.forms import GeneralChartOptionsDisplayForm, DXChartOptionsDisplayForm,
→CTChartOptionsDisplayForm
```

After:

```
from remapp.forms import GeneralChartOptionsDisplayForm, DXChartOptionsDisplayForm,
→CTChartOptionsDisplayForm,\
    RFChartOptionsDisplayForm
```

The RF form items need to be included

Before (abbreviated):

```
if request.method == 'POST':
    general_form = GeneralChartOptionsDisplayForm(request.POST)
```

```
    ct_form = CTChartOptionsDisplayForm(request.POST)
    dx_form = DXChartOptionsDisplayForm(request.POST)
    if general_form.is_valid() and ct_form.is_valid() and dx_form.is_valid() and rf_form.
→is_valid():
        try:
            # See if the user has plot settings in userprofile
            user_profile = request.user.userprofile
        except:
            # Create a default userprofile for the user if one doesn't exist
            create_user_profile(sender=request.user, instance=request.user, created=True)
            user_profile = request.user.userprofile

        user_profile.plotCharts = general_form.cleaned_data['plotCharts']
        ...
        ...
        user_profile.plotHistogramBins = general_form.cleaned_data['plotHistogramBins']

        user_profile.plotCTAcquisitionMeanDLP = ct_form.cleaned_data[
→'plotCTAcquisitionMeanDLP']
        ...
        ...
        user_profile.plotCTInitialSortingChoice = ct_form.cleaned_data[
→'plotCTInitialSortingChoice']

        user_profile.plotDXAcquisitionMeanDAP = dx_form.cleaned_data[
→'plotDXAcquisitionMeanDAP']
        ...
        ...
        user_profile.plotDXInitialSortingChoice = dx_form.cleaned_data[
→'plotDXInitialSortingChoice']

        user_profile.save()

    messages.success(request, "Chart options have been updated")

...
...

general_form_data = {'plotCharts': user_profile.plotCharts,
                     'plotMeanMedianOrBoth': user_profile.plotAverageChoice,
                     'plotInitialSortingDirection': user_profile.
→plotInitialSortingDirection,
                     'plotSeriesPerSystem': user_profile.plotSeriesPerSystem,
                     'plotHistogramBins': user_profile.plotHistogramBins}

ct_form_data = {'plotCTAcquisitionMeanDLP': user_profile.plotCTAcquisitionMeanDLP,
                ...
                ...
                'plotCTInitialSortingChoice': user_profile.plotCTInitialSortingChoice}

dx_form_data = {'plotDXAcquisitionMeanDAP': user_profile.plotDXAcquisitionMeanDAP,
                ...
```

```
                ...
                'plotDXInitialSortingChoice': user_profile.plotDXInitialSortingChoice}


general_chart_options_form = GeneralChartOptionsDisplayForm(general_form_data)
ct_chart_options_form = CTChartOptionsDisplayForm(ct_form_data)
dx_chart_options_form = DXChartOptionsDisplayForm(dx_form_data)

return_structure = {'admin': admin,
                    'GeneralChartOptionsForm': general_chart_options_form,
                    'CTChartOptionsForm': ct_chart_options_form,
                    'DXChartOptionsForm': dx_chart_options_form
                    }
```

After (abbreviated):

```
if request.method == 'POST':
    general_form = GeneralChartOptionsDisplayForm(request.POST)
    ct_form = CTChartOptionsDisplayForm(request.POST)
    dx_form = DXChartOptionsDisplayForm(request.POST)
    rf_form = RFChartOptionsDisplayForm(request.POST)
    if general_form.is_valid() and ct_form.is_valid() and dx_form.is_valid() and rf_form.
→is_valid():
        try:
            # See if the user has plot settings in userprofile
            user_profile = request.user.userprofile
        except:
            # Create a default userprofile for the user if one doesn't exist
            create_user_profile(sender=request.user, instance=request.user, created=True)
            user_profile = request.user.userprofile

        user_profile.plotCharts = general_form.cleaned_data['plotCharts']
        ...
        ...
        user_profile.plotHistogramBins = general_form.cleaned_data['plotHistogramBins']

        user_profile.plotCTAcquisitionMeanDLP = ct_form.cleaned_data[
→'plotCTAcquisitionMeanDLP']
        ...
        ...
        user_profile.plotCTInitialSortingChoice = ct_form.cleaned_data[
→'plotCTInitialSortingChoice']

        user_profile.plotDXAcquisitionMeanDAP = dx_form.cleaned_data[
→'plotDXAcquisitionMeanDAP']
        ...
        ...
        user_profile.plotDXInitialSortingChoice = dx_form.cleaned_data[
→'plotDXInitialSortingChoice']

        user_profile.plotRFStudyPerDayAndHour = rf_form.cleaned_data[
→'plotRFStudyPerDayAndHour']
```

```
        user_profile.plotRFStudyFreq = rf_form.cleaned_data['plotRFStudyFreq']

        user_profile.save()

    messages.success(request, "Chart options have been updated")

...
...

general_form_data = {'plotCharts': user_profile.plotCharts,
                     ...
                     ...
                     'plotHistogramBins': user_profile.plotHistogramBins}

ct_form_data = {'plotCTAcquisitionMeanDLP': user_profile.plotCTAcquisitionMeanDLP,
                ...
                ...
                'plotCTInitialSortingChoice': user_profile.plotCTInitialSortingChoice}

dx_form_data = {'plotDXAcquisitionMeanDAP': user_profile.plotDXAcquisitionMeanDAP,
                ...
                ...
                'plotDXInitialSortingChoice': user_profile.plotDXInitialSortingChoice}

rf_form_data = {'plotDXStudyPerDayAndHour': user_profile.plotDXStudyPerDayAndHour,
                'plotRFStudyFreq': user_profile.plotRFStudyFreq}

general_chart_options_form = GeneralChartOptionsDisplayForm(general_form_data)
ct_chart_options_form = CTChartOptionsDisplayForm(ct_form_data)
dx_chart_options_form = DXChartOptionsDisplayForm(dx_form_data)
rf_chart_options_form = RFChartOptionsDisplayForm(rf_form_data)

return_structure = {'admin': admin,
                    'GeneralChartOptionsForm': general_chart_options_form,
                    'CTChartOptionsForm': ct_chart_options_form,
                    'DXChartOptionsForm': dx_chart_options_form,
                    'RFChartOptionsForm': rf_chart_options_form,
                    }
```

11.17.4 Additions to `displaychartoptions.html`

A new div needs to be added for the fluoroscopy chart options:

```html
<div class="panel-heading">
  <h3 class="panel-title">Fluoroscopy chart options</h3>
</div>
<div class="panel-body">
  <table>
    {% csrf_token %}
    {{ RFChartOptionsForm }}
  </table>
```

```
    <input class="btn btn-default" name="submit" type="submit" />
</div>
```

11.17.5 Additions to `rffiltered.html`

A section of this file sets a JavaScript variable per chart. Two new ones needs to be added.

Additions:

```
{% if request.user.userprofile.plotRFStudyPerDayAndHour %}
    <script>
        var plotRFStudyPerDayAndHour = true;
        result = chartWorkload('piechartStudyWorkloadDIV', 'Studies');
    </script>
{% endif %}

{% if request.user.userprofile.plotRFStudyFreq %}
    <script>
        var plotRFStudyFreq = true;
        var urlStartStudy = '/openrem/rf/?{% for field in filter.form %}{% if field.name
→!= 'study_description' and field.name != 'o' and field.value %}&{{ field.name }}={{
→field.value }}{% endif %}{% endfor %}&study_description=';
        result = chartFrequency('piechartStudyDIV', 'Study description frequency');
    </script>
{% endif %}
```

A second section of code needs to be added to `rffiltered.html` to include a DIV for the new charts:

```
{% if request.user.userprofile.plotRFStudyPerDayAndHour %}
    <!-- HTML to include div container for study workload -->

    <script>
        $(window).resize(function() {
            chartSetExportSize('piechartStudyWorkloadDIV');
            fitChartToDiv('piechartStudyWorkloadDIV');
        });
    </script>

    <div class="panel-group" id="accordion5">
        <div class="panel panel-default">
            <div class="panel-heading">
                <h4 class="panel-title">
                    <a data-toggle="collapse" data-parent="#accordion5" href="
→#collapseStudyWorkloadPieChart" onclick="setTimeout(function() {$(document).resize();},
→ 0);">
                        Pie chart showing a breakdown of number of studies per weekday.
                    </a>
                </h4>
            </div>
            <div id="collapseStudyWorkloadPieChart" class="panel-collapse collapse">
                <div class="panel-body">
```

```
                    <div id="piechartStudyWorkloadDIV" style="height: auto; margin: 0 0">
→</div>
                    <p>Click on a segment to be taken to a pie chart showing the␣
→breakdown per hour for that weekday.</p>
                    <a onclick="enterFullScreen('collapseStudyWorkloadPieChart',
→'piechartStudyWorkloadDIV')" class="btn btn-default btn-sm" role="button">Toggle␣
→fullscreen</a>
                </div>
            </div>
        </div>
    </div>
    <!-- End of HTML to include div container for studies per week day pie chart -->
{% endif %}

{% if request.user.userprofile.plotRFStudyFreq %}
    <!-- HTML to include div container for study name pie chart -->

    <script>
        $(window).resize(function() {
            chartSetExportSize('piechartStudyDIV');
            fitChartToDiv('piechartStudyDIV');
        });
    </script>

    <div class="panel-group" id="accordionPiechartStudy">
        <div class="panel panel-default">
            <div class="panel-heading">
                <h4 class="panel-title">
                    <a data-toggle="collapse" data-parent="#accordionPiechartStudy" href=
→"#collapseStudyPieChart" onclick="setTimeout(function() {$(document).resize();}, 0);">
                        Pie chart showing a breakdown of study name frequency.
                    </a>
                </h4>
            </div>
            <div id="collapseStudyPieChart" class="panel-collapse collapse">
                <div class="panel-body">
                    <div id="piechartStudyDIV" style="height: auto; margin: 0 0"></div>
                    <a onclick="enterFullScreen('collapseStudyPieChart',
→'piechartStudyDIV')" class="btn btn-default btn-sm" role="button">Toggle fullscreen</a>
                </div>
            </div>
        </div>
    </div>
    <!-- End of HTML to include div container for study name pie chart -->
{% endif %}
```

11.17.6 Additions to `rfChartAjax.js`

This file needs to update the skeleton chart with the data that has been provided by `views.py`. It does this via the appropriate routines contained in the `chartUpdateData.js` file. In this case, `updateWorkloadChart` and `updateFrequencyChart`:

```
// Study workload chart data
if(typeof plotRFStudyPerDayAndHour !== 'undefined') {
    updateWorkloadChart(json.studiesPerHourInWeekdays, 'piechartStudyWorkloadDIV',␣
→colour_scale);
}

// Study description frequency chart data start
if(typeof plotRFStudyFreq !== 'undefined') {
    updateFrequencyChart(json.studyNameList, json.studySystemList, json.studySummary,␣
→urlStartStudy, 'piechartStudyDIV', colour_scale);
}
```

That's it - you should now have two new charts visible in the fluoroscopy filtered page.

11.18 Indizes und Tabellen

- genindex
- modindex
- search

Release Notes and Change Log

12.1 Version history change log

12.1.1 OpenREM version history

1.0.0b2 (2023-04-20)

- #995 SkinDose: fixed bug where a missing exposure_time causes an invalid decimal operation
- #992 Charts: fixed bug where an empty x-ray system category caused boxplots to fail
- #991 SkinDose: fixed bug in e-mail alert logic so user options are respected
- #990 SkinDose: fixed bug where a zero study dap causes a TypeError in the e-mail sending code
- #988 Imports: fixing hard-written DX text in MG import messages
- #987 Tasks: configuring Windows WinSW log files to roll over to ensure their size is restricted
- #986 SkinDose: fixed bug where a zero study dap causes a divide by zero error
- #984 Interface: improved performance of home page
- #981 Interface: reorganised the config menu
- #980 Interface: improved standard name form layout
- #977 Documentation: fix issues preventing translations to be built
- #976 NM charts: Fixed bug where graphs were not displayed correctly for some chart options
- #972 DICOM Networking: better analysis of SR series with multiple SOP Class UIDs
- #971 Imports: fixing NM image index error
- #970 Emails: fixed formatting of high fluoro dose alert emails and included peak skin dose for each study in time delta
- #967 Imports: made total DLP optional for Philips CT imports

- #965 SkinDose: fixed bug where a zero study dap causes a divide by zero error
- #964 Tasks: restricted default task history to 2000 stored entries; enabled admin users to configure this value
- #962 Documentation: updated packages to make bullet points work again
- #960 SkinDose: fixed bug where multiple entries in the openskin safelist table with different software versions caused an error
- #959 Installation: upgraded packages to latest versions
- #958 Interface: fixed task table sorting for Started column
- #957 Imports: fixed DX extractor total_number_of_radiographic_frames is NoneType error
- #956 Interface: enable updating of standard name database links
- #955 Imports: fixed error when importing RDSR with empty DAP measured value sequence
- #953 Interface: updated „DX and CR" to „Radiography", and „Radiographic" to „Radiography"
- #950 Documentation: removed (outdated) instructions for Conquest
- #949 Documentation: updated the upgrade instructions from older versions
- #947 Tests: enforce ordering within fluoro exposure sets and specify row by time in test
- #945 Exports: order by plane when populating fluoro data
- #942 SkinDose: try to calculate number of frames using exposure time / pulse width if number of frames not available
- #941 Interface: the filtering submit button now updates chart options for fluoroscopy and mammography
- #940 Installation: upgraded Django to 3.2, packages to latest versions
- #937 Interface: correcting bottom row of exports table
- #936 Tasks: added make_skin_map to background tasks on RF RDSR import
- #935 SkinDose: fixed bug which applied 5x5 cm backscatter factor regardless of field size at skin
- #934 DICOM Networking: QR queries are now logged and can be analysed through the web interface
- #933 SkinDose: bug fixed that caused incorrect field size at skin for exposures with a non-zero secondary angle (caudocranial)
- #931 Exports: export RF DAP as float instead of text
- #928 Documentation: added restriction in postgres version for earlier OpenREM releases
- #925 Docker: docs and config file for enabling bind mounts with SELinux
- #923 Docker: docs and config for virtual directory install
- #922 Database: optimise indexes and duplicate queries
- #919 Interface: fixed bug preventing home page listing if study had no date
- #917 Interface: added horizontal lines between chart option groups and shaded chart option CheckboxSelectMultiple items
- #915 Interface and exports: expose mammo view modifier in interface and exports
- #914 Imports: added compression force, pressure, contact area for mammo RDSR
- #913 SkinDose: made 2d skin dose map overlay visible by default

- #911 Charts: fixed issue with chart data sorting and added label wrap option

- #910 SkinDose: fixed rendering of 2d skin dose map with head

- #909 Code quality: all model imports absolute

- #908 Imports: enabled Device Observer UID to be ignored for specific equipment models when creating display name data during RDSR import

- #906 Charts: upgraded Plotly library to latest version

- #905 Imports: fixed filter extraction code not working for Siemens Multix DX

- #904 Testing: bumped Python image from 3.6 to 3.8

- #903 Interface: added patient weight filters to the CT, RF and DX summary pages

- #901 Charts: fixed issue where mammography mAs values were displayed 1000x too high on scatter plot

- #897 Docker: fixed permissions for PixelMed - now using root throughout

- #896 Imports: enabling import of DX with text string in PatientSize field

- #893 Charts: fixed issue with over-time charts with many sub-plots failing to plot correctly

- #892 Documentation: Removing references to native DICOM store and log

- #888 SkinDose: added option to support skin dose calculations for listed systems only

- #886 Code quality: addressed some SonarCloud issues

- #882 SkinDose: added percentage of exposures that interact with phantom

- #881 Charts: add option to remove multiple and trailing whitespace in category names

- #880 Orthanc: added XA and RF to allowed modalities to enable physics QA images to be kept

- #879 Charts: fixed sorting of fluoroscopy charts when split by physician

- #877 Charts: added acquisition type restrictions to acquisition-level CT charts

- #874 Documentation: updating DICOM query-retrieve documentation

- #872 Charts: added ability to split fluoroscopy over-time and histogram charts by physician

- #871 Charts: corrected RF chart x-axis labels

- #870 Charts: reduced memory footprint of Pandas DataFrame creation

- #869 Charts: added doc strings to new chart code

- #868 Docker: fixed Python version at 3.8

- #867 Documentation: updated chart documentation

- #866 Simplified code for different character sets, enabled MultiValue SpecificCharacterSet

- #865 Imports: enabled workaround to import Spectrum Dynamics RDSR

- #864 Tasks: updated Celery settings for Celery 6.

- #863 Interface: removed height and weight from CT study delete

- #862 Interface: allow mapping of request, study and acquisition names to standard versions

- #861 Interface: added ability to filter mammography on view code, compressed breast thickness and exposure control mode

- #860 DICOM Networking: removed built-in DICOM Store SCP functionality

- #858 DICOM Networking: query-retrieve logging, filtering and error handling improved
- #857 Documentation: resolved documentation build errors
- #856 Interface: removed CT acquisition type restriction tick boxes
- #854 Interface: added date constraints to links on homepage
- #853 Testing: reduced Bitbucket pipeline minutes usage
- #852 Code quality: skin dose code formatted with Black
- #850 Emails: added oldest study accession number to high fluoro dose alert email subject
- #849 Docker: make docker URL an env variable
- #847 Documentation: added copy button to commands, added prompts where appropriate
- #845 Docker: moved Nginx config to bind folder
- #844 Code quality: getting the pipelines right
- #843 Code quality: closing target _blank phishing vulnerability
- #842 Imports: ContextID code_meaning in make_skin_map and dxdetail
- #841 Code quality: format code with Black
- #840 Exports: added performing physician to fluoroscopy standard exports
- #839 Documentation: removed redundant troubleshooting docs
- #838 Imports: fixed issues with changed PersonName behaviour in pydicom 2.0
- #836 Installation: updated requirements, Docker and docs for pynetdicom 1.5, pydicom 2.0
- #835 Docker: fixed timeout issue with slow pagination
- #830 Charts: fixed incorrect histogram data in charts
- #829 Installation: added docs for Docker install on computer without internet access
- #828 Docker: enabled email configuration to work
- #827 SkinDose: made SkinDose results available in OpenREM and made alert triggering possible
- #826 Code quality: split views to make more manageable and testable
- #824 DICOM Networking: enabled declaration and testing of Orthanc Store SCP in Docker
- #822 Code quality: removed remaining future references
- #821 Code quality: fixed literal comparisons Docker was complaining about
- #820 Documentation: converted changes to use sphinx-issues
- #819 Removed colons from commands in documentation as they don't format correctly in PDF
- #818 Interface: refactored event number filtering
- #817 SkinDose: fixed PEP8 and Codacy issues for skinDose
- #816 Interface: fixed password change error
- #815 Interface: fixed patient name filtering
- #814 Deployment: automated deployment to dev.openrem.org and testing.openrem.org reintroduced
- #812 DICOM Networking: separated abort from timeout in move request failure message
- #808 Imports: caters for illegal use of mGy units in RDSR for dose at RP values

- #807 Exports: fixed errors in PHE fluoro export when values are None

- #805 DICOM Networking: fix errors on „association release" and „move complete"

- #803 Fixed problem with multiple ModalitiesInStudy entries in c-find response

- #800 Tasks: import and export tasks and DICOM queries and moves now listed with more information

- #799 DICOM Networking: workaround for stationnames > 16 characters

- #798 Exports: prevented error in export view if task_id is missing

- #797 Exports: fixed string/byte issues with csv exports

- #796 Exports: replaced file() with open() for Python 3.x compatibility

- #795 Exports: included acquisition name in PHE radiographic projection export

- #793 Installation: added Docker installation

- #791 Exports: prevented error when trying to export DX data that has no filter information

- #790 Python 3: remove basestring type

- #789 Python 3: Median function aggregation code simplified; works with Python 3.7

- #788 Tasks: Celery and RabbitMQ dropped, background task processing now managed within Python/OpenREM

- #787 Interface: fixed login error

- #786 Installation: increased Windows IIS timeouts in docs

- #777 Updated OpenREM to use pydicom 1.3

- #772 DICOM Networking: check for station name at series level or study, not both

- #764 Imports: extractor functions log to extractor log instead of default

- #744 Added overwrite mode to size import

- #678 Enabled import of PX modality panoramic exam data - they appear in the Radiographic section

- #664 Documentation: added sample config files to installation

- #657 Documentation: split local_settings.py example into Windows and Linux versions

- #530 Updated OpenREM to use pynetdicom 1.4

- #513 Internationalization: first translation strings added to documentation

- #512 Internationalization: first translation strings added to interface

- #457 Updated OpenREM to use Django 2.2

- #477 Charts: replaced HighCharts with open source Plotly library

- #437 Updated OpenREM to use django-filters v2

- #433 Import: Siemens Arcadis Varic dose reports are now imported

- #404 Ported OpenREM to Python 3

- #233 Charts: added charts of average CTDI and DLP over time

- #94 Nuclear medicine: added nuclear medicine SPECT and PET functionality including RRSDR imports

0.10.0 (2019-11-08)

- #785 Interface: added study level comments to rfdetail.html
- #784 Imports: added laterality under target region as per 2017 CP 1676 change
- #783 Interface: replaced static links by dynamic versions in rfdetail.html
- #782 Exports: fixed RF export issue with studies containing multiple modalities
- #781 Charts: fixed issue where charts were mis-labelled if „Case-insensitive categories" was un-checked
- #780 Interface: changed mammography accumulated laterality to use code_value rather than code_meaning
- #779 Installation: added restriction to django-qsstats-magic version
- #778 Imports: added summary field population tests, fixed CT RDSR Total DLP import error
- #776 Documentation: grammar and spelling correction for PHE exports
- #775 Exports, documentation: fixed units issue and minor docs issue for PHE DX export
- #774 Charts: fixed issue where charts were mis-labelled if „Plot a series per system" was unchecked
- #771 Interface: entire fluoro exam row now highlighted when dose alert exceeded
- #770 Imports: fix to allow non-conformant Opera Swing to import
- #769 Interface: modified to allow detail view display of Ziehm studies with missing summary data
- #768 Charts: study- and request-level charts now use study-level summary fields to improve performance
- #765 Imports: updated error catching to allow Philips BigBore 4DCT RDSR to import
- #763 Imports: corrected delta week fluoro study counting for dual plane modalities
- #762 Interface: fixed error when deleting dual plane radiography studies
- #761 Imports: fixed issue in high dose alert e-mail code where week_delta may be used before assignment
- #759 Database: added study level summary fields and migration function
- #758 Configuration: corrected issues with location of js_reverse static files
- #750 Exports: added export tailored to the 2019 PHE DX dose survey
- #746 Imports: enabled import of GE Elite Mini View C-arm RDSR with no template declaration
- #181 Imports: corrected import of grid information from RDSRs

0.9.1 (2019-05-16)

- #766 Documentation: updated the Windows Celery documentation to reflect changes required to shutdown Celery 3.1.25
- #755 Interface: fix more static URLs to allow virtual directory web server configurations
- #754 Documentation and install: updated docs and minimum version for collectstatic_js_reverse
- #753 Query-retrieve: removed patient age fields from study level C-FIND that were not used
- #752 Exports: fixed missing weight field in PHE CT 2019 export

- #749 Documentation: updated the Linux quick install docs

- #748 Charts: fixed error that caused blank charts if series per system was selected

- #747 Installation: changed minimum Python version for current version of Flower

- #743 Testing: added configuration to enable testing with default logging

- #742 Interface: sorting of task tables now works in Internet Explorer 11

- #740 Installation: fixed Celery version to avoid dependency on Django 1.11

- #739 Imports: fixed import errors for GE surgical fluoroscopy

- #738 Logging: added single_date query date to log, added tasks aborts to logs

- #737 Interface and exports: specify number of events and export to PHE 2019 CT survey specification

- #736 Query-retrieve: duplicate study level responses now removed from query

- #735 Imports: switched to more secure defusedxml for parsing XML in comments

- #734 Query-retrieve: handle illegal image level response with no instance number

- #732 Query-retrieve: added advanced option to workaround empty series issue

- #710 Interface: time-based columns in Celery and RabbitMQ tables now sorted correctly

- #404 Code quality: changes to lead toward Python 3 compliance

0.9.0 (2019-03-06)

- #733 Documentation: post-release fixes for 0.9.0 docs

- #731 Imports: fixed another issue with display names on upgrade to 0.9

- #729 Interface: replaced hard coded URLs in displaynameview.html and review_failed_imports.html with url names

- #727 Imports: fixed issue with display names on upgrade to 0.9

- #726 Documentation: updated to include the new task management function

- #725 Charts: added fluoroscopy charts of DAP and frequency per requested procedure

- #723 Task management: fixed issue with latest version of kombu and amqp on Windows

- #722 Interface: dual-plane DX studies are now displayed without error in filtered list and study detail page

- #721 Documentation: removed Django Debug Toolbar from default install and documented how to install and use it

- #720 Interface: fixed small overlap between skin dose map and irradiation type table

- #719 Interface: fixed hardcoded link in template rffiltered.html

- #717 Query-retrieve: fixed problem where an error was thrown if association is None

- #716 Task manager: removed assumption of queue name from RabbitMQ management interface

- #714 Documentation: add missing documentation about changing STATIC_URL if serving Open-REM in a virtual directory

- #711 Query-retrieve: fixed problem for zero image series when using -toshiba flag

- #710 Interface: Celery and RabbitMQ tables can now be sorted by clicking on column headings

- #709 Query-retrieve: corrected query logic for multiple modalities using #627 Modality tag at study level fix
- #708 Query-retrieve: fixed problem for empty Series Number
- #707 Interface: fixed issue where sigdig returned an error if it was passed an empty string
- #706 Exports: fixed problem where filters were not respected for radiographic exports
- #705 Task manager: added Flower to install and integrated to interface
- #704 Imports: caters for illegal use of dGy.cm2 units in RDSR for DAP values
- #703 Interface: fixed URL lookup error for failed imports on homepage
- #702 Query-retrieve: fixed URLs in DICOM javascript files to allow virtual-directories
- #701 Interface: made the fluoroscopy exposure detail table sortable by clicking on headers
- #698 Imports: allow for incorrect case in Procedure reported tag in RDSR
- #697 Testing: added tests for fluoroscopy high dose alerts (single-plane systems)
- #696 Interface: fixed broken Delete Studies and Entry button
- #695 Imports: added missing name attribute for size_abort url
- #694 Query-retrieve: added extensive logging and summary to interface
- #693 Interface: fixed display of numbers with significant places settings and comma localisation
- #691 Interface: fixed URL lookup error for Display Names page
- #690 Interface: added workload stats user option entry back into config menu
- #689 Interface: fixed URL lookup error for DICOM summary page
- #688 Interface: Add possibility to apply known display name based on Device Observer UID (default: disabled)
- #685 Charts: fixed link code that would otherwise cause DLP per acquisition protocol chart histogram links to fail
- #683 Installation: added VIRTUAL_DIRECTORY to the settings file to avoid updating local_settings file on upgrade
- #682 Charts: fixed problem where links from histogram bars didn't filter correctly when case-insensitive categories selected
- #681 Imports: modified RDSR import to work with Varian RDSRs
- #679 Interface: added ability to filter CT studies on acquisition type
- #677 Interface: added additional filter materials to convert to abbreviations
- #676 Imports: improved error handling on patient size imports
- #675 Exports: improved resilience when export includes malformed studies
- #674 Documentation: amended zip command in example Orthanc configuration to work with Linux and Windows
- #673 Imports: handle empty NumericValues and workaround for incorrect Philips Azurion AcquisitionDeviceType
- #672 Documentation: improve and extend linux one-page install
- #670 Imports: handle illegal multi-value number in Toshiba RDSR with vHP

- #668 Code quality: library import and blank space cleanup

- #667 Web server: enable OpenREM to be hosted from a non-root folder/virtual-directory

- #666 Query-retrieve: handle non-return of ModalitiesInStudy correctly

- #665 Interface: added fluoroscopy high dose highlighting and e-mail alerts

- #662 Administration: added facility to list and purge RabbitMQ queues

- #659 Interface: made the latest study field in summary tables on the home page sort correctly

- #658 Interface: added display of workload stats in home page modality tables

- #637 Administration: added facility to list and purge RabbitMQ queues

- #554 Query-retrieve: added time as matching argument for command line use

- #461 Web server: enable OpenREM to be hosted from a non-root folder/virtual-directory (via #667)

- #479 Administration: added facility to list and delete failed import studies

- #349 Task management: fixed issue with Windows tasks not being killed on request

0.8.1 (2018-09-16)

- #663 Interface: updated column headings on home page

- #660 Documentation: corrected and improved Linux one-page install

- #659 Interface: made the summary tables on the home page sortable by clicking on headers

- #656 Install: pegged django-debug-toolbar to 1.9.1 until Django is upgraded

- #654 Documentation: supplemented the Orthanc Lua file config option docs

- #653 Docs: clarified notes to get link to Orthanc Lua file correct on release

- #652 Documentation: added docs showing Celery daemonisation in Linux

- #651 Documentation: added one-page full setup Ubuntu 18.04 install instructions

- #650 Documentation: modified quick install virtualenv docs

- #649 Documentation: instructions for updating hosts file for Ubuntu and RabbitMQ

- #648 Documentation: clarified Toshiba options when not required

- #647 Documentation: updated link to pixelmed

- #646 Modified Celery import to avoid name clash in some circumstances

- #645 Imports: prevent import failure when text is used in filter thickness field in DX image

- #644 Exports: fixed error in exporting non-ASCII CT protocol acquisition names

- #643 Installation: updated docs to make use of pip binaries for Postgres connector and numpy, Windows and Linux

- #642 Skin dose maps: added catch for error when there are no events in the study

- #641 Exports: mammography exports from filtered pages sorted by AGD no longer result in duplicate studies

- #640 Exports: error in filter listing for NHSBSP csv exports corrected

- #639 Charts: fixed problem where a blank category name may not be displayed correctly

- #638 Skin dose maps: added a link to download data for stand-alone openSkin even when map displayed

- #627 DICOM Networking: implemented workaround for query „bug" in Impax 6.6

- #606 Interface: Made it possible for the user to change his/her password

0.8.0 (2018-06-11)

- #635 Documentation: added Orthanc as preferred third party DICOM Store service

- #634 Documentation: updated docs for import and query-retrieve duplicates processing

- #633 Charts: fixed issue where charts failed if bar chart series name was null

- #632 DICOM: move requests for queries that don't exist now fail gracefully

- #631 Skin dose maps: bug fixed that prevented message from displaying on screen when skin dose map cannot be calculated

- #630 Documentation: improved installation instructions

- #628 Imports: fixed code for importing when there are duplicate DX or MG studies in the database

- #626 DICOM: isolated the generate modalities in study function and added testing

- #625 Imports: now using event level UIDs to process continued, cumulative and duplicate RDSRs

- #624 Charts: removed filter link on number of events histogram as it was not functioning correctly

- #623 Imports: changed name of Toshiba image based extractor routine

- #621 Documentation: reversed install order of openrem and pynetdicom due to new pydicom release

- #619 Documentation: added workaround for outdated dictionary issues

- #618 DICOM: fixed image level query that prevented RDSRs from being found

- #617 Imports: fixed issue with multi study exams crashing the Toshiba extractor

- #616 Documentation: added information for pip download -d

- #615 Exports: added Target Exposure Index and Deviation Index to radiographic exports

- #614 Exports: handle error when study is deleted during sheet creation for exports

- #613 Imports: fixed dual modality type imports after ‚dual' designation from ref #580

- #612 Imports: prevented crash when RDSR was imported with AcquisitionProtocol sequence with no TextValue

- #610 DICOM: query-retrieve changed to work for duplicate RDSRs, ref #114

- #609 Interface: fixed the feature that toggles the selection when clicking anywhere on a display name table row

- #608 Interface: fixed the broken sorting of display name table

- #603 Interface: fixed JavaScript error if there are any None values in fluoro detail irradiation type table

- #602 Skin dose maps: fixed error when there are multiple kVp values for a single irradiation event

- #599 Installation: postgres instructions now include note about differing security choices

- #597 Skin dose maps: documented that using a production webserver the default timeout value must be increased

- #596 Documentation: added docs for using Gunicorn and NGINX on linux

- #594 Display: corrected display of dual-plane DAP and RP dose in RF filtered view

- #593 Imports: properly handles MultiValue filter material tags and permits aluminium spelling

- #592 Documentation: added docs for using IIS on Windows

- #589 Exports: now handles zero studies and studies deleted during exports sensibly

- #587 Documentation: added instructions for Linux users to rotate logs

- #586 Documentation: updated exports and detailed how pulse level data is exported

- #585 Documentation: added information about multiple cumulative RDSRs

- #584 Import, Interface, Export: RDSR with pulse level data now function

- #583 Documentation: added information about dual mode modalities and deleting all from an X-ray unit

- #582 Celery: updated results backend as amqp deprecated and slow

- #581 Import scripts: interpreter line now always first, functions imported specifically

- #580 Imports and Interface: one modality creating both DX and RF can now be handled appropriately

- #579 Imports: dummy values for Toshiba CT import function now in settings.py, log file config in docs

- #578 Exports: fixed NHSBSP export that was excluding RDSR imported Hologic studies

- #575 Exports: export page now updates using AJAX and has a select all button

- #573 Exports: corrected and clarified exposure time and duration units, added number of pulses

- #572 Interface: homepage now populates as AJAX to increase responsiveness

- #570 Charts: simplified chart function code

- #569 Charts: fixed frequency issue with mean averages selected

- #568 Imports: missing DICOM date-time no longer causes an error

- #567 Celery: fixed dual-namespace imports of tasks

- #566 Interface: correctly show „assumed patient mass" in case of set value of zero

- #565 Interface: correctly handle dose area product with zero value

- #564 Skin dose maps: text information on skin dose maps now embedded when saving the 2d or 3d map as a graphic

- #562 Skin dose maps: error message on calculation failure now more explicit

- #561 Imports: patient orientation modifier now correctly extracted from RDSR

- #560 Exports: added study level comments

- #559 Interface: date pickers inconsistent start day fixed

- #558 Skin dose maps: set defaults instead of crashing if kV, dose, table or tube/detector position are missing

- #557 Skin dose maps: improved construction of patient orientation code

- #556 Exports: DX exports where TotalNumberOfRadiographicFrames is not populated now export

- #552 Documentation: documented extractor for older Toshiba CT scanners

- #551 Documentation: added procedure for opening csv files in Excel with non-ASCII characters

- #550 Documentation: added a note to describe exposure time and duration for fluoroscopy studies

- #549 Documentation: added procedure for fixing laterality on Hologic studies, ref #411

- #547 Interface: improved handling of available time information for fluoro studies

- #546 Query Retrieve: added flag and functionality to query for Toshiba images

- #544 Interface: added procedure, requested procedure to summary listings and details and filtering

- #543 Interface: added drop-down box to choose how many studies are displayed on filtered pages

- #542 Interface: added display name to all detailed html pages

- #541 Documentation: updated for celery on Windows

- #540 Documentation: updated for current skinDose functionality

- #539 Documentation: updated chart document to include series toggle buttons

- #537 Charts: hide series function added

- #536 Code quality: reduced javascript duplication and collected file groups into subfolders

- #535 Interface: fixed problem where category names that included a plus symbol caused filtering and chart issues

- #534 Interface: chart drilldown reported as not working - was actually due to a user's database migrations

- #533 Query Retrieve: Reduced number of simultaneous associations to one, reused for everything

- #532 DICOM: documented how to work-around missing encoding charsets due to old pydicom

- #529 Charts: added CT charts of number of irradiation events per study description and requested procedure

- #528 Query Retrieve: reduced number of simultaneous associations to one, reused for everything

- #526 Code quality: addressed some of the code quality/style issues raised by *Codacy*

- #525 Importing: improved mammo import by checking compression force before converting to float

- #524 Importing: improved mammo import by checking anode exists before converting to DICOM terms

- #523 Importing: changed mammo import to use del_no_match instead of del_mg_im if not mammo

- #522 Documentation: made it clearer on offline-install docs that version numbers will change

- #521 Testing: added tests for dual source CT imports

- #520 Imports: removed XML styling from Philips legacy CT comment creation

- #519 Skin dose maps: fixed black on black text issue

- #518 Importing: fixed imports where CT Target Region isn't specified

- #517 Interface: operator name is now displayed on the detail page for each modality, along with physician for CT and fluoro

- #516 Imports: MultiValue person names are now stored as a decoded string, not a list

- #511 Testing: develop and other branches can now be deployed to dev.openrem.org and testing.openrem.org automatically

- #510 Imports: ‚not-patient-indicators' can now be configured in the interface

- #509 Skin dose maps: now recalculated on view if recorded height or weight has changed since last calculation

- #508 Testing: DX sample files are now tested

- #507 Interface: Mammo now filterable by study description, procedure, requested procedure and acquisition protocol

- #506 Documentation: updated query-retrieve docs

- #505 Charts: n is now displayed on charts

- #504 Charts: Fixed issue with null values

- #503 Internationalisation: more robust decoding and use of unicode throughout

- #502 Testing: tests now work with SQLite3 and PostgreSQL databases

- #501 Imports: Changed field type for CodeValue from 16 chars to text, allows for illegal long values

- #500 Imports: Philips SC Dose Info with missing time stamps now import

- #499 Imports: Now aborts gracefully with error log if no template in RDSR

- #498 Exports: Missing units added to header fields

- #497 Interface: Detailed fluoro study view: added irradiation type, pulse rate, dose to ref. point, secondary angle, total DAP and ref. point dose from each irradition type

- #495 Charts: Reduced time taken to render scatter plots with multiple series

- #494 Charts: Charts now ignore blank and zero-value data when calculating mean, median and number of events

- #493 Charts: Added user option to made chart categories all lower case

- #492 Exports: Each view is now unique for NHSBSP mammo exports as required by the NCCPM database

- #491 Imports, Interface and Exports: CT Dose Check alerts and notifications are now extracted, displayed and exported

- #490 Exports: Response object included for messages - removed as now asynchronous

- #489 Exports: NHSBSP mammo exports deals with all views, excludes biopsies and specimens

- #488 Exports: All exports now include study time

- #487 Imports: CT RDSR now imports ‚procedure context' correctly

- #486 Imports: CT RDSR now imports ‚NameOfPhysiciansReadingStudy' correctly

- #485 Imports: CT RDSR now imports ‚target region' correctly

- #484 Exports and Interface: Exports and interface page views are now more efficient and (much) faster

- #482 Imports: DX extractor now extracts acquisition protocol, requested procedure name and study name for Fuji Go mobile; extracts acquisition protocol for Toshiba Radrex equipment; extracts requested procedure name from Carestream DRX-Revolution mobiles

- #480 Imports: Code and instructions to create and import an RDSR from Toshiba CT dose summary images and studies

- #476 Imports: Mixed latin-1 and UTF8 characters now imported, but need to be handled better if possible

- #475 Query Retrieve: Made -sr a stand-alone option - it has a very niche use-case!

- #474 Logging: Changing to DEBUG logging level in `local_settings.py` will now be respected
- #473 Query Retrieve: Added tests
- #472 Query Retrieve: Overhauled the query retrieve routines
- #471 Internationalisation: added configuration and docs to set the timezone
- #470 Query Retrieve: Optimised CT filtering
- #468 Query Retrieve: Station names can now be used for filtering if returned
- #467 Testing: Added tests for mammography RDSR imports
- #466 Query Retrieve: RDSR now retrieved in preference to images for MG and DX/CR
- #465 Added newer SSDE and water equivalent diameter fields to database
- #464 Imports: DX RDSR now imported properly
- #463 Imports: Properly checks that Enhanced SR are GE dose reports before importing
- #460 Interface: Display names table now sortable
- #458 Exports: Filter thicknesses are rounded to max 4 significant figures on export
- #454 Exports: Mean filter thickness now reported in exports
- #453 Imports: DX with min filter thickness greater than max have values switched on import
- #452 Exports: Added CTDIw phantom size to CT exports
- #451 Skin dose maps: fixed issue with filters being referenced before being defined
- #450 Imports: DX imports with filter thickness of 0.00 are now recorded as such
- #449 Exports: Fixed a bug that prevented fluoro exports if protocol names had non-ASCII characters
- #448 Documentation: Added a diagram showing the relationship between the OpenREM system components
- #447 Imports: Modified rdsr and ctdetail template to import and display data from Pixelmed generated Toshiba RDSR
- #446 Import: Extract additional Philips private information for Allura Xper systems, create workaround for missing end angles for rotational acquisitions
- #445 Interface: Added function for user to determine between DX and fluoro for ambiguous modalities
- #444 Imports: DX systems that submit RDSRs that look like fluoro can now be reclassified using #445
- #443 Exports: Accession number and ID are now exported to XLSX as text. Thanks to @LuukO
- #442 Exports: Fixed RF exports with multiple filters, added tests. Thanks to @LuukO
- #441 Charts: Fixed a bug that broke chart links containing non-ASCII characters
- #440 Charts: Fixed a bug in sorting.js so that undefined strings are handled correctly
- #439 Charts: Added controls for plotting a series per system and calculation histogram data to each filtered view
- #438 Skin dose maps: skin dose maps successfully calculated from existing studies; indication of assumed or extracted data shown
- #434 Internationalisation: added passing char_set throughout the extractor functions (since largely made redundant again!)

- #432 Imports: RDSR import function now looks in comment field for *patient_table_relationship* data

- #431 Imports: fixed DX imports with MultiValue filter values (Cu+Al) again!

- #430 Exports: fixed DX exports with multiple filters again, added tests

- #429 Charts: added new mammo scatter plots. Thanks to @rijkhorst

- #427 Testing: added a large number of tests that are automatically run on commit to bitbucket

- #414 Reduced use of JavaScript global variables and improved JavaScript objects

- #411 Imports: fixed laterality and accumulated AGD failure for Hologic DBT proprietary projection images

- #323 Documentation: code autodocumentation largely now working again

- #318 Database management: Display names view can be used to review and delete all studies from one source

- #114 Imports: Subsequent RDSRs of the same study will now replace existing study in database

- #61 Skin dose maps: These have been re-enabled, and currently work for Siemens systems

0.7.4 (2016-10-17)

- #436 Install: temporary fix blocking django-filter latest version that breaks OpenREM

- #431 Imports: fixed DX imports with MultiValue filter values (Cu+Al)

- #430 Exports: fixed DX exports with multiple filters (Cu + Al)

0.7.3 (2016-08-30)

- #426 Charts: added css so that wide chart data tables are displayed above the filter form div

- #425 Exports: fixed error with non-ASCII characters being exported to csv

- #424 Charts: fixed error where png or svg export of chart would show incorrect x-axis labels

- #423 Charts: fixed error where some chart plotting options were not updated after being changed by the user

- #422 Charts: added a button below each chart to toggle the display of the data table

- #421 Charts: fixed error where only some scatter plot data was being exported to csv or xls files

- #420 Charts: fixed error where frequency pie charts were only showing data from the first system

- #419 Interface: fixed error where „Cancel" was ignored when deleting study in Firefox browser

- #418 Exports: fixed error when exporting fluoroscopy study with missing xray_filter_material

- #416 Charts: improved efficiency of JavaScript

- #415 Database: migration for 0.6 upgraded installs to fix acquisition_device_type failures

- #413 Documentation: removed erroneous reference to store queue in stop celery command

- #410 Charts: fixed display of bar charts containing only one data point

- #408 Charts: Increased number of items that can be shown on some Highcharts plots

- #407 Fixed issue where skin dose map data was not being calculated on import

- #406 Replaced Math.log10 JavaScript function with alternative function to fix IE11 skin dose map error
- #405 Altered multi-line cell links in filtered pages so they work with IE8

0.7.1 (2016-06-10)

- #403 Now deals with PersonName fields with latin-1 extended characters correctly
- #402 Skin dose map data pickle files saved using gzip compression to save space
- #401 Updated skin dose map documentation to say it won't be in this release
- #400 Strings are encoded as UTF-8 before being hashed to prevent errors with non-ASCII characters
- #399 Migration file brought up to date for 0.6 to 0.7 upgrades
- #398 Skin exposure maps are now stored in folders (feature postponed for future release)
- #397 Skin exposure maps no longer available until orientation errors are fixed
- #396 Charts: zooming on bar charts of average value vs. category now works
- #395 Docs: offline Windows install instructions created, plus offline upgrade instructions
- #394 Charts: made charts resize to fit containing div when browser is resized
- #392 Charts: normalised histogram tooltip now correctly reports frequency
- #391 Basic troubleshooting is now documented
- #390 Charts: mammography and fluoroscopy charts added
- #389 Charts: series without a name are now plotted under the name of *Blank* rather than not being plotted at all
- #387 Added laterality to mammography exports
- #385 Fixed issue with non-ASCII letters in RDSR sequence TextValue fields
- #384 Fluoro exports for OpenSkin only consider copper filters now
- #383 Refreshed settings.py to django 1.8 including updating template settings and TEMPLA-TE_CONTEXT_PROCESSORS
- #380 Tube current now extracted from Siemens Intevo RDSR despite non-conformance
- #379 Exposure time now populated for fluoro if not supplied by RDSR
- #378 The display name of multiple systems can now be updated together using a single new name
- #376 Corrected an ill-advised model change
- #374 CTDIw phantom size now displayed in CT detail view
- #373 Charts in some releases used GT rather than greater than or equal to for start date, now fixed
- #372 Mammography studies now record an accumulated AGD per breast. Existing joint accumulated AGD values won't be changed. Ordering by Accumulated AGD now creates an entry per accumulated AGD, one per breast
- #371 Mammo RDSR generates average mA where not recorded, mammo image populates mA
- #370 Added study description to mammography export
- #369 Bi-plane fluoroscopy studies now export correctly
- #368 Mammo RDSR now imports correctly

- #365 Tube filtration is now displayed in the RF detail view
- #364 Philips Allura fluorscopy RDSRs now import correctly
- #362 Display of RF where bi-plane RDSRs have been imported no longer crash the interface
- #360 Charts: saving data from average data charts as csv or xls now includes frequency values
- #359 Added missing ,y' to query retrieve command line help
- #358 Charts: chart sorting links and instructions now hidden when viewing histograms
- #357 Charts: button to return from histogram now displays the name of the main chart
- #356 Charts: histogram normalise button appears for all appropriate charts
- #355 Charts: sorting now works as expected for plots with a series per system
- #352 Fixed CT xlsx exports that had complete study data in each series protocol sheet (from earlier beta)
- #351 Charts: simplified chart JavaScript and Python code
- #350 DICOM networking documented for use with 3rd party store and advanced use with native
- #348 Study delete confirmation page now displays total DAP for DX or CR radiographic studies
- #346 Charts: exporting a chart as an image no longer requires an internet connection
- #345 CSV size imports in cm are now stored as m in the database. Interface display of size corrected.
- #343 Charts: user can now specify number of histogram bins in the range of 2 to 40
- #342 Charts: improved the colours used for plotting chart data
- #340 Fixed store failure to save due to illegal values in Philips private tags, improved exception code
- #339 Improved extraction of requested procedure information for radiographic studies
- #338 Fix Kodak illegally using comma in filter thickness values
- #335 DICOM Store keep_alive and echo_scu functions now log correctly
- #334 Fixed issue with tasks needing to be explicitly named
- #333 Fixed StoreSCP not starting in beta 11 error
- #332 Charts: some charts can now be plotted with a series per x-ray system
- #331 Keep_alive tasks are now discarded if not executed, so don't pile up
- #329 All existing logging is now done via the same log files
- #328 Store SCP no longer uses Celery tasks
- #327 Celery workers now only take one task at a time
- #325 Charts: switching charts off now leaves the user on the same page, rather than going to the home page
- #324 Charts: forced chart tooltip background to be opaque to make reading the text easier
- #320 The week now begins on Monday rather than Sunday on date form fields
- #316 Query retrieve function can now exclude and include based on strings entered
- #315 Charts: made size of exported chart graphics follow the browser window size
- #314 One version number declaration now used for distribute, docs and interface

- #313 Replaced non-working function with code to extract SeriesDescription etc in query response message
- #312 Display names are now grouped by modality
- #311 Queries are deleted from database after a successful C-Move
- #310 Series level QR feedback now presented. Any further would require improvements in pynetdicom
- #309 StoreSCP now deals safely with incoming files with additional transfer syntax tag
- #308 Secondary capture images that don't have the manufacturer field no longer crash the StoreSCP function
- #306 Charts: added a button to each chart to toggle full-screen display
- #305 Added links to documentation throughout the web interface
- #304 Date of birth is now included in all exports that have either patient name or ID included
- #303 Fixed a typo in 0.6.0 documents relating to the storescp command
- #302 Improved handling of Philips Dose Info objects when series information sequence has UN value representation
- #301 Charts: fixed bug that could stop average kVp and mAs radiographic plots from working
- #300 Calling AE Title for Query Retrieve SCU is now configured not hardcoded
- #299 Hash of MultiValued DICOM elements now works
- #298 Added ordering by accumulated AGD for mammographic studies
- #297 Fixed ordering by Total DAP for radiographic studies
- #296 StoreSCP now logs an error message and continues if incoming file has problems
- #295 Charts: fixed bug that arose on non-PostgreSQL databases
- #294 Harmonised time display between filter list and detail view, both to HH:mm
- #292 Added keep-alive and auto-start to DICOM stores
- #291 Charts: fixed issue with CTDI and DLP not showing correct drilldown data
- #290 Added new tables and fields to migration file, uses #288 and median code from #241
- #289 Crispy forms added into the requires file
- #288 Added device name hashes to migration file
- #286 Increased granularity of permission groups
- #285 Tidied up Options and Admin menus
- #284 Fixed DICOM Query that looped if SCP respected ModalitiesInStudy
- #282 Missing javascript file required for IE8 and below added
- #281 Added check to import function to prevent extract failure
- #280 Fixed typo in mammography export
- #279 Charts: Fixed issue with median CTDI series from appearing
- #278 Charts: Fixed javascript namespace pollution that caused links to fail
- #277 Overhaul of acquisition level filters to get tooltip generated filters to follow through to export

- #276 Unique fields cannot have unlimited length in MySQL - replaced with hash
- #274 Charts: Fixed legend display issue
- #273 Charts: Added plots of average kVp and mAs over time for DX
- #272 Tweak to display of exam description for DX
- #271 Fixed DX import failure where `AcquisitionDate` or `AcquisitionTime` are `None`
- #270 Django 1.8 Admin site has a ‚view site' link. Pointed it back to OpenREM
- #268 Improved population of procedure_code_meaning for DX imports
- #266 DICOM C-Store script added back in - largely redundant with web interface
- #265 DICOM Store and Query Retrieve services documented
- #263 Settings for keeping or deleting files once processed moved to database and web interface
- #262 Dealt with issue where two exposures from the same study would race on import
- #260 Fixed issue where import and export jobs would get stuck behind StoreSCP task in queue
- #259 Link to manage users added to Admin menu
- #258 Fixed DX import error where manufacturer or model name was not provided
- #257 Documentation update
- #256 Fixed errors with non-ASCII characters in imports and query-retrieve
- #255 Charts: Small y-axis values on histograms are more visible when viewing full-screen
- #254 Charts: Simplified chart data processing in the templates
- #253 Charts: AJAX used to make pages responsive with large datasets when charts enabled
- #252 Fixed duplicate entries in DX filtered data for studies with multiple exposures
- #248 Charts: can now be ordered by frequency or alphabetically
- #247 Fixed incorrect reference to manufacturer_model_name
- #246 Charts: Added median data for PostgreSQL users
- #245 Fixed error in csv DX export
- #244 Fixed issue where scripts wouldn't function after upgrade to Django 1.8
- #243 Added distance related data to DX exports
- #242 Distance source to patient now extracted from DX images
- #241 Charts: Median values can be plotted for PostgreSQL users
- #240 Charts: Improved DAP over time calculations
- #239 Configurable equipment names to fix multiple sources with the same station name
- #237 Charts: Tidied up plot data calculations in `views.py`
- #235 Added patient sex to each of the exports
- #234 Charts: Fixed error with datetime combine
- #232 Charts: on or off displayed on the home page
- #231 Charts: made links from requested procedure frequency plot respect the other filters
- #230 Fixed error in OperatorsName field in DICOM extraction

- #229 Charts: Added chart of DLP per requested procedure
- #223 Charts: speed improvement for weekday charts
- #217 Charts: Further code optimisation to speed up calculation time
- #207 DICOM QR SCU now available from web interface
- #206 DICOM Store SCP configuration now available from web interface
- #183 Added options to store patient name and ID, and options to hash name, ID and accession number
- #171 Root URL now resolves so `/openrem` is not necessary
- #151 Suspected non-patient studies can now be filtered out
- #135 GE Senographe DS now correctly records compression force in Newtons for new imports
- #120 Improved testing of data existing for exports
- #118 Upgraded to Django 1.8
- #70 User is returned to the filtered view after deleting a study
- #61 Skin dose maps for fluoroscopy systems can now be calculated and displayed

0.6.2 (2016-01-27)

- #347 Django-filter v0.12 has minimum Django version of 1.8, fixed OpenREM 0.6.2 to max django-filter 0.11
- #341 Changed references to the OpenSkin repository for 0.6 series.

0.6.1 (2015-10-30)

- #303 Corrected name of Store SCP command in docs

0.6.0 (2015-05-14)

- #227 Fixed import of RDSRs from Toshiba Cath Labs
- #226 Charts: Updated Highcharts code and partially fixed issues with CTDIvol and DLP combined chart
- #225 Charts: Added link from mAs and kVp histograms to associated data
- #224 Charts: Added link from CTDIvol histograms to associated data
- #221 Charts: Fixed issue where filters at acquisition event level were not adequately restricting the chart data
- #219 Charts: Fixed issue where some charts showed data beyond the current filter
- #217 Charts: Code optimised to speed up calculation time
- #216 Fixed typo that prevented import of RSDR when DICOM store settings not present
- #215 Charts: Fixed x-axis labels for mean dose over time charts
- #214 Charts: Improved consistency of axis labels
- #213 Fixed admin menu not working
- #212 Charts: Created off-switch for charts

- #210 OpenSkin exports documented

- #209 Charts: Fixed server error when CT plots switched off and filter form submitted

- #208 Charts: Fixed blank chart plotting options when clicking on histogram tooltip link

- #205 Charts: Fixed issue of histogram tooltip links to data not working

- #204 Charts: Fixed issue of not being able to export with the charts features added

- #203 Charts: Fixed display of HTML in plots issue

- #202 Charts: Added mean CTDIvol to charts

- #200 Charts: Now exclude Philips Ingenuity SPRs from plots

- #196 Added comments and entrance exposure data to DX export

- #195 Fixed error with no users on fresh install

- #194 Added more robust extraction of series description from DX

- #193 Charts: Fixed reset of filters when moving between pages

- #192 Created RF export for OpenSkin

- #191 Charts: Factored out the javascript from the filtered.html files

- #190 Charts: Added time period configuration to dose over time plots

- #189 Charts: Fixed plotting of mean doses over time when frequency not plotted

- #187 Charts: Merged the charts work into the main develop branch

- #186 Fixed duplicate data in DX exports

- #179 Charts: Added kVp and mAs plots for DX

- #177 Charts: Fixed issue with date ranges for DX mean dose over time charts

- #176 Charts: Added link to filtered dataset from mean dose over time charts

- #175 Charts: Allowed configuration of the time period for mean dose trend charts to improve performance

- #174 Charts: Fixed number of decimal places for mean DLP values

- #173 Charts: Fixed plot of mean DLP over time y-axis issue

- #170 Charts: Added plot of mean dose over time

- #169 Charts: Improved chart colours

- #157 Charts: Added chart showing number of studies per day of the week, then hour in the day

- #156 Charts: Fixed issue with some protocols not being displayed

- #155 Charts: Added chart showing relative frequency of protocols and study types

- #140 Charts: Added configuration options

- #139 Charts: Link to filtered dataset from histogram chart

- #138 Charts: Number of datapoints displayed on tooltip

- #135 Mammography compression force now only divides by 10 if model contains *senograph ds*
  Change in behaviour

- #133 Documented installation of NumPy, initially for charts

- #41 Preview of DICOM Store SCP now available

- #20 Modality sections are now suppressed until populated

0.5.1 (2015-03-12)

- #184 Documentation for 0.5.1

- #180 Rename all reverse lookups as a result of #62

- #178 Added documentation regarding backing up and restoring PostgreSQL OpenREM databases

- #172 Revert all changes made to database so #62 could take place first

- #165 Extract height and weight from DX, height from RDSR, all if available

- #161 Views and exports now look for accumulated data in the right table after changes in #159 and #160

- #160 Created the data migration to move all the DX accumulated data from TID 10004 to TID 10007

- #159 Modified the DX import to populate TID 10007 rather than TID 10004. RDSR RF already populates both

- #158 Demo website created by DJ Platten: http://demo.openrem.org/openrem

- #154 Various decimal fields are defined with too few decimal places - all have now been extended.

- #153 Changed home page and modality pages to have whole row clickable and highlighted

- #150 DJ Platten has added Conquest configuration information

- #137 Carestream DX multiple filter thickness values in a DS VR now extracted correctly

- #113 Fixed and improved recording of grid information for mammo and DX and RDSR import routines

- #62 Refactored all model names to be less than 39 characters and be in CamelCase to allow database migrations and to come into line with PEP 8 naming conventions for classes.

0.5.0 (2014-11-19)

- Pull request from DJ Platten: Improved display of DX data and improved export of DX data

- #132 Fixed mammo export error that slipped in before the first beta

- #130 Only creates ExposureInuAs from Exposure if Exposure exists now

- #128 Updated some non-core documentation that didn't have the new local_settings.py reference or the new openremproject folder name

- #127 DX IOD studies with image view populated failed to export due to lack of conversion to string

- #126 Documentation created for the radiographic functionality

- #125 Fixes issue where Hologic tomo projection objects were dropped as they have the same event time as the 2D element

- #123 Fixed issue where filters came through on export as lists rather than strings on some installs

- #122 Exports of RF data should now be more useful when exporting to xlsx. Will need refinement in the future

- #26 Extractors created for radiographic DICOM images. Contributed by DJ Platten

- #25 Views and templates added for radiographic exposures - either from RDSRs or from images - see #26. Contributed by DJ Platten

- #9 Import of *.dcm should now be available from Windows and Linux alike

0.4.3 (2014-10-01)

- #119 Fixed issue where Celery didn't work on Windows. Django project folder is now called openremproject instead of openrem

- #117 Added Windows line endings to patient size import logs

- #113 Fixed units spelling error in patient size import logs

- #112 File system errors during imports and exports are now handled properly with tasks listed in error states on the summary pages

- #111 Added abort function to patient size imports and study exports

- #110 Converted exports to use the FileField handling for storage and access, plus modified folder structure.

- #109 Added example `MEDIA_ROOT` path for Windows to the install docs

- #108 Documented ownership issues between the webserver and Celery

- #107 Documented process for upgrading to 0.4.2 before 0.4.3 for versions 0.3.9 or earlier

- #106 Added the duration of export time to the exports table. Also added template formatting tag to convert seconds to natural time

- #105 Fixed bug in Philips CT import where `decimal.Decimal` was not imported before being used in the age calculation

- #104 Added documentation for the additional study export functions as a result of using Celery tasks in task #19 as well as documentation for the code

- #103 Added documentation for using the web import of patient size information as well as the new code

- #102 Improved handling of attempts to process patient size files that have been deleted for when users go back in the browser after the process is finished

- #101 Set the security of the new patient size imports to prevent users below admin level from using it

- #100 Logging information for patient size imports was being written to the database - changed to write to file

- #99 Method for importing remapp from scripts and for setting the *DJANGO_SETTINGS_MODULE* made more robust so that it should work out of the box on Windows, debian derivatives and virtualenvs

- #98 Versions 0.4.0 to 0.4.2 had a settings.py.new file to avoid overwriting settings files on upgrades; renaming this file was missing from the installation documentation for new installs

- #97 Changed the name of the export views file from ajaxviews as ajax wasn't used in the end

- #96 Changed mammo and fluoro filters to use named fields to avoid needing to use the full database path

- #93 Set the security of the new exports to prevent users below export level from creating or downloading exports

- #92 Add *NHSBSP specific mammography csv export* from Jonathan Cole - with Celery

- #91 Added documentation for Celery and RabbitMQ

- #90 Added delete function for exports
- #89 Added the Exports navigation item to all templates, limited to export or admin users
- #88 Converted fluoroscopy objects to using the Celery task manager after starting with CT for #19
- #87 Converted mammography objects to using the Celery task manager after starting with CT for #19
- #86 Digital Breast Tomosynthesis systems have a projections object that for Hologic contains required dosimetry information
- #85 Fix for bug introduced in #75 where adaption of ptsize import for procedure import broke ptsize imports
- #74 ‚Time since last study' is now correct when daylight saving time kicks in
- #39 Debug mode now defaults to False
- #21 Height and weight data can now be imported through forms in the web interface
- #19 Exports are now sent to a task manager instead of locking up the web interface

Reopened issue

- #9 Issue tracking import using *.dcm style wildcards reopened as Windows `cmd.exe` shell doesn't do wildcard expansion, so this will need to be handled by OpenREM in a future version

0.4.2 (2014-04-15)

- #83 Fix for bug introduced in #73 that prevents the import scripts from working.

0.4.1 (2014-04-15)

- #82 Added instructions for adding users to the release notes

0.4.0 (2014-04-15)

Bemerkung:

- #64 includes **changes to the database schema and needs a user response** - see version 0.4.0 release notes
- #65 includes changes to the settings file which **require settings information to be copied** and files moved/renamed - see version 0.4.0 release notes

- #80 Added docs for installing Apache with auto-start on Windows Server 2012. Contributed by JA Cole
- #79 Updated README.rst instructions
- #78 Moved upgrade documentation into the release notes page
- #77 Removed docs builds from repository
- #76 Fixed crash if exporting from development environment

- #75 Fixed bug where requested procedure wasn't being captured on one modality
- #73 Made launch scripts and ptsizecsv2db more robust
- #72 Moved the secret key into the local documentation and added instructions to change it to release notes and install instructions
- #71 Added information about configuring users to the install documentation
- #69 Added documentation about the new delete study function
- #68 Now checks sequence code meaning and value exists before assigning them. Thanks to JA Cole
- #67 Added ‚Contributing authors' section of documentation
- #66 Added ‚Release notes' section of documentation, incuding this file
- #65 Added new `local_settings.py` file for database settings and other local settings
- #64 Fixed imports failing due to non-conforming strings that were too long
- #63 The mammography import code stored the date of birth unnecessarily. Also now gets decimal_age from age field if necessary
- #60 Removed extraneous colon from interface data field
- #18 Studies can now be deleted from the web interface with the correct login
- #16 Added user authentication with different levels of access
- #9 Enable import of `*.dcm`

0.3.9 (2014-03-08)

Bemerkung: #51 includes changes to the database schema – make sure South is in use before upgrading. See https://docs.openrem.org/page/upgrade.html

---

- #59 CSS stylesheet referenced particular fonts that are not in the distribution – references removed
- #58 Export to xlsx more robust - limitation of 31 characters for sheet names now enforced
- #57 Modified the docs slightly to include notice to convert to South before upgrading
- #56 Corrected the mammography target and filter options added for issue #44
- #53 Dates can now be selected from a date picker widget for filtering studies
- #52 Split the date field into two so either, both or neither can be specified
- #51 Remove import modifications from issue #28 and #43 now that exports are filtered in a better way after #48 and #49 changes.
- #50 No longer necessary to apply a filter before exporting – docs changed to reflect this
- #49 CSV exports changed to use the same filtering routine introduced for #48 to better handle missing attributes
- #48 New feature – can now filter by patient age. Improved export to xlsx to better handle missing attributes
- #47 Install was failing on pydicom – fixed upstream

0.3.8 (2014-03-05)

- – File layout modified to conform to norms
- #46 Updated documentation to reflect limited testing of mammo import on additional modalities
- #45 mam.py was missing the licence header - fixed
- #44 Added Tungsten, Silver and Aluminum to mammo target/filter strings to match – thanks to DJ Platten for strings
- #43 Mammography and Philips CT import and export now more robust for images with missing information such as accession number and collimated field size
- #42 Documentation updated to reflect #37
- #37 Studies now sort by time and date

0.3.7 (2014-02-25)

- #40 Restyled the filter section in the web interface and added a title to that section
- #38 Column titles tidied up in Excel exports
- #36 openrem_ptsizecsv output of log now depends on verbose flag
- #35 Numbers no longer stored as text in Excel exports

0.3.6 (2014-02-24)

- #34 Localised scripts that were on remote web servers in default Bootstrap code
- #33 Documentation now exists for adding data via csv file
- #24 Web interface has been upgraded to Bootstrap v3
- #5 Web interface and export function now have some documentation with screenshots

0.3.5-rc2 (2014-02-17)

- #32 Missing sys import bug prevented new patient size import from working

0.3.5 (2014-02-17)

- – Prettified this document!
- #31 Promoted patient size import from csv function to the scripts folder so it will install and can be called from the path
- #30 Improved patient size import from csv to allow for arbitary column titles and study instance UID in addition to accession number.
- #29 Corrected the docs URL in the readme

0.3.4-rc2 (2014-02-14)

- #28 XLSX export crashed if any of the filter fields were missing. Now fills on import with ‚None'
- #27 Use requested procedure description if requested procedure code description is missing

0.3.4 (2014-02-14)

- – General improvements and addition of logo to docs
- #23 Added Windows XP MySQL backup guide to docs
- #22 Added running Conquest as a Windows XP service to docs
- #15 Added version number and copyright information to xlsx exports
- #14 Added version number to the web interface
- #13 Improve the docs with respect to South database migrations

0.3.3-r2 (2014-02-04)

- #12 Added this version history
- #11 Documentation is no longer included in the tar.gz install file – see http://openrem.trfd.org instead

0.3.3 (2014-02-01)

Bemerkung: Installs of OpenREM earlier than 0.3.3 will break on upgrade if the scripts are called from other programs. For example openrem_rdsr is now called openrem_rdsr.py

---

- – Added warning of upgrade breaking existing installs to docs
- #10 Added .py suffix to the scripts to allow them to be executed on Windows (thanks to DJ Platten)
- #8 Removed superfluous ‚/' in base html file, harmless on linux, prevented Windows loading stylesheets (thanks to DJ Platten)
- #7 Added windows and linux path examples for test SQLite database creation
- #6 Corrected renaming of example files installation instruction (thanks to DJ Platten)
- #4 Added some text to the documentation relating to importing files to OpenREM
- #3 Corrected copyright notice in documentation

0.3.2 (2014-01-29)

- Initial version uploaded to bitbucket.org

12.2 Release notes and upgrade instructions

Each release comes with specific upgrade instructions, so please follow the links below for the appropriate version.

12.2.1 Version specific information

This release:

OpenREM Release Notes version 1.0.0

Headline changes

- Python 3

- Django 2.2

- Docker or direct install on Windows and Linux

- Celery, Flower and RabbitMQ removed from requirements

- Performing physician added to standard fluoroscopy exports (#840)

- Station name checked at series level only, option to check at study level only instead (#772)

Upgrade from 0.10.0

Review the *Installation* doc to find the upgrade options

Upgrade from an older version

Upgrade to OpenREM 0.10.0 from 0.7.3 or later

Previous releases:

OpenREM Release Notes version 0.10.0

Headline changes

- Database: new summary fields introduced to improve the responsiveness of the interface - requires additional migration step

- Imports: enabled import of GE Elite Mini View C-arm, Opera Swing R/F and Philips BigBore CT RDSRs that have issues

- Imports: updated event level laterality to import from new location after DICOM standard change proposal CP1676

- Interface: highlight row when dose alert exceeded

- Exports: added fluoroscopy and radiography exports tailored for UK PHE dose survey

- General: Lots of fixes to imports, interface, charts etc

Upgrade to current version

Upgrade to OpenREM 0.10.0 from 0.7.3 or later and then upgrade to 1.0.

Original upgrade instructions

For the original upgrade instructions, the last docs release to include them was 0.10.0-docs

OpenREM Release Notes version 0.9.1

Headline changes

- Imports: fixed imports for GE surgical flat panel c-arm with irregular value types and value meanings
- Interface: added feature to filter by specific number of exposure types – CT only
- Query-retrieve: new option to get SR series when PACS returns empty series level response
- Query-retrieve: handle illegal missing instance number in image level response
- Query-retrieve: improved logging
- Exports: added export to UK PHE 2019 CT survey format
- General documentation and interface improvements, bug fixes, and changes to prepare for Python 3

Upgrade to current version

Upgrade to OpenREM 0.10.0 from 0.7.3 or later and then upgrade to 1.0.

Original upgrade instructions

For the original upgrade instructions, the last docs release to include them was 0.10.0-docs

OpenREM Release Notes version 0.9.0

Headline changes

- Interface: added feature to display workload stats in the home page modality tables
- Interface: added *Fluroscopy high dose alerts* feature
- Interface: dual-plane DX studies can now be handled in summary list and study detail pages
- Interface: new option to set display name when unique fields change based on device observer UID in RDSR
- Charts: added fluoroscopy charts of DAP and frequency per requested procedure, fixed bugs in links for others
- Query-retrieve: handle non-return of ModalitiesInStudy correctly
- Query-retrieve: increased query logging and summary feedback
- Query-retrieve: use time range in search (command line only)
- Imports: fix for empty NumericValues in RDSR

- Imports: fix for Toshiba RDSR with incorrect multiple values in SD field for vHP

- Imports: fix for Philips Azurion RDSR with incorrect AcquisitionDeviceType

- Imports: fix for Varian RDSRs

- Exports: made more robust for exporting malformed studies, fixed filtering bugs

- Administration: automatic e-mail alerts sent when fluoroscopy studies exceed a dose alert level

- Administration: added facility to list and delete studies where the import failed

- Administration: added interface to RabbitMQ queues and Celery tasks

- Administration: short-term fix for task performance and control on Windows

- Documentation: further refinement of the linux one-page install

- Installation: *Running the OpenREM website in a virtual directory*

Upgrade to current version

Upgrade to OpenREM 0.10.0 from 0.7.3 or later and then upgrade to 1.0.

Original upgrade instructions

For the original upgrade instructions, the last docs release to include them was 0.10.0-docs

OpenREM Release Notes version 0.8.1

Headline changes

- Documentation: improved docs and added one-page complete install on Ubuntu instructions

- Install: temporary fix for dependency error

- Interface: added feature to allow users to change their own password

- Charts: fixed problem where a blank category name may not be displayed correctly

- Imports: reduced list of scanners that work with the legacy Toshiba CT extractor

- Imports: improved handling of non-conformant DX images with text in filter thickness fields

- Query-Retrieve: added non-standard option to work-around bug in Impax C-FIND SCP

- Exports: fixed bug in mammography NHSBSP exports that incorrectly reported the filter material in some circumstances

- Exports: fixed bug where sorting by AGD would cause duplicate entries for bilateral studies

- Exports: fixed another non-ASCII bug

Upgrade to current version

Upgrade to OpenREM 0.10.0 from 0.7.3 or later and then upgrade to 1.0.

Specific upgrade instructions

For the original upgrade instructions, the last docs release to include them was 0.10.0-docs

OpenREM Release Notes version 0.8.0

Headline changes

- This release has extensive automated testing for large parts of the codebase (for the first time)
- Code quality is much improved, reduced duplication, better documentation, many bugs fixed
- Imports: RDSR from a wider range of systems now import properly
- Imports: Better distinction and control over defining RDSR studies as RF or DX
- Imports: Code and instructions to generate and import RDSR from older Toshiba CT scanners
- Imports: DICOM Query-Retrieve functionality has been overhauled
- Imports: Duplicate checking improved to allow cumulative and continued study RDSRs to import properly
- Imports: indicators that a study is not a patient can now be configured in the web interface
- Imports, display and export: Better handling of non-ASCII characters
- Interface: More detailed, consistent and faster rendering of the data in the web interface
- Interface: Maps of fluoroscopy radiation exposure incident on a phantom (Siemens RDSRs only)
- Interface: More and better charts, including scatter plots for mammography
- Interface: Display names dialogue has been extended to allow administration of all studies from each source
- Exports: Much faster, and more consistent
- Documentation: Extensive user documentation improvements

Upgrade to current version

Upgrade to OpenREM 0.10.0 from 0.7.3 or later and then upgrade to 1.0.

Specific upgrade instructions

For the original upgrade instructions, the last docs release to include them was 0.10.0-docs

OpenREM Release Notes version 0.7.4

Headline changes

- Imports: DX images now import with multiple filters that are MultiValue as well as comma separated
- Exports: DX data now correctly exports to csv and xlsx if studies include multiple filters (eg Cu+Al)
- Install: New release of dependency django-filter breaks OpenREM. Pegged at previous version for now

Upgrade to current version

Upgrade to OpenREM 0.10.0 from 0.7.3 or later and then upgrade to 1.0.

Specific upgrade instructions

For the original upgrade instructions, the last docs release to include them was 0.10.0-docs

OpenREM Release Notes version 0.7.3

Headline changes

- Database: New migration file for upgrades from 0.6 series databases
- Charts: Fixed display and export errors, improved layout and increased the number of data points that can be plotted
- Interface: Fixed multi-line cells in tables so that the links work in IE8
- Interface: Fixed delete cancel button in firefox
- Exports: Fixed export of non-ASCII characters to csv file

Upgrade to current version

Upgrade to OpenREM 0.10.0 from 0.7.3 or later and then upgrade to 1.0.

Specific upgrade instructions

For the original upgrade instructions, the last docs release to include them was 0.10.0-docs

OpenREM Release Notes version 0.7.1

Headline changes

- System
  - Django upgraded to version 1.8
  - Median function added to the database if using PostgreSQL

- New user-defined display name for each unique system so that rooms with the same DICOM station name are displayed separately

- Patient name and ID can optionally be stored in system, available for searching and export, but not displayed

- Patient name, ID and accession number can be stored as a one-way hash, and remain searchable

- Permission system has become more granular

- System can now accept non-ASCII characters in protocol names etc

- Menus have been tidied up

- Settings file has been updated

- Charts and interface

  - Bar chart data points sorted by frequency, value or name in ascending or descending order

  - CT chart of DLP per requested procedure type

  - CT chart of requested procedure frequency

  - CT chart of CTDIvol per study description

  - Chart data returned using AJAX to make pages more responsive

  - Chart plotting options available via Config menu

  - Charts can now be made full-screen

  - CTDIw phantom size is displayed with the CTDIvol measurement on the CT study detail page

  - Charts show a series called „Blank" when the series name is `None`

  - Queries for chart data now faster in most situations

  - Histograms can be disabled or enabled for bar charts

  - User-specified number of histogram bins from 2 to 40

  - Mammography chart of average glandular dose vs. compressed thickness

  - Mammography chart showing the number of studies carried out per weekday

  - Fluoroscopy chart of average DAP for each study description

  - Fluoroscopy chart of the frequency of each study description

  - Fluoroscopy chart showing the number of studies carried out per weekday

  - Context specific documentation has been added to the Docs menu

- DICOM Networking

  - Query retrieve function is now built in to query PACS systems or modalities via the Import menu

  - Configuring and running DICOM Store SCP is available and managed in the web interface, but not recommended

  - Documentation improved

- Importe

  - Mammography RDSRs import correctly

  - Mammography imports from images **now create an accumulated AGD value per breast**

  - GE Senographe DS compression **now recorded correctly in Newtons** for new imports

  - Philips Allura fluoroscopy RDSRs import correctly, including calculating the exposure time

---

- Bi-plane fluoroscopy imports can now be displayed in the web interface

- Patient height imports from csv **now convert from cm to m** - previously height was assumed to be cm and inserted into database without change. Existing height data will remain as cm value for csv imports, and m value for RDSR imports

- Better handling of non-ASCII characters

- Tube current is now extracted from Siemens Intevo RDSRs

- Exports

  - Patient sex is included in all exports

  - Filters generated by navigating through charts can now be used to filter export data

  - Study description and laterality are now included in mammography exports

  - Bi-fluoroscopy studies can be exported

- Skin dose maps

  - Skin dose maps have been withdrawn from OpenREM version 0.7.0 due to incorrect orientation calculations that need to be fixed before openSkin can be reimplemented into OpenREM

Changes since 0.7.0

Extremely minor change to the documenation links

Specific upgrade instructions

For the original upgrade instructions, the last docs release to include them was 0.10.0-docs

OpenREM Release Notes version 0.6.0

Headline changes

- Charts

- Preview of DICOM Store SCP functionality

- Exports available to import into openSkin

- Modalities with no data are hidden in the user interface

- Mammography import compression force behaviour changed

- Import of Toshiba planar RDSRs fixed

Changes for 0.6.2

Minor update due prevent new installs from installing a non-compatible version of `django-filter`. The link to openSkin has also been updated in the fluoroscopy detail page.

There is no advantage to updating to this version over 0.6.0

Release 0.6.1 was just a documentation only change to update the link to openSkin.

Specific upgrade instructions

For the original upgrade instructions, the last docs release to include them was 0.10.0-docs

Summary of new features

Charts

Release 0.6.0 has a range of charting options available for CT and radiographic data. These charts allow visualisation of trends and frequencies to inform surveys and monitor performance. For more information, please see *Charts*.

DICOM Store Service Class Provider

OpenREM can now act as the DICOM Store service, allowing direct sending of DICOM objects from modalities to OpenREM without needing to use Conquest or any other DICOM Store SCP. This feature is a preview as it hasn't been extensively tested, but it is expected to work. For more information, please see *Direct from modalities*.

Exports for openSkin

Fluoroscopy studies can now be exported in a format suitable for importing into Jonathan Cole's openSkin software. The export link is on the fluoroscopy study detail page. The software for creating the exposure incidence map can be downloaded from https://bitbucket.org/openskin/openskin/downloads (choose the zip file), and information about the project can be found on the openSkin wiki. The software allows the user to choose between a 2D phantom that would represent the dose to a film laying on the couch surface, or a simple 3D phantom made up of a cuboid and two semi-cylinders (these can be seen on the Phantom design section of the wiki). For both options the output is an image of the dose distribution in 2D, along with calculated peak skin dose information.

Automatic hiding of unused modality types

A fresh install of OpenREM will no longer show any of the four modality types in the tables or in the navigation bar at the top. As DICOM objects are ingested, the appropriate tables and navigation links are created.

Therefore a site that has no mammography for example will no longer have that table or navigation link in their interface.

Mammography import compression force change

Prior to version 0.6, the compression force extracted from the mammography image header was divided by ten before being stored in the database. This was because the primary author only had access to GE Senograph DS units, which store the compression force in dN, despite claiming using Newtons in the DICOM conformance statement.

The code now checks for the term *senograph ds* contained in the model name. If it matches, then the value is divided by ten. Otherwise, the value is stored without any further change. We know that later GE units, the GE Senograph Essential for example, and other manufacturer's units store this value in N. If you have a case that acts like the Senograph DS, please let us know and we'll try and cater for that.

If you have existing non-GE Senograph mammography data in your database, the compression force field for those studies is likely to be incorrect by a factor of ten (it will be too small). Studies imported after the upgrade will be correct. If this is a problem for you, please let us know and we'll see about writing a script to correct the existing data.

Import of Toshiba Planar RDSRs fixed

Toshiba include Patient Orientation and Patient Orientation Modifier information in their cath lab RDSRs. The extractor code was deficient for this as the RDSRs previously used didn't have this information. This has now been fixed. There might however be an issue with Station Name not being provided - it is not yet clear if this is a configuration issue.

OpenREM Release Notes version 0.5.1

Headline changes

- Major database modification to remove table name length errors
- Extended the field value lengths to better incorporate all possible values and decimal places
- Improved import of grid and filter information from DX images
- Improved DX summary and detail web pages
- Any item in a row can now be clicked to move between the home and filtered pages

Specific upgrade instructions

For the original upgrade instructions, the last docs release to include them was 0.10.0-docs

OpenREM Release Notes version 0.5.0

Headline changes

- Import, display and export of CR/DX data from image headers
- Export of study data from fluoroscopy to xlsx files
- Importing data from Windows using `*.dcm` style wildcards
- Hologic tomography projection images are no longer excluded if part of a Combo exposure

Specific upgrade instructions

For the original upgrade instructions, the last docs release to include them was 0.10.0-docs

OpenREM Release Notes version 0.4.3

Headline changes

- Export of study information is now handled by a task queue - no more export time-outs.
- Patient size information in csv files can now be uploaded and imported via a web interface.
- Proprietary projection image object created by Hologic tomography units can now be interrogated for details of the tomosynthesis exam.
- Settings.py now ships with its proper name, this will overwrite important local settings if upgrade is from 0.3.9 or earlier.
- Time since last study is no longer wrong just because of daylight saving time!
- Django release set to 1.6; OpenREM isn't ready for Django 1.7 yet
- The inner `openrem` Django project folder is now called `openremproject` to avoid import conflicts with Celery on Windows
- DEBUG mode now defaults to False

Specific upgrade instructions

For the original upgrade instructions, the last docs release to include them was 0.10.0-docs

OpenREM Release Notes version 0.4.2

Headline changes

- This release fixes a major bug introduced in 0.4.0 regarding the import scripts.

Specific upgrade instructions

Upgrading from 0.3.9 or earlier

Follow the instructions in *OpenREM Release Notes version 0.4.0*

Upgrading from 0.4.0 or above

Move straight to version 0.4.3 and follow the instructions in *OpenREM Release Notes version 0.4.3*

OpenREM Release Notes version 0.4.1

Headline changes

- This release is exacly the same as 0.4.1 bar some documentation corrections

Specific upgrade instructions

Please use the 0.4.0 release notes for upgrades from 0.3.9

OpenREM Release Notes version 0.4.0

OpenREM Release Notes version 0.4.0

Headline changes

- User authentication has been added
- Studies can be deleted from the web interface
- Import scripts can now be passed a list of files, eg `python openrem_rdsr.py *.dcm`
- Date of birth no longer retained for mammography (bug fix - correct behaviour already existed for other imports)
- General bug fixes to enable import from wider range of sources
- Improved user documentation

Specific upgrade instructions

For the original upgrade instructions, the last docs release to include them was 0.10.0-docs

12.3 Contributing authors

Many people have contributed to OpenREM - either with code, documentation, translations, bugs, examples or ideas, including:

- BjörnAlbers
- Erlend Andersen
- Njål Brekke
- Elly Castellano
- Jonathan Cole
- Jamie Dormand
- Ben Earner

- Louise Giansante Martins
- Daniel Gordon
- Hamid Khosravi
- Laurence King
- Wens Kong
- Eivind Larsen
- John Loveland
- Paolo Marcheschi
- Ed McDonagh
- Richard Miles
- Allan Nordhøy
- Luuk Oostveen
- David Platten
- Richard Raynor
- Erik-Jan Rijkhorst
- Kevin Schärer - Kantonsspital Aarau AG
- Arnold Schilham
- Marcelo Sosa
- Georg Stamm
- Jannis Widmer - Kantonsspital Aarau AG
- Tim de Wit
- Daniel Wyatt

Special thanks go to Gerd Lutters of Kantonsspital Aarau AG for providing ideas and and civil service computer science students to work on OpenREM (Jannis and Kevin).

KAPITEL 13

Diagram of system components

Client
web browser

X-ray imaging
modality

PACS

via user-requests
and Ajax

via modality
configuration

via OpenREM
query-retrieve

OpenREM server

Web server[1]

DICOM StoreSCP[2]

Data export to
local file system

Does the
object contain
useful data?

Yes

No

OpenREM
Django app

Extract information from
the DICOM object to the
OpenREM database

Delete the DICOM object
from the DICOM store

via psycopg2
Python adapter

Database[3]

Calculate
skin dose
map?

Yes

Skin dose map
data calculation,
storage, retrieval

Kapitel 13. Diagram of system components

13.1 Alternatives

13.1.1 1: Web servers

The recommended web server for Windows is Microsoft IIS - see *to be written* docs for details. This has replaced the recommendation to use Apache due to difficulties in obtaining the required binary files, as described in the *Advanced server configuration* section of the installation document.

The recommended web server for Linux is Gunicorn with NGINX - see *Webserver* for details.

Alternatively, a built-in web server is included that will suffice for testing purposes and getting started.

13.1.2 2: DICOM Store node

Any DICOM Store can be used, as long as it can be used to call the OpenREM import script. See *DICOM Network Configuration* for more details. Orthanc is the recommended DICOM Store services to use; it is installed by default in Docker, see *DICOM Store SCP* for Linux installation, *to be written* for Windows installation, and the *DICOM Store* section for configuration help.

13.1.3 3: Database

PostgreSQL is the recommended database to use with OpenREM. It is the only database that OpenREM will calculate median values for charts with. Other databases can be used with varying capabilities; see the Django documentation for more details. For testing only, the built-in SQLite3 database can be used, but this is not suitable for later migration to a production database.

KAPITEL 14

Indizes und Tabellen

- genindex
- modindex
- search

Stichwortverzeichnis