OpenREM Documentation

Release 0.9.1

Ed McDonagh

Jul 19, 2019

Contents

OpenREM is a free, open source application for patient radiation dose monitoring. The software is capable of importing and displaying data from a wide variety of x-ray dose related sources with filtering, charts and analysis. The software also enables easy export of the data in a form that is suitable for further analysis by suitably qualified medical physics personnel.

Please see openrem.org for more details.

Contents:

CHAPTER 1

Installation

1.1 Pre-installation preparations

1.1.1 Install Python 2.7.x and pip

- Windows – instructions and downloads are available at python.org
- Linux – likely to be installed already except on newer distributions

  Check by typing `python -V` - if the response is `Python 2.7.x` (x can be any number 9 or greater) then move on to installing pip.

  If the response is that the command can't be found, you will need to install Python. On Ubuntu, Python 2.7 is python and Python 3.x is Python3:

  ```
  sudo apt install python
  ```

Add Python and the scripts folder to the path

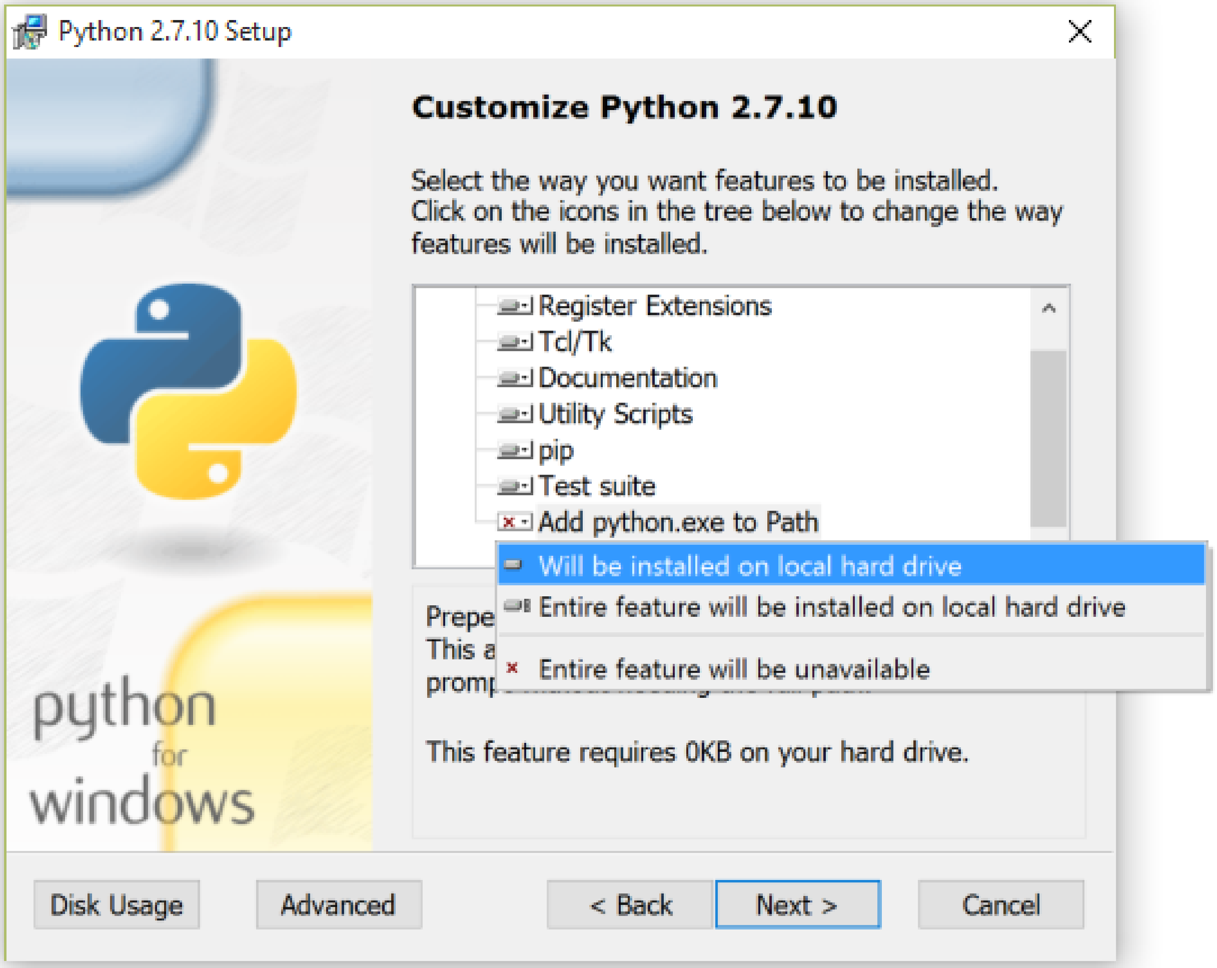
Windows only – this is usually automatic in linux

During the Windows Python 2.7 installation, 'install' Add Python.exe to Path:

If Python is already installed, you can add Python to Path yourself:

> Add the following to the end of the `path` environment variable (to see how to edit the environment variables, see http://www.computerhope.com/issues/ch000549.htm):

```
;C:\Python27\;C:\Python27\Scripts\
```



Python 2.7.10 Setup

Customize Python 2.7.10

Select the way you want features to be installed.
Click on the icons in the tree below to change the way features will be installed.

- Register Extensions
- Tcl/Tk
- Documentation
- Utility Scripts
- pip
- Test suite
- Add python.exe to Path

Will be installed on local hard drive
Entire feature will be installed on local hard drive
Entire feature will be unavailable

Prepe
This a
promp

This feature requires 0KB on your hard drive.

Fig. 1: Python installation customisation dialogue

Python package installer pip

Install pip – for details go to https://pip.pypa.io/en/latest/installing/. The quick version is as follows:

Linux

Download the latest version using the same method as for Windows, or get the version in your package manager, for example:

```
sudo apt-get install python-pip
```

Windows

Pip is normally installed with Python. If it hasn't been, download the installer script get-pip.py and save it locally – right click and *Save link as…* or equivalent.

Open a command window (Start menu, cmd.exe) and navigate to the place you saved the get-pip.py file:

```
python get-pip.py
```

Quick check of python and pip

To check everything is installed correctly so far, type the following in a command window/shell. You should have the version number of pip returned to you:

```
pip -V
```

1.1.2 Install RabbitMQ

- Linux - Follow the guide at http://www.rabbitmq.com/install-debian.html
- Windows - Follow the guide at http://www.rabbitmq.com/install-windows.html

For either install, just follow the defaults – no special configurations required. Please note that RabbitMQ requires that Erlang is installed first, as described in the above links.

Note: If you encounter problems running RabbitMQ as a service under Windows then try the following:

- Create a folder called `c:\rabbitmq`
- From an administrator command prompt run Advanced System Properties by typing `sysdm.cpl`
- Create a new system environment variable called `RABBITMQ_BASE` and set its value to `c:\rabbitmq`
- In the command prompt navigate to the folder containing the RabbitMQ commands and run:

```
rabbitmq_service.bat remove
rabbitmq_service.bat install
rabbitmq_service.bat start
```

Enable RabbitMQ queue management interface

Linux

In a terminal:

```
sudo rabbitmq-plugins enable rabbitmq_management
```

Windows

In a terminal, find the sbin folder of the RabbitMQ installation directory - it will be something like `C:\Program Files\RabbitMQ Server\rabbitmq_server-3.7.8\sbin`. You might like to find the `sbin` folder using Windows Explorer and then type `cd` into a terminal followed by a space then drag the folder icon from the left hand end of the address bar and drop it into the terminal, followed by the `Enter` key.

```
rabbitmq-plugins enable rabbitmq_management
```

Optional - RabbitMQ Administrator

An administrator user is not required to view RabbitMQ queues and to purge queues from the OpenREM web interface.

However, if you wish to interact directly with the RabbitMQ management interface, you should create a user for this purpose.

The password is printed to the terminal, so in Linux add a space before the `sudo` so that the command does not get saved to your history file, and then `clear` the terminal so it isn't displayed any longer. In Windows, use `cls`:

Linux:

```
sudo rabbitmqctl add_user <username> <password>
clear
sudo rabbitmqctl set_user_tags <username> administrator
sudo rabbitmqctl set_permissions -p / <username> "." "." ".*"
```

Windows:

```
rabbitmqctl add_user <username> <password>
cls
rabbitmqctl set_user_tags <username> administrator
rabbitmqctl set_permissions -p / <username> "." "." ".*"
```

Note: Before continuing, *consider virtualenv*

1.1.3 Install PostgreSQL database

For production use, you will need to install and configure a database. We strongly recommend PostgreSQL, but you can use any of the databases listed on the Django website such as MySQL, Oracle or MS SQL Server, with the limitations listed there. There is one additional limitation - the calculation of median values for charts in OpenREM is dependent on using PostgreSQL.

If this is your first time installing OpenREM and you just want to test it out, you *can* skip this step and make use of the in-built SQLite database. However, you should expect to start again when you move to a production grade database.

- *PostgreSQL database (Linux)*
- *PostgreSQL database (Windows)*

1.1.4 Install a DICOM Store service

To have modalities send DICOM objects to your OpenREM server, or to use query-retrieve from a PACS, you need to install a DICOM Store service. For testing, you can make use of the DICOM Store OpenREM can provide. However, because this is not stable over longer periods of time we recommend using a third-party DICOM Store service. You can use any one you like, as long as it can be scripted to call OpenREM scripts when DICOM objects are received. We recommend Orthanc or Conquest for this and provide details of how to configure them in the *Third-party DICOM Stores* section.

Orthanc

- Ubuntu users: `sudo apt install orthanc`
- Windows users: Download from https://www.orthanc-server.com/download-windows.php after filling in the form
- Configuration instructions can be found in the *Third-party DICOM Stores* section.

Alternative - Conquest

- Download Conquest DICOM server from https://ingenium.home.xs4all.nl/dicom.html
- Install using the instructions included in the download - there is a PDF with Windows install instructions and general usage instructions, and another PDF with Linux install instructions. The guides in *Third-party DICOM Stores* should be consulted when making configuration decisions.
- Alternatively, Ubuntu 16.04 users can use the following instructions:

Conquest DICOM store node on Ubuntu 16.04

Installation

Ubuntu 16.04 has the Conquest DICOM server in its repositories, so this makes installation very easy. However, it isn't the latest version and isn't going to be updated further. However, if you are using Ubuntu 16.04 then it does make for a relatively easy install.

There are options to install with different databases – for OpenREM we're not really going to use the database so the easiest option is to use SQLite:

```
sudo apt-get install conquest-sqlite
```

Basic configuration

Modify dgatesop.lst

Edit the `dgatesop.lst` file in the `/etc/conquest-dicom-server` folder, for example

```
sudo nano /etc/conquest-dicom-server/dgatesop.lst
```

And add the following line

```
XRayRadiationDoseSR 1.2.840.10008.5.1.4.1.1.88.67   sop
```

It isn't critical where it goes, but I tend to add it where it belongs between `KeyObjectSelectionDocument` and `PETStorage`. I also add in the spaces to make it line up, but again this is just for aesthetic reasons!

If you are pasting from the clipboard into nano from within Linux, use `Shift-Ctrl-v`. If you are using PuTTY in Windows to interact with Ubuntu, a right click on the mouse or `Shift-Insert` should paste the text into the terminal.

To save and exit from nano, use `Ctrl-o` (out), press return to confirm the filename and then `Ctrl-x` (exit).

Configuration

Here will be a link to the dicom.ini config… the text below will be subsumed into it

Configure the Store SCP

Edit the `dicom.ini` file in the `/etc/conquest-dicom-server` folder, for example

```
sudo nano /etc/conquest-dicom-server/dicom.ini
```

Modify the following lines as required. The server name field – with the Conquest default of `CONQUESTSRV1` – is the AE Title, so should be 16 characters or less and consist of letters and numbers with no spaces. It is case insensitive. The `TCPPort` is normally either 104, the standard DICOM port, or any number greater than 1023.

```
# Network configuration: server name and TCP/IP port#
MyACRNema                 = CONQUESTSRV1
TCPPort                   = 11112
```

Again, save and exit.

If you've changed the AE Title and/or port, restart conquest:

```
sudo /etc/init.d/dgate restart
```

Testing basic configuration

Test the Store SCP by returning to OpenREM and navigating to `Config -> DICOM network configuration`.

Click to `Add new store` and enter the AE title and port you have set, along with a reference name.

Click to `Submit`, and you will return to the summary page which should inform you if the server is running.

Configure Conquest to work with OpenREM

The next stage is to configure Conquest to store the incoming object and ask OpenREM to process them.

Conquest configuration

At the end of the `/etc/conquest-dicom-server/dicom.ini` file, add the following lines. You will need to tailor them to save the file to an appropriate place. The `_conquest` user will need to be able to write to that location. You will also need to make sure the path to the scripts you just created are correct.

The example below assumes images will be saved in `/var/lib/conquest-dicom-server/incoming/`, which you can create as follows:

```
sudo mkdir /var/lib/conquest-dicom-server/incoming
sudo chown _conquest:_conquest /var/lib/conquest-dicom-server/incoming
```

Each instruction in the `dicom.ini` file below has a `destroy` instruction to delete Conquest's copy of the file and to remove it from it's database. This isn't the version we've saved in `incoming` to process.

```
sudo nano /etc/conquest-dicom-server/dicom.ini
```

```
# RDSR
ImportConverter0  = ifequal "%V0008,0016","1.2.840.10008.5.1.4.1.1.88.67"; {save␣
↪to /var/lib/conquest-dicom-server/incoming/%o.dcm; system /etc/conquest-dicom-
↪server/openrem-rdsr.sh /var/lib/conquest-dicom-server/incoming/%o.dcm; destroy}
# Import arguments for GE CT - uses Enhanced SR instead of Radiation Dose SR
ImportConverter1  = ifequal "%V0008,0016","1.2.840.10008.5.1.4.1.1.88.22"; {save␣
↪to /var/lib/conquest-dicom-server/incoming/%o.dcm; system /etc/conquest-dicom-
↪server/openrem-rdsr.sh /var/lib/conquest-dicom-server/incoming/%o.dcm; destroy}

# MG images
ImportModality2   = MG
ImportConverter2  = save to /var/lib/conquest-dicom-server/incoming/%o.dcm;␣
↪system /etc/conquest-dicom-server/openrem-mg.sh /var/lib/conquest-dicom-server/
↪incoming/%o.dcm; destroy

# DX images
ImportModality3   = DX
ImportConverter3  = save to /var/lib/conquest-dicom-server/incoming/%o.dcm;␣
↪system /etc/conquest-dicom-server/openrem-dx.sh /var/lib/conquest-dicom-server/
↪incoming/%o.dcm; destroy
# CR images
ImportModality4   = CR
ImportConverter4  = save to /var/lib/conquest-dicom-server/incoming/%o.dcm;␣
↪system /etc/conquest-dicom-server/openrem-dx.sh /var/lib/conquest-dicom-server/
↪incoming/%o.dcm; destroy

# Philips CT
ImportConverter5  = ifequal "%V0008,0016","1.2.840.10008.5.1.4.1.1.7"; {save to /
↪var/lib/conquest-dicom-server/incoming/%o.dcm; system /etc/conquest-dicom-
↪server/openrem-ctphilips.sh /var/lib/conquest-dicom-server/incoming/%o.dcm;␣
↪destroy}

# Other objects
ImportConverter6  = destroy
```

Finally, restart conquest to make use of the new settings:

```
sudo /etc/init.d/dgate restart
```

Unlike with the database, it is possible to change DICOM Store service at a later point.

1.1.5 Resources for creating RDSR for older Toshiba CT scanners

New in version 0.8.0

If you need to import data from older Toshiba CT scanners into OpenREM then the following tools need to be available on the same server as OpenREM:

- The Offis DICOM toolkit
- Java
- pixelmed.jar from the PixelMed Java DICOM Toolkit

For more information see *For CT dose summary files from older Toshiba CT scanners*. The locations of these executables needs to be configured in the `local_settings.py` - see *Toshiba CT RDSR creation*.

1.1.6 Install OpenREM

You are now ready to install OpenREM, so go to the *Installing OpenREM* docs.

1.1.7 Further instructions

Virtualenv and virtualenvwrapper

If the server is to be used for more than one python application, or you wish to be able to test different versions of OpenREM or do any development, it is highly recommended that you use virtualenv or maybe virtualenvwrapper

Virtualenv sets up an isolated python environment and is relatively easy to use.

If you do use virtualenv, all the paths referred to in the documentation will be changed to:

- Linux: `vitualenvfolder/lib/python2.7/site-packages/openrem/`
- Windows: `virtualenvfolder\Lib\site-packages\openrem`

In Windows, even when the virtualenv is activated you will need to call *python* and provide the full path to script in the *Scripts* folder. If you call the script (such as *openrem_rdsr.py*) without prefixing it with *python*, the system wide Python will be used instead. This doesn't apply to Linux, where once activated, the scripts can be called without a *python* prefix from anywhere.

1.2 Installing OpenREM

1.2.1 Windows only - install Celery

For Linux users, the correct version of Celery will be install automatically with OpenREM.

Activate virtualenv if you are using one

```
pip install celery==3.1.25
```

1.2.2 Install OpenREM

```
pip install openrem
```

Will need `sudo` *or equivalent if installing on linux without using a virtualenv*

1.2.3 Install pynetdicom (edited version)

Pynetdicom is used for the DICOM Store SCP and Query Retrieve SCU functions. See *Direct from modalities* for details.

```
pip install https://bitbucket.org/edmcdonagh/pynetdicom/get/default.tar.gz
↪#egg=pynetdicom-0.8.2b2
```

Note: You must install the `pynetdicom` package from the link above - the version in pypi or the newer versions in GitHub won't work with the current version of OpenREM. Future versions of OpenREM will be adapted to work with `pynetdicom3` and `pydicom>1.0`.

If you are using the latest version of `pip` you will get error messages including the phrase `Failed building wheel for pynetdicom` - it should be ok to ignore this message as long as you end up with the message `Successfully installed pynetdicom-0.8.2b2`

1.2.4 Configuration

Locate install location

- 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\`

There are two files that need renaming:

- `openremproject/local_settings.py.example` to `openremproject/local_settings.py`
- `openremproject/wsgi.py.example` to `openremproject/wsgi.py`

Edit local_settings.py

Note: Windows notepad will not recognise the Unix style line endings. Please use an editor such as Notepad++ or Notepad2 if you can, else use WordPad – on the View tab you may wish to set the Word wrap to 'No wrap'

Important: In local_settings.py, always use forward slashes and not backslashes, even for paths on Windows systems.

The directories in this local_settings.py file must already exist - OpenREM will not create them for you.

Database

Note: SQLite is great for getting things running quickly and testing if the setup works, but is not recommended for production use.

We recommend using PostgreSQL as it is the best supported database for Django, **and the only one for which the median value will be calculated and displayed in OpenREM charts.** Alternatively, other databases such as MySQL/MariaDB, Oracle, and some others with lower levels of support can be used.

There are some further guides to setting up PostgreSQL – see *Databases*

If you are using SQLite:

```
'ENGINE': 'django.db.backends.sqlite3',
'NAME': '/ENTER/PATH/WHERE/DB/FILE/CAN/GO',
```

- Linux example: `'NAME': '/home/user/openrem/openrem.db',`
- Windows example: `'NAME': 'C:/Users/myusername/Documents/OpenREM/openrem.db',`

If you are using PostgreSQL:

```
'ENGINE': 'django.db.backends.postgresql_psycopg2',
'NAME': 'openremdb',
'USER': 'openremuser',
'PASSWORD': 'openrem_pw',
```

Location for imports and exports

Csv and xlsx study information exports and patient size csv imports are written to disk at a location defined by `MEDIA_ROOT`.

The path set for `MEDIA_ROOT` is up to you, but the user that runs the webserver must have read/write access to the location specified because it is the webserver than reads and writes the files. In a debian linux, this is likely to be www-data for a production install. Remember to use forward slashes for the config file, even for Windows.

Linux example:

```
MEDIA_ROOT = "/var/openrem/media/"
```

Windows example:

```
MEDIA_ROOT = "C:/Users/myusername/Documents/OpenREM/media/"
```

Secret key

Generate a new secret key and replace the one in the `local_settings.py` file. You can use http://www.miniwebtool.com/django-secret-key-generator/ for this.

Allowed hosts

The `ALLOWED_HOSTS` needs to be defined, as the `DEBUG` mode is now set to `False`. This needs to contain the OpenREM server hostname or IP address that will be used in the URL in the web browser. For example:

```
ALLOWED_HOSTS = [
    '192.168.56.102',
    '.doseserver.',
    'localhost',
]
```

A dot before a hostname allows for subdomains (eg www.doseserver), a dot after a hostname allows for FQDNs (eg doseserver.ad.trust.nhs.uk). Alternatively, a single `'*'` allows any host, but removes the security the feature gives you.

Customised date format in xlsx exports

# TODO: check csv situation The default date form at for OpenREM xlsx exports is dd/mm/yyyy. If you wish to customise this, uncomment the *XLSX_DATE* line, for example the standard US date format would be:

```
XLSX_DATE = 'mm/dd/yyyy'
```

Log file

There are two places logfiles need to be configured - here and when starting Celery. The logs defined here capture most of the information; the Celery logs just capture workers starting and tasks starting and finishing.

Configure the filename to determine where the logs are written. In linux, you might want to send them to a sub-folder of `/var/log/`. In this example, they are written to the `MEDIA_ROOT`; change as appropriate:

```
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")

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
```

If you want all the logs in one file, simply set them all to the same filename.

In the settings file, there are `simple` and `verbose` log message styles. We recommend you leave these 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
```

Next, you can set the logging level. Options are `DEBUG`, `INFO`, `WARNING`, `ERROR`, and `CRITICAL`, with progressively less logging. `INFO` is probably a good choice for most circumstances. `DEBUG` is useful if something is going wrong, but it is quite chatty for routine use!

```
LOGGING['loggers']['remapp']['level'] = 'INFO'                    # General logs
LOGGING['loggers']['remapp.netdicom.qrscu']['level'] = 'INFO'     # Query Retrieve␣
↪SCU logs
```

(continues on next page)

(continued from previous page)

```
LOGGING['loggers']['remapp.netdicom.storescp']['level'] = 'INFO'  # Store SCP logs
LOGGING['loggers']['remapp.extractors.ct_toshiba']['level'] = 'INFO'  # Toshiba RDSR␣
↪creation extractor logs
```

Finally, if you are using Linux you can set the system to start a new log file automatically when the current one gets to a certain size. The settings described below don't work with Windows - we'll try to include Windows settings in the next release. See issue 483 to find out the progress on this!

To activate the 'rotating' log function, uncomment the remaining lines by removing the # from the beginning of the lines. For example for the query retrieve logs:

```
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
```

Time zone

Configure the local timezone in order to get correct times in the logfiles. The default timezone is set at 'Europe/London'. Valid options can be found here: http://en.wikipedia.org/wiki/List_of_tz_zones_by_name

```
TIME_ZONE = 'Europe/London'
```

Language

Configure the local language. Default language is set at us-english. Valid options can be found here: http://www.i18nguy.com/unicode/language-identifiers.html

```
LANGUAGE_CODE = 'en-us'
```

Toshiba CT RDSR creation

If you need to import data from older Toshiba CT scanners that do not create RDSRs then the following tools need to be available on the same server as OpenREM:

- The Offis DICOM toolkit
- Java
- pixelmed.jar from the PixelMed Java DICOM Toolkit

The paths to these must be set in `local_settings.py` for your system:

```
# Locations of various tools for DICOM RDSR creation from CT images
DCMTK_PATH = 'C:/Apps/dcmtk-3.6.0-win32-i386/bin'
DCMCONV = os.path.join(DCMTK_PATH, 'dcmconv.exe')
DCMMKDIR = os.path.join(DCMTK_PATH, 'dcmmkdir.exe')
JAVA_EXE = 'C:/Apps/doseUtility/windows/jre/bin/java.exe'
JAVA_OPTIONS = '-Xms256m -Xmx512m -Xss1m -cp'
PIXELMED_JAR = 'C:/Apps/doseUtility/pixelmed.jar'
PIXELMED_JAR_OPTIONS = '-Djava.awt.headless=true com.pixelmed.doseocr.OCR -'
```

The example above is for Windows. On linux, if you have installed the Offis DICOM toolkit with `sudo apt install dcmtk` or similar, you can find the path for the configuration above using the command `which dcmconv`. This will be something like `/usr/bin/dcmconv`, so the `DCMTK_PATH` would be `'/usr/bin` and the `DCMCONV` would be `os.path.join(DCMTK_PATH, 'dcmconv')`. Similarly for `DCMMKDIR` and `JAVA_EXE`, which might be `/usr/bin/java`. The pixelmed.jar file should be downloaded from the link above, and you will need to provide the path to where you have saved it.

Note: If you do not intend to use the RDSR creation feature (all your CT scanners create RDSRs already, or your older scanners are Philips), then these paths do not need to be changed for your install.

E-mail configuration

The settings below must be correctly configured for fluoroscopy high dose alert e-mail messages to be sent. It is recommended that you liaise with your IT department in order to obtain the settings required in this section.

```
# E-mail server settings
EMAIL_HOST = 'localhost'
EMAIL_PORT = 25
EMAIL_HOST_USER = ''
EMAIL_HOST_PASSWORD = ''
EMAIL_USE_TLS = False
EMAIL_USE_SSL = False
EMAIL_DOSE_ALERT_SENDER = 'your.alert@email.address'
EMAIL_OPENREM_URL = 'http://your.openrem.server'
```

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.

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 below should not be required.

The `EMAIL_USE_TLS` and `EMAIL_USE_TLS` options should be configured to match the encryption requirements of the e-mail server.

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.

Create the 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\`

Create the database:

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

The last command will list each Django app migrations. Each should have a cross inside a pair of square brackets something like below:

```
admin
 [X] 0001_initial
auth
 [X] 0001_initial
 [X] 0002_alter_permission_name_max_length
 [X] 0003_alter_user_email_max_length
 [X] 0004_alter_user_username_opts
 [X] 0005_alter_user_last_login_null
 [X] 0006_require_contenttypes_0002
contenttypes
 [X] 0001_initial
 [X] 0002_remove_content_type_name
remapp
 [X] 0001_initial
sessions
 [X] 0001_initial
sites
 [X] 0001_initial
```

Finally, create a Django super user:

```
python manage.py createsuperuser
```

Answer each question as it is asked – this user is needed to set up the other users and the permissions.

Add the median database function: PostgreSQL databases only

Rename the file

```
remapp/migrations/0002_0_7_fresh_install_add_median.py.inactive
```

to

```
remapp/migrations/0002_0_7_fresh_install_add_median.py
```

and then run

```
python manage.py migrate
```

This command runs the migration file, and will display the text `Applying remapp.0002_0_7_fresh_install_add_median... OK`, indicating that the median function has been added.

1.2.5 Start all the services!

You are now ready to start the services to allow you to use OpenREM - go to *Start all the services* to see how!

A standard installation assumes access to the internet from the computer where OpenREM is being installed. Sometimes this isn't possible, so we've added instructions for an offline installation too. Currently it focuses on Windows only (for the server - the computer connected to the internet can be running any operating system).

1.3 Offline Installation on Windows

In order to carry out an offline installation you will need to download the OpenREM package and dependencies. The instructions below should work for downloading on any operating system, as long as you have Python 2.7 (version 2.7.9 or later) and a reasonably up to date version of pip installed.

If you have trouble when installing the Python packages 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.

1.3.1 On a computer with internet access

Download independent binaries

Python from https://www.python.org/downloads/windows/

- Follow the link to the 'Latest Python 2 release'
- Download either the `Windows x86 MSI installer` for 32-bit Windows or
- Download `Windows x86-64 MSI installer` for 64-bit Windows

Erlang from https://www.erlang.org/downloads

- Download the latest version of Erlang/OTP. Again, choose between
- `Windows 32-bit Binary File` or
- `Windows 64-bit Binary File`

RabbitMQ from http://www.rabbitmq.com/install-windows.html

- Download `rabbitmq-server-x.x.x.exe` from either option

PostgreSQL from http://www.enterprisedb.com/products-services-training/pgdownload#windows

Note: Other databases such as MySQL are also suitable, though the median function for charts will not be available. For testing purposes only, you could skip this step and use SQLite3 which comes with OpenREM

- Download by clicking on the icon for `Win x86-32` or `Win x86-64`

A webserver although this can be left till later - you can get started with the built-in web server

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:

```
mkdir openremfiles
pip download -d openremfiles setuptools
```

Download specific version of Celery:

Linux server:

```
pip download -d openremfiles celery==4.2.2
```

Windows server:

```
pip download -d openremfiles celery==3.1.25
```

Download OpenREM and all other dependencies:

```
pip download -d openremfiles openrem==0.9.1
pip download -d openremfiles psycopg2-binary
pip download --no-deps -d openremfiles https://bitbucket.org/edmcdonagh/pynetdicom/
↪get/default.tar.gz#egg=pynetdicom-0.8.2b2
```

pynetdicom version

This of `pynetdicom` is modified in comparison to the version in PyPI, and will malfunction if you use the official version

Copy everything to the Windows machine

- Copy this directory plus all the binaries to the Windows server that you are using

1.3.2 On the Windows server without internet access

Installation of binaries

Install the binaries in the following order:

1. Python
2. Erlang
3. RabbitMQ

Installation of the python packages

In a console, navigate to the directory that your `openremfiles` directory is in, and

Ensure setuptools is up to date:

```
pip install --no-index --find-links=openremfiles setuptools -U
```

Install specific version of Celery:

Linux server:

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

Windows server:

```
pip install celery==3.1.25
```

Install OpenREM and other dependencies:

```
pip install --no-index --find-links=openremfiles openrem==0.9.1  # where openremfiles␣
↪is the directory you created
pip install --no-index --find-links=openremfiles psycopg2-binary

pip install openremfiles\default.tar.gz  # this is the custom version of pynetdicom
```

Install PostgreSQL

See the instructions to *Install PostgreSQL* on Windows.

Install webserver

If you are doing so at this stage.

1.3.3 Configure OpenREM ready for use

OpenREM is now installed, so go straight to the *Configuration* section of the standard installation docs

1.4 Upgrading an existing installation

1.4.1 Upgrade to OpenREM 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 preparation

Version 0.9 of OpenREM has a minimum Python version of 2.7.9 (still needs to be 2.7 not 3) and a minimum version of setuptools. If your installation was originally for OpenREM 0.6 in 2014 or earlier, these may now be too old and need updating.

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.

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 *Restore the database* 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
  ```

Upgrading an OpenREM server with no internet access

Follow the instructions found at *Upgrade an offline OpenREM installation*, before returning here to update the configuration, migrate the database and complete the upgrade.

Upgrading from version 0.7.1 or earlier

Follow the instructions to *OpenREM Release Notes version 0.7.3* first, then return to these instructions to upgrade to 0.9.1.

Upgrading from version 0.7.3 or later

Upgrade

- Back up your database
  - For PostgreSQL on linux you can refer to *Backup the database*
  - For PostgreSQL on Windows you can refer to *Backing up a PostgreSQL database (Windows)*
  - 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 version of Celery:

  Linux server:

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

  Windows server:

```
pip install celery==3.1.25
```

- Install the new version of OpenREM:

```
pip install openrem==0.9.1
```

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`

Date format - changed with 0.8

Set the date format for xlsx exports (need to check csv situation). Copy the following code into your `local_settings.py` file if you want to change it from `dd/mm/yyy`:

```
# 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'
```

Time zone and language - changed with 0.8

Consider setting the timezone and language in `local_settings.py`. See `local_settings.py.example`.

Add additional log file configuration - changed with 0.8

Warning: If the configuration is not added for the new `openrem_extractor.log` you will find it being created where ever you start the webserver from, and starting the webserver may fail.

Add the new extractor log file configuration to the `local_settings.py` - you can copy the 'Logging configuration' section from `local_settings.py.example` if you haven't made many changes to this section. See the *Log file* settings in the install instructions.

Warning: If you are upgrading from an earlier beta with the Toshiba RDSR creation logs defined, this has changed names and must be modified in `local_settings.py` before the migration below. It should be changed to:

```
LOGGING['loggers']['remapp.extractors.ct_toshiba']['level'] = 'INFO'  # Toshiba␣
↪RDSR creation extractor logs
```

substituting `INFO` for whichever level of logging is desired.

E-mail server settings - changed with 0.9.0

If you want selected OpenREM users to be automatically sent fluoroscopy high dose alerts then set the details of the e-mail server to be used in the *E-mail server settings* part of your `local_settings.py` file. 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`

Then change the e-mail section settings to reflect the e-mail server that is to be used:

```
EMAIL_HOST = 'localhost'
EMAIL_PORT = 25
EMAIL_HOST_USER = ''
EMAIL_HOST_PASSWORD = ''
EMAIL_USE_TLS = False
EMAIL_USE_SSL = False
EMAIL_DOSE_ALERT_SENDER = 'your.alert@email.address'
EMAIL_OPENREM_URL = 'http://your.openrem.server'
```

See the *E-mail configuration* documentation for full details.

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
```

Update static files

In the same shell/command window as you used above run the following command to clear the static files belonging to your previous OpenREM version and replace them with those belonging to the version you have just installed (assuming you are using a production web server. . .):

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

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`:

```
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.

Enable task management - changed in 0.9.0

RabbitMQ management interface

To make use of the RabbitMQ queue display and purge control, the management interface needs to be enabled. To do so, follow the instructions at *Enable RabbitMQ queue management interface*.

Celery management interface, Flower

To make use of the Celery task management, Flower needs to be running. To do so, follow the instructions in *Celery task management: Flower*. For 'one-page Ubuntu' installs, add the Flower related config and create, register and start the systemd service files as described in *Celery and Flower*. If you need to change the default Flower port of 5555 then make sure you do so in `openremproject\local_settings.py` to add/modify the line `FLOWER_PORT = 5555` as well as when you start Flower.

Celery for Windows config - changed in 0.9.0

For best performance and reliability when using Celery on Windows, if your command for starting Celery specifies a pool option, for example `-P solo`, remove it so that Celery reverts to using the default `prefork` pool. This will enable multiple tasks to run concurrently and it will be possible to terminate tasks.

If you are a Windows user you may also wish to review *Daemonising Celery and Flower on Windows* as the example control batch files have been updated.

Ubuntu installs that followed One page complete Ubuntu install

Systemd service files have been renamed in these 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!)

```
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
```

Restart all the services

Follow the guide at *Start all the services*.

1.4.2 Upgrade an offline OpenREM installation

Upgrading OpenREM requires new Python packages to be available as well as the latest version of OpenREM. These can be downloaded on any computer with Python 2.7 installed and an internet connection, though if you have trouble when installing the packages you might need to use a similar computer to the one you are installing on - same operating system and matching 32-bit or 64-bit.

OpenREM version 0.9 has a minimum Python version of 2.7.9. Use the instructions in the *Upgrade to OpenREM 0.9.1* release notes to check this before downloading the new OpenREM packages.

On a computer with internet access

In a console, navigate to a suitable place and create a new directory to collect all the packages in, then use pip to download them all:

```
mkdir openremfiles
pip download -d openremfiles setuptools
```

Download specific version of Celery:

Linux server:

```
pip download -d openremfiles celery==4.2.2
```

Windows server:

```
pip download -d openremfiles celery==3.1.25
```

Download OpenREM and all other dependencies:

```
pip download -d openremfiles openrem==0.9.1
```

Copy everything to the OpenREM server

- Copy the directory to the OpenREM server

On the OpenREM server without internet access

- Back up your database
  - For PostgreSQL on linux you can refer to *Backup the database*
  - For PostgreSQL on Windows you can refer to *Backing up a PostgreSQL database (Windows)*
  - 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

Upgrade setuptools:

```
pip install --no-index --find-links=openremfiles setuptools -U
```

Install specific version of Celery:

Linux server:

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

Windows server:

```
pip install celery==3.1.25
```

Install OpenREM:

```
pip install --no-index --find-links=openremfiles openrem==0.9.1
```

Now go back to *Update the configuration*, migrate the database and finish the upgrade.

1.5 Databases

During the installation process, you will need to install a database. For testing only, you can use the built in SQLite3 database, but for production use you will need a production grade database. This is covered in the *Pre-installation preparations* documentation, but as you will probably want to find the database instructions again, the links are repeated here.

1.5.1 PostgreSQL database (Linux)

Creating the database

Install PostgreSQL and the python connector

```
sudo apt-get install postgresql
```

If you are using a virtualenv, make sure you are in it and it is active (`source bin/activate`)

```
pip install psycopg2-binary
```

Change the security configuration

The default security settings are too restrictive to allow access to the database. Assumes version `10`, change as appropriate.

```
sudo nano /etc/postgresql/10/main/pg_hba.conf
```

Scroll down to the bottom of the file and edit the following line from `peer` to `md5`:

```
local   all             all                                     md5
```

Don't worry about any lines that start with a `#` as they are ignored. If you can't access the database when everything else is configured, you might need to revisit this file and see if there are other lines with a method of `peer` that need to be `md5`

Note: If you need to have different settings for different databases on your server, you can use the database name instead of the first `all`, and/or the the database user name instead of the second `all`.

Restart PostgreSQL so the new settings take effect:

```
sudo service postgresql restart
```

Optional: Specify the location for the database files

Advanced: specify location for the database files

Create a user for the OpenREM database

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

Enter a new password for the `openremuser`, twice

Create the OpenREM database

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

If this is your initial install, you are now ready to install OpenREM, so go to the *Installing OpenREM* docs.

If you are replacing a SQLite test install with PostgreSQL, continue here.

Configure OpenREM to use the database

Move to the OpenREM install directory:

- 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\`

Edit the settings file, eg

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

Set the following (changing database name, user and password as appropriate)

```
'ENGINE': 'django.db.backends.postgresql_psycopg2',
'NAME': 'openremdb',
'USER': 'openremuser',
'PASSWORD': 'openrem_pw',
```

Backup the database

Ad-hoc backup from the command line

```
sudo -u postgres pg_dump openremdb > /path/to/backup.bak
```

If you are moving a backup file between systems, or keeping a few backups, you may like to compress the backup; for example a 345 MB OpenREM database compresses to 40 MB:

```
tar -czf backup.bak.tar.gz backup.bak
```

Automated backup with a bash script

```
#! /bin/bash
rm -rf /path/to/db/backups/*
PGPASSWORD="openrem_pw" /usr/bin/pg_dump -Uopenremuser openremdb > /path/to/db/
↪backups/openrem.bak
```

This script could be called by a cron task, or by a backup system such as backuppc prior to running the system backup.

Restore the database

If the restore is taking place on a different system,

- ensure that PostgreSQL is installed and the same user has been added as was used to create the initial database (see *Creating the database*) – check `local_settings.py` if you can't remember the the user name used!
- Ensure that the new system has the same version of OpenREM installed as the system the database was backed up from.
- Ensure the `openrem/remapp/migrations/` folder has no files in except __init__.py

Create a fresh database and restore from the backup:

```
sudo -u postgres createdb -T template0 new_openremdb_name
sudo -u postgres psql new_openremdb_name < /path/to/db/backups/openrem.bak
```

Reconfigure `local_settings.py` with the new database details and introduce OpenREM to the restored database:

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

If you are creating a second system in order to test upgrading, you can do this now followed by the usual `python manage.py makemigrations remapp` then `python manage.py migrate remapp` as per the upgrade instructions.

Useful PostgreSQL commands

```
-- Start the PostgreSQL console
sudo -u postgres psql

-- List users
\du

-- List databases
\l

-- Exit the console
\q
```

Advanced: specify location for the database files

You might like to do this if you want to put the database on an encrypted location instead of `/var/lib/postgresql`.

For this example, I'm going to assume all the OpenREM programs and data are in the folder `/var/openrem/` and PostgreSQL is at version `10` (change both as appropriate)

```
sudo service postgresql stop
mkdir /var/openrem/database
sudo cp -aRv /var/lib/postgresql/10/main /var/openrem/database/
sudo nano /etc/postgresql/10/main/postgresql.conf
```

Change the line

```
data_directory = '/var/lib/postgresql/10/main'
```

to

```
data_directory = '/var/openrem/database/main'
```

then restart PostgreSQL:

```
sudo service postgresql start
```

1.5.2 PostgreSQL database (Windows)

Note: Original author JA Cole

Get PostgreSQL and the python connector

- Download the installer from http://www.enterprisedb.com/products-services-training/pgdownload#windows
- Install the Python PostgreSQL connector (activate the virtualenv first if using):

  ```
  pip install psycopg2-binary
  ```

Install PostgreSQL

Run the the postgresql installer. It will ask for a location. Ensure the "data" directory is *not* under "Program Files" as this can cause permissions errors. Enter a superuser password when prompted. Make sure you keep this safe as you will need it.

Create a user and database

Open pgAdmin

- Click on servers to expand
- Double click on PostgreSQL 9.6 (or whichever version you have installed)
- Enter your superuser password
- Right click on "login roles" and choose "New login role"
- Create the openremuser (or whatever you want your user to be called) and under "Definition" add a password
- Under "Privileges" ensure that "Can login" and "Create databases" are set to "Yes"
- Click OK
- Right click on databases and choose "New database"
- Name the database (openremdb is fine) and assign the the owner to the user you just created

If this is your initial install, you are now ready to install OpenREM, so go to the *Installing OpenREM* docs.

If you are replacing a SQLite test install with PostgreSQL, continue here.

Configure OpenREM to use the database

Find and edit the settings file (notepad works fine). The path depends on your python install, but could be something like:

- `C:\lib\python2.7\site-packages\openrem\openremproject\local_settings.py`

Set the following (changing name, user and password as appropriate):

- `'ENGINE': 'django.db.backends.postgresql_psycopg2',`
- `'NAME': 'openremdb',`
- `'USER': 'openremuser',`
- `'PASSWORD': 'openrem_pw',`

1.5.3 Backing up a PostgreSQL database (Windows)

Note: Content contributed by DJ Platten

These instructions are based on PostgreSQL 9.1 and OpenREM 0.5.0 running on Windows Server 2008. The database restore has been tested on Ubuntu 12.04 LTS.

Create a PostgreSQL user called `backup` with a password of `backup`. This is easiest to do using the `pgAdminIII` tool: you'll need to create a new login role. In the role privileges ensure that at least `Can login`, `Superuser` and `Can initiate streaming replication and backups` are checked.

The `pgAdminIII` tool is available by default on Windows, but needs to be explicitly installed if using Ubuntu with the following command:

```
sudo apt-get install pgadmin3
```

For the remainder of this article I'm going to assume that your OpenREM database is called `openrempostgresql`.

To backup the contents of `openrempostgresql` to a file called `backup.sql` run the following at the command line in a command prompt (Windows), or terminal window (Ubuntu):

```
pg_dump -U backup -F c -b -v -f backup.sql openrempostgresql
```

You will need to add your `C:\path\to\postgres\bin` folder to the `path` environment variable for this to work. Make sure to use the actual path to your PostgreSQL `bin` folder rather than the example text provided here. See http://www.computerhope.com/issues/ch000549.htm for instructions on editing the path environment variable.

The `-U backup` indicates that the `backup` user is to carry out the task. The `-F c` option archives in a suitable format for input into the `pg_restore` command. Further information on `pg_dump` and backing up a PostgreSQL database can be found here: http://www.postgresql.org/docs/9.3/static/app-pgdump.html and here: http://www.postgresql.org/docs/9.3/static/backup-dump.html

1.5.4 Restoring a PostgreSQL database (Windows)

The `pg_restore` command can be used to restore the database using one of the backed-up SQL files that were produced using the `pg_dump` command.

Use the `pgAdminIII` tool to ensure that there is a PostgreSQL user called `openremuser`.

Use `pgAdminIII` to create a database called `openrempostgresql`; set the owner to `openremuser` and the encoding to `UTF8`.

Run the following command in a command prompt window (Windows) or terminal window (Ubuntu) to restore the contents of `backupFile` to the `openrempostgresql` database, where `backupFile` is the file created by the `pg_dump` command:

```
pg_restore -U postgres -d openrempostgresql backupFile
```

Ensure that `openremuser` has an entry in PostgreSQL's `pg_hpa.conf` file for md5 authentication:

```
local all openremuser md5
```

The PostgreSQL server will need to be restarted if you have changed `pg_hpa.conf`.

See http://www.postgresql.org/docs/9.3/static/backup-dump.html#BACKUP-DUMP-RESTORE for further details.

1.5.5 Backing up MySQL on Windows

Note: Content contributed by DJ Platten

These instructions are based on Windows XP.

As a one-off, create a MySQL user called `backup` with a password of `backup` that has full rights to the database:

```
C:\Program Files\MySQL\MySQL Server 5.6\bin\mysql.exe -u root -p -e "CREATE USER
↪'backup'@'localhost' IDENTIFIED BY 'backup'";
```

Grant the `backup` user full privileges on the database called `openremdatabasemysql`:

```
C:\Program Files\MySQL\MySQL Server 5.6\bin\mysql.exe -u root -p -e "GRANT ALL␣
↪PRIVILEGES ON openremdatabasemysql .* TO 'backup'@'localhost'";
```

Grant the `backup` user privileges to create databases:

```
C:\Program Files\MySQL\MySQL Server 5.6\bin\mysql.exe -u root -p -e "GRANT CREATE ON␣
↪*.* TO 'backup'@'localhost'";
```

Reload the privileges to ensure that MySQL registers the new ones:

```
C:\Program Files\MySQL\MySQL Server 5.6\bin\mysql.exe -u root -p -e "FLUSH PRIVILEGES
↪";
```

To backup the contents of the database from the command line to a file called `backup.sql` (note that the lack of spaces after the `-u` and `-p` is not a typo):

```
C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqldump.exe -ubackup -pbackup␣
↪openremdatabasemysql > backup.sql
```

To restore the database, assuming that it doesn't exist anymore, first it needs to be created:

```
C:\Program Files\MySQL\MySQL Server 5.6\bin\mysql.exe -ubackup -pbackup -e "CREATE␣
↪DATABASE openremdatabasemysql";
```

Then restore the contents of the database from a file called `backup.sql`:

```
C:\Program Files\MySQL\MySQL Server 5.6\bin\mysql.exe -ubackup -pbackup␣
↪openremdatabasemysql < backup.sql
```

An example DOS batch file to back up the contents of the `openremdatabasemysql` database using a time stamp of the form `yyyy-mm-dd_hhmm`, zip up the resulting file, delete the uncompressed version and then copy it to a network location (the network copy will only work if the user that runs the batch file has permission on the network):

```
@echo off
For /f "tokens=1-4 delims=/ " %%a in ('date /t') do (set mydate=%%c-%%b-%%a)
For /f "tokens=1-2 delims=/:" %%a in ('time /t') do (set mytime=%%a%%b)

"C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqldump.exe" -ubackup -pbackup␣
↪openremdatabasemysql > F:\OpenREMdatabase\backup\%mydate%_%mytime%_
↪openremdatabasemysql.sql

"C:\Program Files\7-Zip\7z.exe" a F:\OpenREMdatabase\backup\%mydate%_%mytime%_
↪openremdatabasemysql.zip F:\OpenREMdatabase\backup\%mydate%_%mytime%_
↪openremdatabasemysql.sql

del F:\OpenREMdatabase\backup\%mydate%_%mytime%_openremdatabasemysql.sql

copy F:\OpenREMdatabase\backup\%mydate%_%mytime%_openremdatabasemysql.zip "\\Srv-mps-
↪001\xls_protect\PATDOSE\OpenREMbackup\"
```

1.6 Web servers

Unlike the database, the production webserver can be left till later and can be changed again at any time. However, for performance it is recommended that a production webserver is used instead of the inbuilt 'runserver'.

On Windows or Linux, it is possible to use Apache, however for reasons relating to how Python, Apache and mod_wsgi are compiled using old Microsoft tools, this is now nearly impossible to do on the Windows platform. There is no reason for existing Windows installs with Apache to change webserver.

For Apache installs on Linux, the django website has instructions and links to get you set up.

Our recommendations for Windows and Linux are:

1.6.1 Running OpenREM on Windows with IIS

These instructions are for running OpenREM under IIS on Windows Server 2012, but should work on Windows 7/8/10 and later versions of Windows Server with minimal modification. The instructions are based on http://blog.mattwoodward.com/2016/07/running-django-application-on-windows.html

Why use IIS?

The built-in Django webserver is not advised for production environments. There are a few alternatives for serving a Django application. Apache is probably the best known web server, but has some requirements under Windows in combination with Python / Django that are hard to fulfill (see https://github.com/GrahamDumpleton/mod_wsgi/blob/develop/win32/README.rst). IIS is default available on Windows Server and doesn't have these requirements.

Prerequisites

- **A working OpenREM installation, that serves web pages using the built-in web server.** You can test this using the instructions in *Start all the services*

Steps to perform

Install wfastcgy Python package

- Open a command prompt and navigate to the pythonscript directory if that directory is not in your path
- Type `pip install wfastcgi`
- Close the command prompt by entering `exit`

Install IIS

- Open Control panel
- Take care the `view by` is on small or large icons (not category view)
- Select Program and Features
- Select Turn Windows features on or off
- **Windows Server 2012**
  - Click `Next`
  - Leave the `Role-based or feature-based installation` radio button selected and click `Next`
  - Leave the current server highlighted and click `Next`
  - Scroll to the bottom of the list and check `Web Server (IIS)`
  - In the dialog that appears, leave the `Include management tools (if applicable)` checkbox checked and click `Add Features`
  - On the `Select server roles` step, now that `Web Server (IIS)` is checked, click `Next`
  - Leave the defaults and click `Next`
  - On the `Web Server Role (IIS)` step, click `Next`
  - On the `Select role services` step, scroll down to `Application Development`, expand that section, and check the `CGI` box. click `Next`.
  - Click `Install` and after installation, click `Close`
- **Window 7 & Windows 10**
  - Tick `Internet information services` to turn the default features on
  - In `World Wide Web Services, Application Development Features`, tick `CGI`
  - Click `OK`
  - Click `Close` (Windows 10 only, In Windows 7 the `Windows Features` window will close automatically)

- Close the Server Manager and the Control Panel

To test the IIS installation browse to http://localhost. You should see the default IIS "Welcome" Page.

Configure IIS

- Open Administrative Tools and double-click the Internet Information Services (IIS) Manager link. You may need to right click and `Run as Adminstrator`. - For Windows 7 the `Administrative Tools` are not available by default, you can also type `inetmgr` in the `search program and files` text box.
- Click on the name of the server within the IIS manager
- Click No if a pop-up about the Web Platform Components appears
- Double-click on the `FastCGI Settings` icon
- In the right pane under `actions` click `Add Application...`
- In the `Full Path` box type the path to the Python executable, e.g.: `c:\python27\python.exe`
- In the `Arguments` box type the path to wfastcgi.py file, e.g.: `c:\python27\Lib\site-packages\wfastcgi.py`.

Note: If your path contains spaces be sure to have double quotes(`"`) around the argument

---

- Under `FastCGI properties`, click on `(Collection)` next to `Environment Variables` and click on the grey `...` box
- In the EnvironmentVariables Collection Editor dialog, click `Add`
- Under `Name properties` on the right, click the input box to the right of `Name`, and replace the text `Name` by `DJANGO_SETTINGS_MODULE` (capitals is important)
- As `Value` enter `openremproject.settings`
- Click Add again and add a variable with name `PYTHON_PATH` and value the path to the openrem path, e.g. `C:\Python27\Lib\site-packages\openrem`
- Click Add for the third time and a variable with name `WSGI_HANDLER` and value `django.core.wsgi.get_wsgi_application()`
- Click `Ok` to close the `EnviromentVariables Collection Editor`.
- Under `FastCGI properties` find the `Activity Timeout` entry and increase the value to 300. This is to ensure that the server allows enough time for skin dose map data to be calculated.
- Click `Ok` to close the `Add FastCGI Application dialog`
- Start Windows Explorer and browse to the Python folder, e.g. `C:\Python27`, or if using a virtualenv browse to that
- Right click on the `Lib` sub-folder and choose properties. In the security tab, make sure `IIS_IUSRS` has `Read`, `Write`, `Execute` and `Change` permissions for this directory and the subdirectories.
- Repeat the above step for the `Scripts` sub-folder, and also for the `MEDIA_ROOT` folder (as configured in `local_settings.py`; default `c:/Temp/OpenREM/media`)

Create a new website

- In the IIS manager under connections expand the tree under server name
- Right-click on sites and click `Add Website...`
- Enter as sitename `OpenREM`
- As physical path enter the same path as the `PYTHON_PATH` in the `FastCGI` settings above, e.g. `C:\Python27\Lib\site-packages\openrem`
- Set the port to the port you desire. If you wish to use the default port 80, you need to stop and/or remove the default website or change the port of the default website
- Click `OK`

Configure the new website

- In IIS manager **double** click on the OpenREM website under Sites
- Double click on the `Handler Mappings` icon in the middle pane
- In the right pane, under `Actions`, click `Add Module Mappings` or `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 `path\to\python.exe|path\to\wfastcgi.py`, e.g.: `c:\python27\python.exe|c:\python27\Lib\site-packages\wfastcgi.py`. The `|` character between the two paths is usually to be found with `Shift \`.
- In `Name` type `OpenREM cgi handler` (value of name is not important)

Note: If one of your paths contains a space use quotations marks around that path. Don't use quotations marks around the full statement.

- Click the `Request Restrictions` button and uncheck the `Invoke handler only if request is mapped to:` checkbox
- Click `Ok` twice to close the Request Restrictions dialog and the Add Module Mapping dialog
- When prompted `Do you want to create a FastCGI application for this executable?` click `No`

The website should work now: browse to http://localhost:port (port is the number you configured the website on. If the port is 80, you can omit the colon and port number).

Note: The website will look "ugly" as the static files (like the css-files) are not yet configured

Configure Django and IIS to serve static files

- Create a directory called `static` in your openrem directory, e.g. `C:\Python27\Lib\site-packages\openrem\static`

- In the Openrem `local_settings.py` file, located in the openremproject directory (e.g. `C:\Python27\Lib\site-packages\openrem\oprenremproject\local_settings.py`) find the `STATIC_ROOT` variable and set the value to match the directory you just created. The backslashes should be replaced by forward slashed. e.g. `STATIC_ROOT = 'C:/Python27/Lib/site-packages/openrem/static'`
- Open a command prompt and navigate to the openrem directory, e.g. `C:\Python27\Lib\site-packages\openrem`
- Type `python manage.py collectstatic`
- Type `Yes` to confirm if the static root directory mentioned is correct
- Close the command prompt by typing `exit`
- In IIS right-click on the OpenREM website (under Sites)
- Click `Add Virtual Directory`
- Type `static` as alias and the path to the static directoy as `Physical Path`, e.g. `C:\Python27\Lib\site-packages\openrem\static`
- Click `Ok` to close the dialog box
- Click on the `static` directory in IIS within the OpenREM site (unfold the OpenREM site)
- Double click on the `Handler Mappings` icon in the middle pane
- On the right pane click `View Ordered Lists...` under Actions
- Click on the `StaticFile Handler` in the middle pane and on `Move Up` in the right pane until the `StaticFile Handler` is on the top

Note: You may get a warning that you are detaching the virtual directory. Click `Yes` on this warning.

Check the website by browsing to http://localhost:port, everything should be fine now.

1.6.2 Running OpenREM on Linux with Gunicorn and NGINX

These instructions are for running OpenREM with Gunicorn and NGINX on Ubuntu Server, but should work on other Linux distributions with minimal modification. These instructions are based on a guide at obeythetestingoat.com

Prerequisites

- A working OpenREM installation, that serves web pages using the built-in web server. You can test this using the instructions in *Start all the services*

Contents

Steps to perform

Note: If you get stuck somewhere in these instructions, please check the *Troubleshooting and tips* section at the end of this page.

Install NGINX

```
sudo apt install nginx
sudo systemctl start nginx
```

You should now be able to see the 'Welcome to nginx' page if you go to your server address in a web browser.

Create initial nginx configuration

Create a new config file - you can name the file as you like, but it is usually has the server name.

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

Start with the following settings - replace the server name with the hostname of your server. For this example, I will use `openrem-server`

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

    location / {
        proxy_pass http://localhost:8000;
    }
}
```

Save and exit (see *nano* below for tips).

Now delete the default nginx configuration from `sites-enabled` and make a link to our new one:

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

Now we reload nginx and start our server as before to test this step. At this stage, nginx is simply passing requests to the default port (80) on to port 8000 to be dealt with (which is why we need to start the test server again).

```
sudo systemctl reload nginx
# activate your virtual environment if you are using one
# navigate to the openrem folder with manage.py in
python manage.py runserver
```

Now use your web browser to look at your server again - the 'Welcome to nginx' page should be replaced by an ugly version of the OpenREM website - this is because the 'static' files are not yet being served - we'll fix this later.

Replace runserver with Gunicorn

Activate your virtualenv if you are using one (add sudo if your aren't), and:

```
pip install gunicorn
```

Make sure you have stopped the test webserver (`Ctrl-c` in the shell `runserver` is running in), then from the same openrem folder:

```
gunicorn openremproject.wsgi:application
```

The Gunicorn server should start, and you should be able to see the same broken version of the web interface again.

Serve static files using nginx

Create a folder called `static` somewhere that your webserver user will be able to get to - for example alongside the `media` folder, and set the permissions. So if you created your media folder in `/var/openrem/media`, you might do this:

```
sudo mkdir /var/openrem/static
sudo chown $USER:www-data /var/openrem/static
sudo chmod 775 /var/openrem/static
```

Now edit your `openremproject/local_settings.py` config file to put the same path in the `STATIC_ROOT`:

```
nano local_settings.py

# Find the static files section
STATIC_ROOT = '/var/openrem/static/'  # replacing path as appropriate
```

Now use the Django `manage.py` application to pull all the static files into the new folder:

```
python manage.py collectstatic
```

Now we need to tell nginx to serve them:

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

And modify the file to add the `static` section - remember to put the path you have used instead of `/var/openrem/static`

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

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

    location / {
        proxy_pass http://localhost:8000;
    }
}
```

Now reload nginx and gunicorn to see if it is all working…

```
sudo systemctl reload nginx
# activate your virtual environment if you are using one
# navigate to the openrem folder with manage.py in
gunicorn openremproject.wsgi:application
```

Take another look, and it should all be looking nice now!

Switch to using Unix Sockets

This step is optional, but does allow you more flexibility if you need to do anything else on this server using port 8000 as this installation of OpenREM will no longer be using that port. Instead we'll use 'sockets', which are like files on the disk. We put these in `/tmp/`.

Change the nginx configuration again (`sudo nano /etc/nginx/sites-available/openrem-server`):

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

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

    location / {
        proxy_pass http://unix:/tmp/openrem-server.socket;
    }
}
```

Now restart Gunicorn, this time telling it to use the socket, after reloading nginx:

```
sudo systemctl reload nginx
gunicorn --bind unix:/tmp/openrem-server.socket \
openremproject.wsgi:application
```

The \ just allows the command to spread to two lines - feel free to put it all on one line.

Check the web interface again, hopefully it should still be working!

Start Gunicorn automatically

We can use systemd on Ubuntu to ensure Gunicorn starts on boot and restarts if it crashes. As before, change each instance of `openrem-server` for the name of your server. You will need to change the `WorkingDirectory` to match the path to your openrem folder.

For the gunicorn command, you will need to provide the full path to gunicorn, whether that is in `/usr/local/bin/gunicorn` or the bin folder of your virtualenv.

```
# Customise the name of the file as you please - it must end in .service
 sudo nano /etc/systemd/system/gunicorn-openrem.service
```

```
[Unit]
Description=Gunicorn server for openrem-server

[Service]
Restart=on-failure
User=www-data
WorkingDirectory=/usr/local/lib/python2.7/dist-packages/openrem

ExecStart=/usr/local/bin/gunicorn \
    --bind unix:/tmp/openrem-server.socket \
    openremproject.wsgi:application

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

Make sure you have customised the `WorkingDirectory` path, the path to gunicorn, and the name of the socket file. The `User` would normally be `www-data`.

Warning: If the user you have configured can't write to the `STATIC_ROOT` folder, the `MEDIA_ROOT` folder and the location the logs are configured to be written (usually in `MEDIA_ROOT`), the systemd gunicorn service is likely to fail when started.

Now enable the new configuration:

```
# Load to config
sudo systemctl daemon-reload
# Enable start on boot - change the name as per how you created it
sudo systemctl enable gunicorn-openrem.service
# Now start the service
sudo systemctl start gunicorn-openrem.service
```

You might like to see if it worked...

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

Look for `Active: active (running)`

Making use of ALLOWED_HOSTS in local_settings.py

The default setting of `ALLOWED_HOSTS` is `*` which isn't secure, but is convenient! We should really change this to match the hostname of the server.

If your hostname is `openrem-server`, and the fully qualified domain name is `openrem-server.ad.hospital.org` and IP address is `10.212.18.209`, then you might configure `ALLOWED_HOSTS` in `openremproject/local_settings.py` to:

```
ALLOWED_HOSTS = [
    'openrem-server',
    'openrem-server.ad.hospital.org',
    '10.212.18.209',
]
```

Note: Which hostnames do I need to put in `ALLOWED_HOSTS`? You need to put in any hostnames you want people to be able to access your OpenREM web interface at. So if in your hospital you only type in the address bar the hostname (`http://openrem-server` in this example), then that is all you need to add. If you only use the IP address, then add that. If you can use any of them, add them all :-)

Next we need to edit the nginx configuration again to make sure Django can see the hostname by adding the `proxy_set_header` configuration (else it gets lost before Django can check it):

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

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

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

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

Now reload the nginx configuration and reload Gunicorn:

```
sudo systemctl reload nginx
sudo systemctl restart gunicorn-openrem.service
```

And check the web interface again. If it doesn't work due to the `ALLOWED_HOSTS` setting, you will get a 'Bad request 400' error.

Increasing the timeout

You may wish to do this to allow for *Skin dose maps* that can take more than 30 seconds for complex studies. Both Gunicorn and nginx configurations need to be modified:

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

Add the `--timeout` setting to the end of the `ExecStart` command, time is in seconds (300 = 5 minutes, 1200 = 20 minutes)

```
[Unit]
Description=Gunicorn server for openrem-server

[Service]
Restart=on-failure
User=www-data
WorkingDirectory=/usr/local/lib/python2.7/dist-packages/openrem

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

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

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

Add the `proxy_read_timeout` setting in seconds (note the trailing `s` this time).

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

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

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

Reload everything:

```
sudo systemctl daemon-reload
sudo systemctl restart gunicorn-openrem.service
sudo systemctl reload nginx
```

Note: If you have jumped straight to here to get the final config, then make sure you substitute all the following values to suit your install:

- `gunicorn-openrem.service` - name not important (except the `.service`, but you need to use it in the reload commands etc
- `User=www-data` as appropriate. This should either be your user or `www-data`. You will need to ensure folder permissions correspond
- `WorkingDirectory` needs to match the path to your `openrem` folder (the one with `manage.py` in)
- `ExecStart=/usr/local/bin/gunicorn \` needs to match the path to your `gunicorn` executable - either in your virtualenv bin folder or system wide as per the example
- `--bind unix:/tmp/openrem-server.socket \` name in `tmp` doesn't matter, needs to match in gunicorn and nginx configs

- `/etc/nginx/sites-available/openrem-server` ie name of config file in nginx, doesn't matter, usually matches hostname
- `server_name openrem-server` - should match hostname
- `/var/openrem/static` folder must exist, with the right permissions. Location not important, must match setting in `local_settings`
- `proxy_pass http://unix:/tmp/openrem-server.socket;` must match setting in gunicorn config, prefixed with `http://`

You will also need to `collectstatic`, symlink the nginx configuration into enabled, enable the gunicorn systemd config to start on reboot, and you should configure the `ALLOWED_HOST` setting. And you will need to have installed nginx and gunicorn!

Troubleshooting and tips

less

Use `less` to review files without editing them

- Navigate using arrow keys, page up and down,
- `Shift-G` to go to the end
- `Shift-F` to automatically update as new logs are added. `Ctrl-C` to stop.
- `/` to search

nano

Use `nano` to edit the files.

- `Ctrl-o` to save ('out')
- `Ctrl-x` to exit

Nginx

- Logs are located in `/var/log/nginx/`
- You need root privileges to view the files:
  - To view latest error log: `sudo less /var/log/nginx/error.log`
- Reload: `sudo systemctl reload nginx`
- Check nginx config: `sudo nginx -t`

Systemd and Gunicorn

- Review the logs with `sudo journalctl -u gunicorn-openrem` (change as appropriate for the the name you have used)
- Check the systemd configuration with `systemd-analyze verify /etc/systemd/system/gunicorn-openrem.service` - again changing the name as appropriate.

- If you make changes, you need to use `sudo systemctl daemon-reload` before the changes will take effect.
- Restart: `sudo systemctl restart gunicorn-openrem.service`

If you want to run OpenREM-webinterface in a virtual directory (other than the root virtual directory), take a look at the following documentation before setting up the web server:

1.6.3 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 application that you are using (IIS or nginx). Next to that you also need to configure this in OpenREM. The following steps are necessary:

- Configure virtual directory in `local_settings.py`
- Update the `reverse.js` file

Configure virtual directory in local_settings.py

Django should know in what virtual directory you are running OpenREM. Perform the following steps to do this.

- In the OpenREM `local_settings.py` file, located in the openremproject directory (e.g. `C:\Python27\Lib\site-packages\openrem\oprenremproject\local_settings.py`) find the `VIRTUAL_DIRECTORY` variable - if there isn't one, somewhere in the `local_settings.py` file add `VIRTUAL_DIRECTORY=''` at the start of a line.
- Set this variable to the desired virtual directory
- Add under this line the following code to set the `STATIC_URL` variable

  ```
  STATIC_URL = '/' + os.path.join(VIRTUAL_DIRECTORY, STATIC_URL.
  ↪lstrip('/'))
  ```

- In order to make this command work `os` has to be imported, add in `local_settings.py` as third line

  ```
  import os
  ```

- Instead of the above two changes, you can also put a hard-coded STATIC_URL as follows

  ```
  STATIC_URL = '/VIRTUAL_DIRECTORY/static/'
  ```

  (replace `VIRTUAL_DIRECTORY` by the actual value):

Note:

- Take care the virtual directory name ends with a slash (`/`)
- Take care the virtual directory name is exactly the same as configured in the web server (this is case-sensitive)

Update reverse.js

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

- Open a command prompt and navigate to the openrem directory, e.g. `C:\Python27\Lib\site-packages\openrem`
- Type `python manage.py collectstatic_js_reverse`

Note: Take care the resulting `reverse.js` is written to the correct static directory. If that is not the case copy the file manually to the correct location.

1.7 Quick start: One page complete install

1.7.1 One page complete Ubuntu install

A one page install based on Ubuntu 18.04 using:

- Python 2.7 running in a virtualenv
- Database: PostgreSQL
- DICOM Store SCP: Orthanc running on port 104
- Webserver: NGINX with Gunicorn
- Daemonisation: systemd scripts for Celery and Gunicorn
- All OpenREM files in `/var/dose/` with group owner of `openrem`
- Collects any Physics (QA) images and zips them

Initial prep

Host file

First edit `/etc/hosts` to add the local server name – else `rabbitmq-server` will not start when installed:

```
sudo nano /etc/hosts
```

Modify the content to ensure the following two lines are present – **substitute the correct server hostname on the second line**:

```
127.0.0.1 localhost
127.0.1.1 openremserver
```

`Ctrl-o` (write-[o]ut), `Return` to accept, then `Ctrl-x` to e[x]it

Apt sources

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

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

Look for:

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

If these two lines are not there, add them in (`sudo nano /etc/apt/sources.list`).

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
```

Note: At a later stage, to add a second administrator just add them to the `openrem` group in the same way.

Folders

Create the folders we need, and set the permissions. In due course, the `orthanc` user and the `www-data` user will be added to the `openrem` group, and the 'sticky' group setting below will enable both users to write to the logs etc:

```
sudo mkdir /var/dose
sudo chown $USER:openrem /var/dose
sudo chmod 775 /var/dose
cd /var/dose
mkdir celery
mkdir log
mkdir media
mkdir -p orthanc/dicom
mkdir -p orthanc/physics
mkdir pixelmed
mkdir static
mkdir veopenrem
sudo chown -R $USER:openrem /var/dose/*
sudo chmod -R g+s /var/dose/*
sudo setfacl -R -dm u::rwx,g::rwx,o::r /var/dose/
```

Install apt packages and direct downloads

The \ just allows the `sudo apt install` command to spread to two lines – feel free to put it all on one line.

```
sudo apt update
sudo apt upgrade
sudo apt install python python-pip virtualenv rabbitmq-server \
postgresql nginx orthanc dcmtk default-jre zip

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

Create the virtualenv

Create a virtualenv (Python local environment) in the folder we created:

```
virtualenv /var/dose/veopenrem
```

Activate the virtualenv

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

```
. /var/dose/veopenrem/bin/activate
```

Install Python packages

```
pip install numpy psycopg2-binary gunicorn
pip install openrem
pip install https://bitbucket.org/edmcdonagh/pynetdicom/get/default.tar.gz
↪#egg=pynetdicom-0.8.2b2
```

Note: There will be error messages when you install pynetdicom from this source. As long as the final line is `Successfully installed pynetdicom-0.8.2b2` then everything is ok!

Add orthanc and www-data users to openrem group

```
sudo adduser orthanc openrem
sudo adduser www-data openrem
```

Database and OpenREM config

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 OpenREM:

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

If you are migrating from another server, you could at this point create a template0 database to restore into. See *Restore the database* for details.

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).

`sudo nano /etc/postgresql/10/main/pg_hba.conf`:

```
local   all     openremuser                     md5
```

Reload postgres:

```
sudo systemctl reload postgresql
```

Configure OpenREM

First navigate to the Python openrem folder and copy the example local_settings and wsgi files to remove the `.example` suffixes:

```
cd /var/dose/veopenrem/lib/python2.7/site-packages/openrem/
cp openremproject/local_settings.py{.example,}
cp openremproject/wsgi.py{.example,}
```

Edit the new local_settings file (`nano openremproject/local_settings.py`):

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'openremdb',
        'USER': 'openremuser',
        'PASSWORD': 'mysecretpassword',      # This needs changing, hopefully!
        'HOST': '',
        'PORT': '',
    }
}

MEDIA_ROOT = '/var/dose/media/'

STATIC_ROOT = '/var/dose/static/'

# Change secret key

# 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"
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")

# 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']['store_file']['class'] = 'logging.handlers.RotatingFileHandler'
LOGGING['handlers']['store_file']['maxBytes'] = 10 * 1024 * 1024  # 10*1024*1024 = 10␣
↪MB
```

(continues on next page)

(continued from previous page)

```
LOGGING['handlers']['store_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

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 -'
```

Now create the database. Make sure you are still in the openrem python folder and the virtualenv is active (prompt will look like `(veopenrem)username@hostname:/var/dose/veopenrem/lib/python2.7/site-packages/openrem/$`). Otherwise see *Activate the virtualenv* and navigate back to that folder:

```
python manage.py makemigrations remapp
python manage.py migrate
python manage.py createsuperuser
mv remapp/migrations/0002_0_7_fresh_install_add_median.py{.inactive,}
python manage.py migrate
```

Webserver

Configure NGINX and Gunicorn

Create the OpenREM site config file `sudo nano /etc/nginx/sites-available/openrem-server`:

```
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
```

Add the static files to the static folder for NGINX to serve. Again, you need to ensure the virtualenv is active in your console and you are in the `site-packages/openrem/` folder:

```
python manage.py collectstatic
```

Create the Gunicorn systemd service file:

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

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

[Service]
Restart=on-failure
User=www-data
WorkingDirectory=/var/dose/veopenrem/lib/python2.7/site-packages/openrem

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

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

Load the new systemd configurations:

```
sudo systemctl daemon-reload
```

Set the new Gunicorn service to start on boot:

```
sudo systemctl enable openrem-gunicorn.service
```

Start the Gunicorn service, and restart the NGINX service:

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

Test the webserver

You should now be able to browse to the OpenREM server from another PC.

You can check that NGINX and Gunicorn are running with the following two commands:

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

Celery and Flower

First, create a Celery configuration file:

`nano /var/dose/celery/celery.conf`:

```
# Name of nodes to start
CELERYD_NODES="default"

```

(continues on next page)

(continued from previous page)

```
# Absolute or relative path to the 'celery' command:
CELERY_BIN="/var/dose/veopenrem/bin/celery"

# App instance to use
CELERY_APP="openremproject"

# How to call manage.py
CELERYD_MULTI="multi"

# Extra command-line arguments to the worker
# Adjust the concurrency as appropriate
CELERYD_OPTS="-O=fair --concurrency=4 --queues=default"

# - %n will be replaced with the first part of the nodename.
# - %I will be replaced with the current child process index
#   and is important when using the prefork pool to avoid race conditions.
CELERYD_PID_FILE="/var/dose/celery/%n.pid"
CELERYD_LOG_FILE="/var/dose/log/%n%I.log"
CELERYD_LOG_LEVEL="INFO"

# Flower configuration options
FLOWER_PORT=5555
FLOWER_LOG_PREFIX="/var/dose/log/flower.log"
FLOWER_LOG_LEVEL="INFO"
```

Now create the systemd service files:

`sudo nano /etc/systemd/system/openrem-celery.service`:

```
[Unit]
Description=Celery Service
After=network.target

[Service]
Type=forking
Restart=on-failure
User=www-data
Group=www-data
EnvironmentFile=/var/dose/celery/celery.conf
WorkingDirectory=/var/dose/veopenrem/lib/python2.7/site-packages/openrem
ExecStart=/bin/sh -c '${CELERY_BIN} multi start ${CELERYD_NODES} \
  -A ${CELERY_APP} --pidfile=${CELERYD_PID_FILE} \
  --logfile=${CELERYD_LOG_FILE} --loglevel=${CELERYD_LOG_LEVEL} ${CELERYD_OPTS}'
ExecStop=/bin/sh -c '${CELERY_BIN} multi stopwait ${CELERYD_NODES} \
  --pidfile=${CELERYD_PID_FILE}'
ExecReload=/bin/sh -c '${CELERY_BIN} multi restart ${CELERYD_NODES} \
  -A ${CELERY_APP} --pidfile=${CELERYD_PID_FILE} \
  --logfile=${CELERYD_LOG_FILE} --loglevel=${CELERYD_LOG_LEVEL} ${CELERYD_OPTS}'

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

`sudo nano /etc/systemd/system/openrem-flower.service`:

```
[Unit]
Description=Flower Celery Service
After=network.target
```

(continues on next page)

(continued from previous page)

```
[Service]
User=www-data
Group=www-data
EnvironmentFile=/var/dose/celery/celery.conf
WorkingDirectory=/var/dose/veopenrem/lib/python2.7/site-packages/openrem
ExecStart=/bin/sh -c '${CELERY_BIN} flower -A ${CELERY_APP} --port=${FLOWER_PORT} \
  --address=127.0.0.1 --log-file-prefix=${FLOWER_LOG_PREFIX} --loglevel=${FLOWER_LOG_
↪LEVEL}'
Restart=on-failure
Type=simple

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

Now register, set to start on boot, and start the services:

```
sudo systemctl daemon-reload
sudo systemctl enable openrem-celery.service
sudo systemctl start openrem-celery.service
sudo systemctl enable openrem-flower.service
sudo systemctl start openrem-flower.service
```

DICOM Store SCP

Open the following link in a new tab and copy the content (select all then Ctrl-c):

Create the lua file to control how we process the incoming DICOM objects and paste the content in (Shift-Ctrl-v if working directly in the Ubuntu terminal, something else if you are using PuTTY etc):

`nano /var/dose/orthanc/openrem_orthanc_config.lua`

Then edit the top section as follows – keeping Physics test images has been configured, set to false to change this. There are other settings too that you might like to change in the second section (not displayed here):

```
-------------------------------------------------------------------------------------
-- OpenREM python environment and other settings

-- Set this to the path and name of the python executable used by OpenREM
local python_executable = '/var/dose/veopenrem/bin/python'

-- Set this to the path of the python scripts folder used by OpenREM
local python_scripts_path = '/var/dose/veopenrem/bin/'

-- Set this to the path where you want Orthanc to temporarily store DICOM files
local temp_path = '/var/dose/orthanc/dicom/'

-- Set this to 'mkdir' on Windows, or 'mkdir -p' on Linux
local mkdir_cmd = 'mkdir -p'

-- Set this to '\\'' on Windows, or '/' on Linux
local dir_sep = '/'

-- 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
```

(continues on next page)

(continued from previous page)

```
-- Set this to the path where you want to keep physics-related DICOM images
local physics_to_keep_folder = '/var/dose/orthanc/physics/'

-- Set this to the path and name of your zip utility, and include any switches that
-- are needed to create an archive (used with physics-related images)
local zip_executable = '/usr/bin/zip -r'

-- Set this to the path and name of your remove folder command, including switches
-- for it to be quiet (used with physics-related images)
local rmdir_cmd = 'rm -r'
-------------------------------------------------------------------------------------
```

Add the Lua script to the Orthanc config:

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

```
// 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.lua"
],
```

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 Orthan binary special permission to do so:

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

Then edit the Orthanc configuration again:

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

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

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

Finish off

Restart Orthanc:

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

New users, and quick access to physics folder

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
```

The new user should now be able to get to the physics folder by clicking on the `physicsimages` link when they log in, and should be able to browse, copy and delete the zip files and folders.

Enable RadbbitMQ queue management interface

```
sudo rabbitmq-plugins enable rabbitmq_management
```

Optional – RabbitMQ Administrator

This is not required unless you wish to interact with the RabbitMQ management interface directly. Most functions can be carried out in the OpenREM interface instead. If you do wish to create a user for this purpose, see the general instructions to *Enable RabbitMQ queue management interface*.

Log locations

- OpenREM: `/var/dose/log/`
- Celery: `/var/dose/log/default.log`
- Celery systemd: `sudo journalctl -u openrem-celery`
- NGINX: `/var/log/nginx/`
- Orthanc: `/var/log/orthanc/Orthanc.log`
- Gunicorn systemd: `sudo journalctl -u openrem-gunicorn`

CHAPTER 2

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. For full functionality start the *Celery task queue* before moving on to *Configure the settings*.

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. For full functionality start the *Celery task queue* before moving on to *Configure the settings*.

Note: 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 Celery task queue

Celery will have been automatically installed with OpenREM, and along with RabbitMQ allows for asynchronous task processing for imports, exports and DICOM networking tasks.

Note: Celery needs to be able to write to the place where the Celery logs and pid file are to be stored, so make sure:

- the folder exists (the suggestion below is to create a folder in the `MEDIA_ROOT` location)
- the user that starts Celery can write to that folder

You can put the folder wherever you like, for example you might like to create a `/var/log/openrem/` folder on a linux system.

If you are using the built-in *Test web server* then Celery and the webserver will be running as your user. If you are running a production webserver, such as Apache or nginx on linux, then the user that runs those daemons will need to be able to write to the `MEDIA_ROOT` and the Celery log files folder. In this case, you need to change the ownership of the folders and change to the right user before running Celery. On Ubuntu:

```
mkdir /path/to/media/celery  # change as appropriate
sudo chown www-data /path/to/media  # change as appropriate
sudo su -p www-data
```

Now start celery...

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)

Linux - \ is the line continuation character:

```
celery multi start default -Ofair -A openremproject -c 4 -Q default \
--pidfile=/path/to/media/celery/%N.pid --logfile=/path/to/media/celery/%N.log
```

Windows - `celery multi` doesn't work on Windows, and ^ is the continuation character:

```
celery worker -n default -Ofair -A openremproject -c 4 -Q default ^
--pidfile=C:\path\to\media\celery\default.pid --
↪logfile=C:\path\to\media\celery\default.log
```

2.2.1 Celery concurrency

Set the number of workers (concurrency, `-c`) according to how many processor cores you have available. The more you have, the more processes (imports, exports, query-retrieve operations etc) can take place simultaneously. However, each extra worker uses extra memory and if you have too many they will be competing for CPU resources too.

Problems with Celery 4 on Windows

Full support for Celery on Windows was dropped with version 4 due to lack of Windows based developers. Therefore for Windows the instructions fix Celery at version `3.1.25` to retain full functionality.

To stop the celery queues in Linux:

```
celery multi stop default --pidfile=/path/to/media/celery/%N.pid
```

For Windows, just press `Ctrl+c`

You will need to do this twice if there are running tasks you wish to kill.

For production use, see *Daemonising Celery* below.

2.3 Celery task management: Flower

Flower will have been automatically installed with OpenREM and enables monitoring and management of Celery tasks.

You should start Flower with the same user that you started Celery with, and put the log file in the same place too.

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)

If you need to change the default port from 5555 then you need to make the same change in `openremproject\local_settings.py` to add/modify the line `FLOWER_PORT = 5555`

If you wish to be able to use the Flower management interface independently of OpenREM, then omit the `--address` part of the command. Flower will then be available from any PC on the network at http://yourdoseservernameorIP:5555/

Linux - \ is the line continuation character:

```
celery flower -A openremproject --port=5555 --address=127.0.0.1  --loglevel=INFO \
---log-file-prefix=/path/to/media/celery/flower.log
```

Windows - ^ is the line continuation character:

```
celery flower -A openremproject --port=5555 --address=127.0.0.1  --loglevel=INFO ^
---log-file-prefix=C:\path\to\media\celery\flower.log
```

For production use, see *Daemonising Celery* below.

2.4 Celery periodic tasks: beat

Note: Celery beat is only required if you are using the *Native DICOM store node with direct import*. Please read the warnings there before deciding if you need to run Celery beat. At the current time, using a third party DICOM store service is recommended for most users. See the *Third-party DICOM Stores* documentation for more details

Celery beat is a scheduler. If it is running, then every 60 seconds a task is run to check if any of the DICOM Store SCP nodes are set to `keep_alive`, and if they are, it tries to verify they are running with a DICOM echo. If this is not successful, then the Store SCP is started.

To run celery beat, open a new shell and 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)

Linux:

```
celery -A openremproject beat -s /path/to/media/celery/celerybeat-schedule \
-f /path/to/media/celery/celerybeat.log \
--pidfile=/path/to/media/celery/celerybeat.pid
```

Windows:

```
celery -A openremproject beat -s C:\path\to\media\celery\celerybeat-schedule ^
-f C:\path\to\media\celery\celerybeat.log ^
--pidfile=C:\path\to\media\celery\celerybeat.pid
```

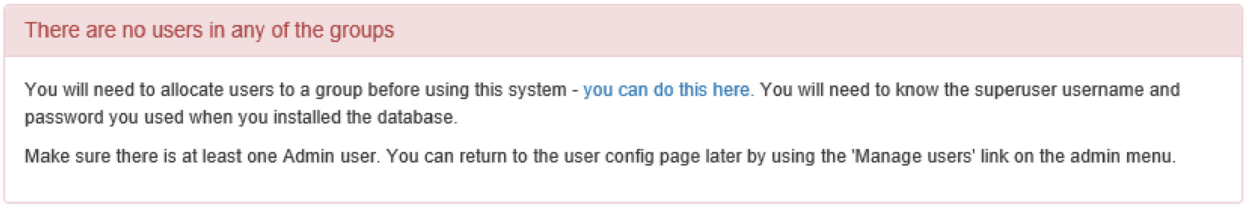
For production use, see *Daemonising Celery* below

As with starting the Celery workers, the folder that the pid, log and for beat, schedule files are to be written **must already exist** and the user starting Celery beat must be able write to that folder.

To stop Celery beat, just press `Ctrl+c`

2.5 Configure the settings

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

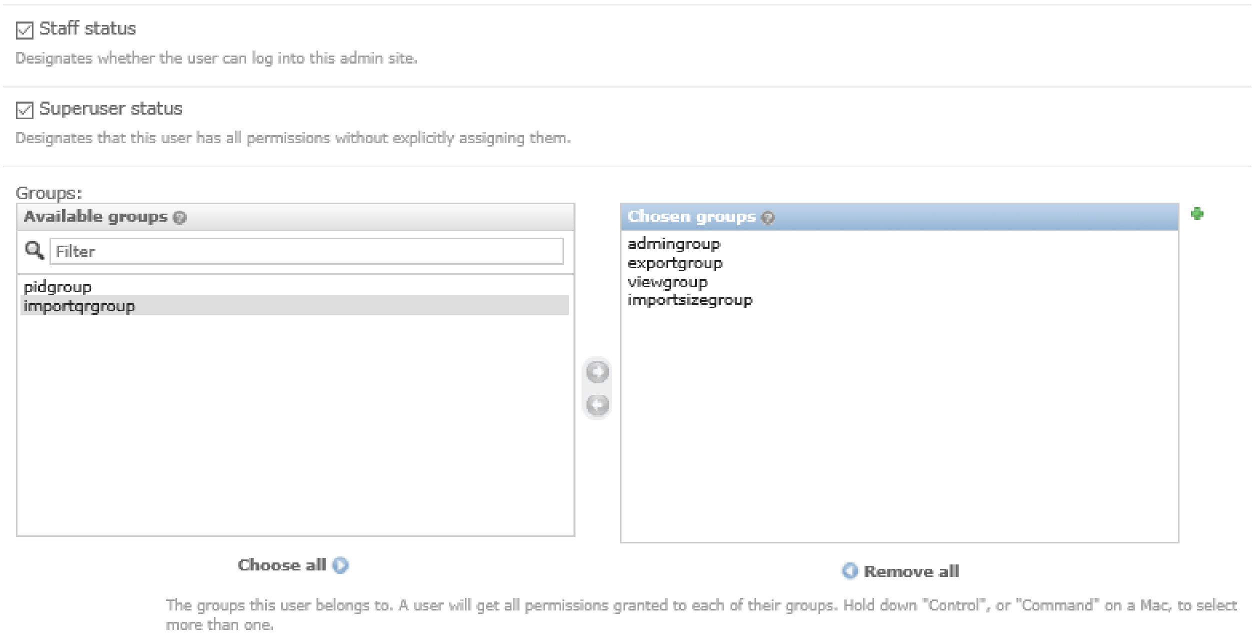


There are no users in any of the groups

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.*



☑ Staff status

Designates whether the user can log into this admin site.

☑ Superuser status

Designates that this user has all permissions without explicitly assigning them.

Groups:

The groups this user belongs to. A user will get all permissions granted to each of their groups. Hold down "Control", or "Command" on a Mac, to select more than one.

- 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)

Welcome, Ed. View site / Change password / Log out

- 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

2.6 Start using it - add some data!

See *Importing data to OpenREM*

2.7 Further instructions

2.7.1 Daemonising Celery

In a production environment, Celery will need to start automatically and not depend on a particular user being logged in. Therefore, much like the webserver, it will need to be daemonised.

Daemonising Celery and Flower on Linux

Guides to daemonising Celery can be found in the Celery documentation at http://docs.celeryproject.org/en/latest/userguide/daemonizing.html.

Alternatively, if you are running Ubuntu 18.04 or another systemd based Linux operating system, the instructions below are taken from the Celery docs but customised for OpenREM.

In this example, the following folders have been created:

- `/var/dose/celery/`
- `/var/dose/log/`
- `/var/dose/veopenrem/`

OpenREM is installed in a virtualenv in `/var/dose/veopenrem/`.

Adjust all the paths as appropriate. If you change the default port from 5555 then you need to make the same change in `openremproject\local_settings.py` to add/modify the line `FLOWER_PORT = 5555`

First, create a Celery configuration file:

`nano /var/dose/celery/celery.conf`:

```
# Name of nodes to start
CELERYD_NODES="default"

# Absolute or relative path to the 'celery' command:
CELERY_BIN="/var/dose/veopenrem/bin/celery"

# App instance to use
CELERY_APP="openremproject"

# How to call manage.py
CELERYD_MULTI="multi"

# Extra command-line arguments to the worker
# Adjust the concurrency as appropriate
CELERYD_OPTS="-O=fair --concurrency=4 --queues=default"

# -
↪ %n will be replaced with the first part of the nodename.
# -␣
↪%I will be replaced with the current child process index
#   and is important␣
↪when using the prefork pool to avoid race conditions.
CELERYD_PID_FILE="/var/dose/celery/%n.pid"
CELERYD_LOG_FILE="/var/dose/log/%n%I.log"
CELERYD_LOG_LEVEL="INFO"

# Flower configuration options
FLOWER_PORT=5555
FLOWER_LOG_PREFIX="/var/dose/log/flower.log"
FLOWER_LOG_LEVEL="INFO"
```

Now create the systemd service files:

`sudo nano /etc/systemd/system/celery-openrem.service`:

```
[Unit]
Description=Celery Service
After=network.target

[Service]
Type=forking
Restart=on-failure
User=www-data
Group=www-data
EnvironmentFile=/var/dose/celery/celery.conf
WorkingDirectory=/
↪var/dose/veopenrem/lib/python2.7/site-packages/openrem
ExecStart=/
↪bin/sh -c '${CELERY_BIN} multi start ${CELERYD_NODES} \
  -A ${CELERY_APP} --pidfile=${CELERYD_PID_FILE} \
  --logfile=${CELERYD_LOG_
↪FILE} --loglevel=${CELERYD_LOG_LEVEL} ${CELERYD_OPTS}'
```

(continues on next page)

(continued from previous page)

```
ExecStop=/bin/
↪sh -c '${CELERY_BIN} multi stopwait ${CELERYD_NODES} \
  --pidfile=${CELERYD_PID_FILE}'
ExecReload=/
↪bin/sh -c '${CELERY_BIN} multi restart ${CELERYD_NODES} \
  -A ${CELERY_APP} --pidfile=${CELERYD_PID_FILE} \
  --logfile=${CELERYD_LOG_
↪FILE} --loglevel=${CELERYD_LOG_LEVEL} ${CELERYD_OPTS}'

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

`sudo nano /etc/systemd/system/flower-openrem.service`:

```
[Unit]
Description=Flower Celery Service
After=network.target

[Service]
User=www-data
Group=www-data
EnvironmentFile=/var/dose/celery/celery.conf
WorkingDirectory=/
↪var/dose/veopenrem/lib/python2.7/site-packages/openrem
ExecStart=/bin/sh -c '${CELERY_
↪BIN} flower -A ${CELERY_APP} --port=${FLOWER_PORT} \
  --address=127.0.0.1 --log-file-prefix=
↪${FLOWER_LOG_PREFIX} --loglevel=${FLOWER_LOG_LEVEL}'
Restart=on-failure
Type=simple

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

Now register, set to start on boot, and start the services:

```
sudo systemctl daemon-reload
sudo systemctl enable celery-openrem.service
sudo systemctl start celery-openrem.service
sudo systemctl enable flower-openrem.service
sudo systemctl start flower-openrem.service
```

Daemonising Celery and Flower on Windows

To ensure that the Celery task queue and Flower are started at system start-up it is advisable to launch them using batch files and configure Windows Task Scheduler to run each of these at system start-up.

Celery will sometimes fall over during the execution of a long task. If Celery frequently falls over on your system then Windows Task Scheduler can be used to restart Celery on a regular basis. The Task Scheduler can also be used to ensure celery is running a few minutes prior to scheduled PACS queries.

An example batch file is shown below for running and restarting Celery. This calls separate batch files to start Celery and Flower, also shown below.

Celery control batch file

celery_task.bat, to be run as a scheduled task.

```
:: Create variables containing␣
↪the name and path of the Celery pid file and the
:: names and␣
↪paths to the batch files used to run Celery and Flower.
SET celeryPidFile=E:\media_root\celery\default.pid
SET celeryStartFile=D:\Server_Apps\celery\celery_start.bat
SET flowerStartFile=D:\Server_Apps\flower\flower_start.bat

:: Celery 3.1.25␣
↪cannot be shutdown gracefully, and has to be killed. The
:: following command␣
↪will kill all celery.exe processes and any python.exe
:: processes associated␣
↪with Celery. The celery.exe process that is running
:: Flower will also be killed by this command.
TASKKILL /IM celery.exe /T /F

:: Force the deletion␣
↪of the Celery pid file so that Celery can be restarted.
DEL /F "%celeryPidFile%"

:: Start Flower again.
START /B CMD /C CALL "%flowerStartFile%"

:: Start Celery again.
START /B CMD /C CALL "%celeryStartFile%"
```

Celery start batch file

celery_start.bat, called by *celery_task.bat*.

```
:: Create variables containing␣
↪the drive and path to OpenREM and the name and
:: path of the Celery pid and log files.
SET openremDrive=D:
SET openremPath=D:\Server_
↪Apps\python27\Lib\site-packages\openrem
SET celeryPidFile=E:\media_root\celery\default.pid
SET celeryLogFile=E:\media_root\celery\default.log

:: Change to the␣
↪drive on which OpenREM is installed and navigate to the
:: OpenREM folder.
%openremDrive%
CD "%openremPath%"

:: Start Celery.
celery worker␣
↪-n default -Ofair -A openremproject -c 4 -Q default␣
↪--pidfile=%celeryPidFile% --logfile=%celeryLogFile%
```

(continues on next page)

(continued from previous page)

Flower start batch file

flower_start.bat, called by *celery_task.bat* and also used to start Flower at system start-up.

```
:: Create variables containing␣
↪the drive and path to OpenREM and the name and
:: path of the Flower log file and the Flower port.
SET openremDrive=D:
SET openremPath=D:\Server_
↪Apps\python27\Lib\site-packages\openrem
SET flowerLogFile=E:\media_root\celery\flower.log
SET flowerPort=5555

:: Change to the␣
↪drive on which OpenREM is installed and navigate to the
:: OpenREM folder.
%openremDrive%
CD "%openremPath%"

:: Start Flower using Celery.
celery -A openremproject flower --port="%flowerPort
↪%" --loglevel=info --log-file-prefix="%flowerLogFile%"
```

Setting up a scheduled task

For Celery

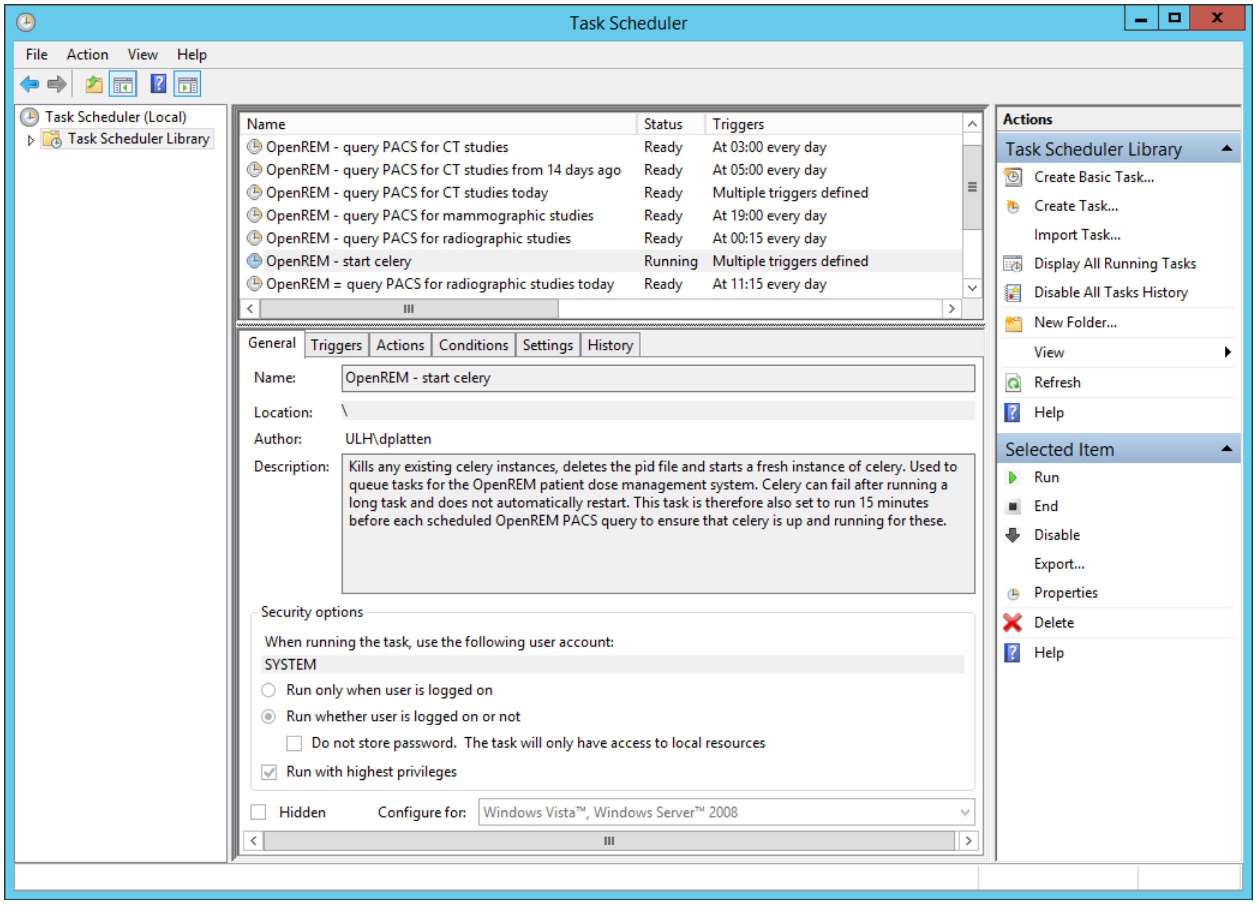
Open `Task Scheduler` on the OpenREM server and then click on the `Task Scheduler Library` item in the left-hand pane. This should look something like figure 1 below, but without the OpenREM tasks present.

To create a new task for celery click on `Create Task...` in the `Actions` menu in the right-hand pane. Give the task a name and description. Next, click on the `Change User or Group` button and type `system` in to the box, then click `Check Names`, then click `OK`. This sets the server's `SYSTEM` user to run the task. Also check the `Run with highest prilileges` box. Your task should now look similar to figure 2.

Next, click on the `Triggers` tab so that you can set when the task will be run. As a minimum you should add an `At startup` trigger. To do this, click `New. ...` In the dialogue box that appears select `At startup` from the `Begin the task` options and ensure that the `Enabled` checkbox is selected. Then click `OK`. You may wish to add other triggers that take place at specific times during the day, as shown in figure 3.

In the example shown in figure 3 celery is started at system start up, and restarted multiple times each day to ensure that it is running before any PACS queries. Your requirements may be more straightforward than this example.

Now click on the `Actions` tab so that you can add the action that is taken when the task is run. Click on `New...`, and in the dialogue box that appears select



Name	Status	Triggers
OpenREM - query PACS for CT studies	Ready	At 03:00 every day
OpenREM - query PACS for CT studies from 14 days ago	Ready	At 05:00 every day
OpenREM - query PACS for CT studies today	Ready	Multiple triggers defined
OpenREM - query PACS for mammographic studies	Ready	At 19:00 every day
OpenREM - query PACS for radiographic studies	Ready	At 00:15 every day
OpenREM - start celery	Running	Multiple triggers defined
OpenREM = query PACS for radiographic studies today	Ready	At 11:15 every day

Name: OpenREM - start celery

Location: \

Author: ULH\dplatten

Description: Kills any existing celery instances, deletes the pid file and starts a fresh instance of celery. Used to queue tasks for the OpenREM patient dose management system. Celery can fail after running a long task and does not automatically restart. This task is therefore also set to run 15 minutes before each scheduled OpenREM PACS query to ensure that celery is up and running for these.

Security options

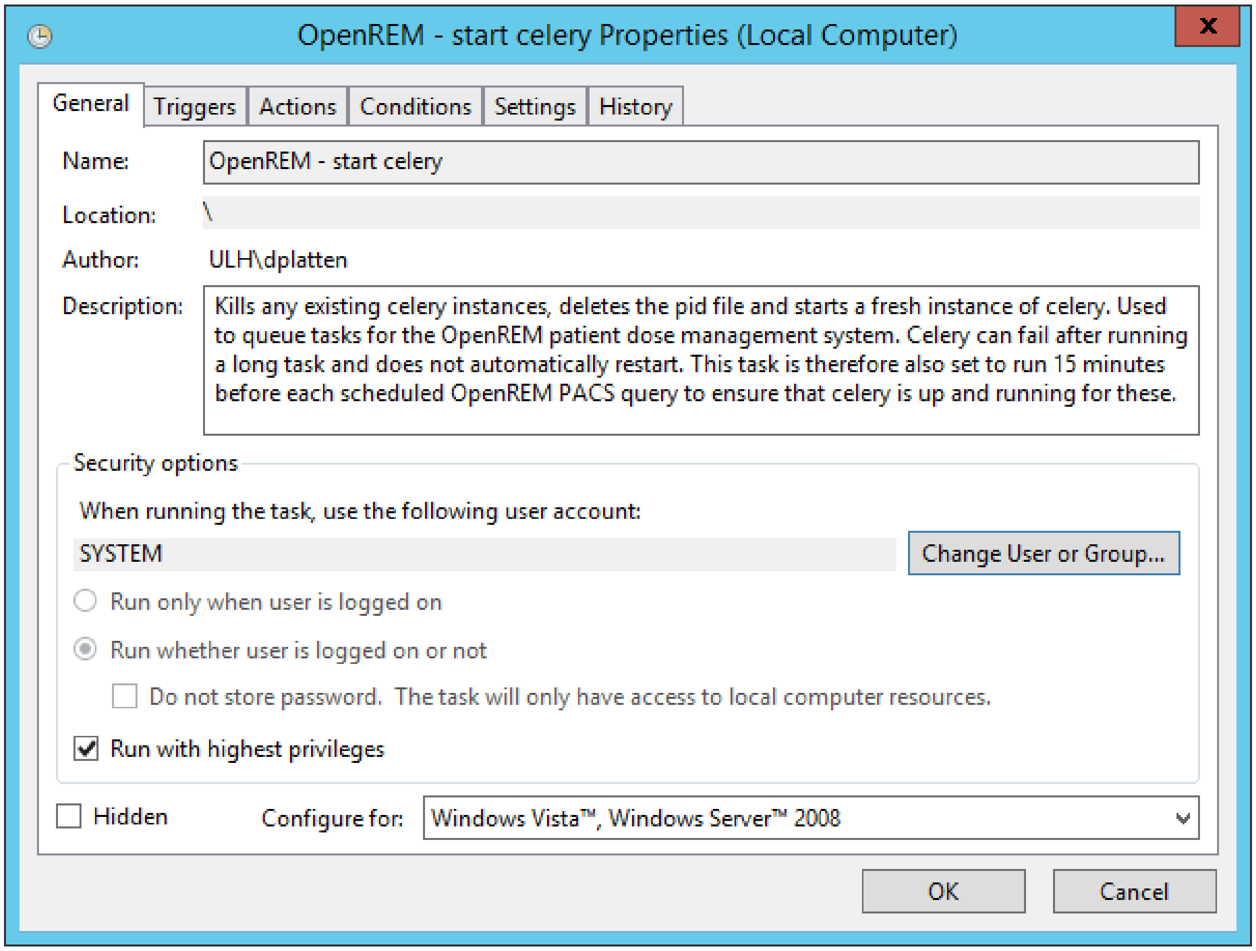
When running the task, use the following user account:

SYSTEM

- ○ Run only when user is logged on
- ● Run whether user is logged on or not
  - ☐ Do not store password. The task will only have access to local resources
- ☑ Run with highest privileges

☐ Hidden   Configure for: Windows Vista™, Windows Server™ 2008

Fig. 1: Figure 1: An overview of Windows Task Scheduler



Name: OpenREM - start celery

Location: \

Author: ULH\dplatten

Description: Kills any existing celery instances, deletes the pid file and starts a fresh instance of celery. Used to queue tasks for the OpenREM patient dose management system. Celery can fail after running a long task and does not automatically restart. This task is therefore also set to run 15 minutes before each scheduled OpenREM PACS query to ensure that celery is up and running for these.

Security options

When running the task, use the following user account:

SYSTEM

○ Run only when user is logged on

● Run whether user is logged on or not

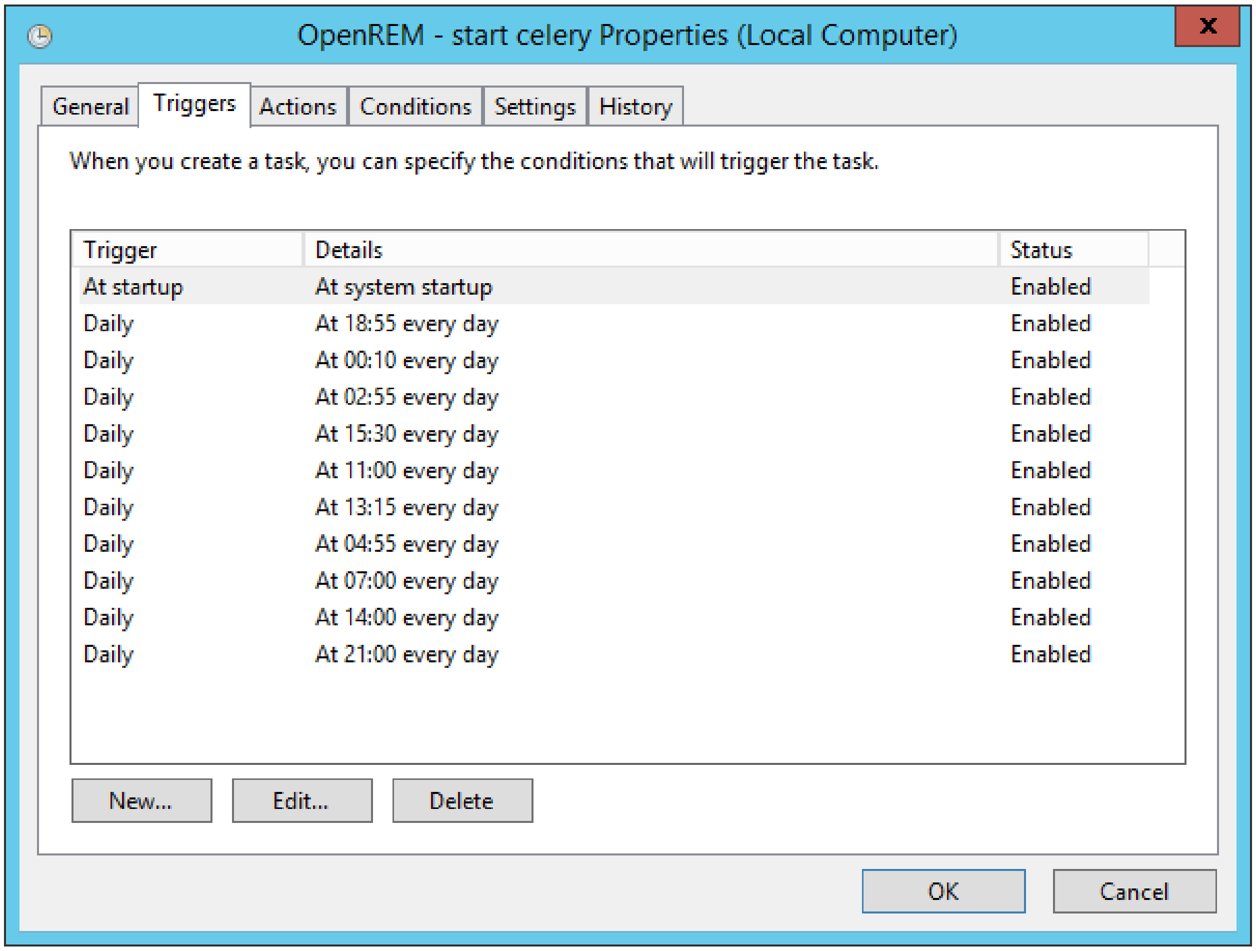
☐ Do not store password. The task will only have access to local computer resources.

☑ Run with highest privileges

☐ Hidden

Configure for: Windows Vista™, Windows Server™ 2008

Fig. 2: Figure 2: General properties

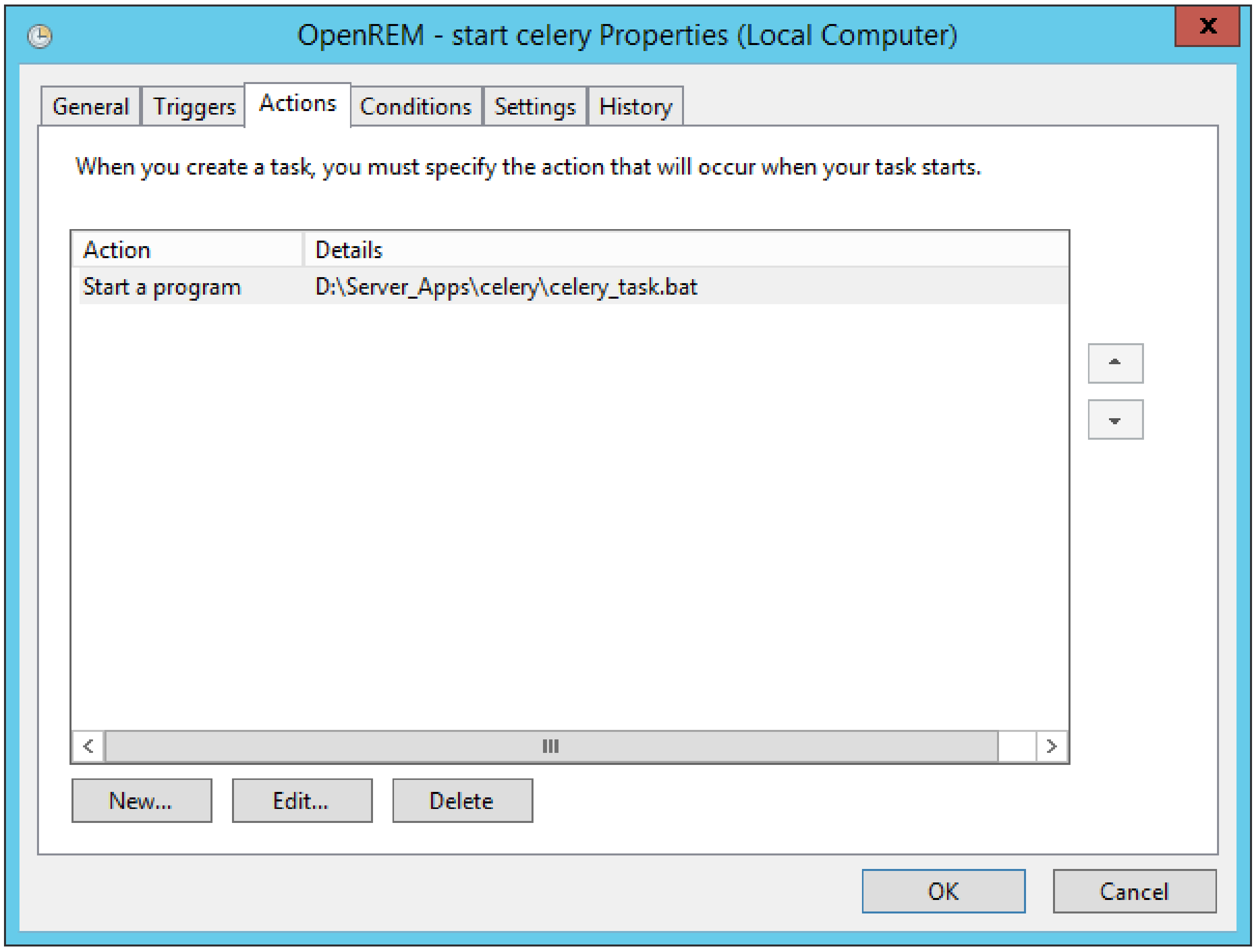

OpenREM - start celery Properties (Local Computer)

When you create a task, you can specify the conditions that will trigger the task.

Trigger	Details	Status
At startup	At system startup	Enabled
Daily	At 18:55 every day	Enabled
Daily	At 00:10 every day	Enabled
Daily	At 02:55 every day	Enabled
Daily	At 15:30 every day	Enabled
Daily	At 11:00 every day	Enabled
Daily	At 13:15 every day	Enabled
Daily	At 04:55 every day	Enabled
Daily	At 07:00 every day	Enabled
Daily	At 14:00 every day	Enabled
Daily	At 21:00 every day	Enabled

Fig. 3: Figure 3: Trigger properties

`Start a program` as the `Action`. Click on `Browse` and select the celery batch file that you created earlier. Click `OK` to close the `New Action` dialogue box. Figure 4 shows an example of the the `Actions` tab.



When you create a task, you must specify the action that will occur when your task starts.

Action	Details
Start a program	D:\Server_Apps\celery\celery_task.bat

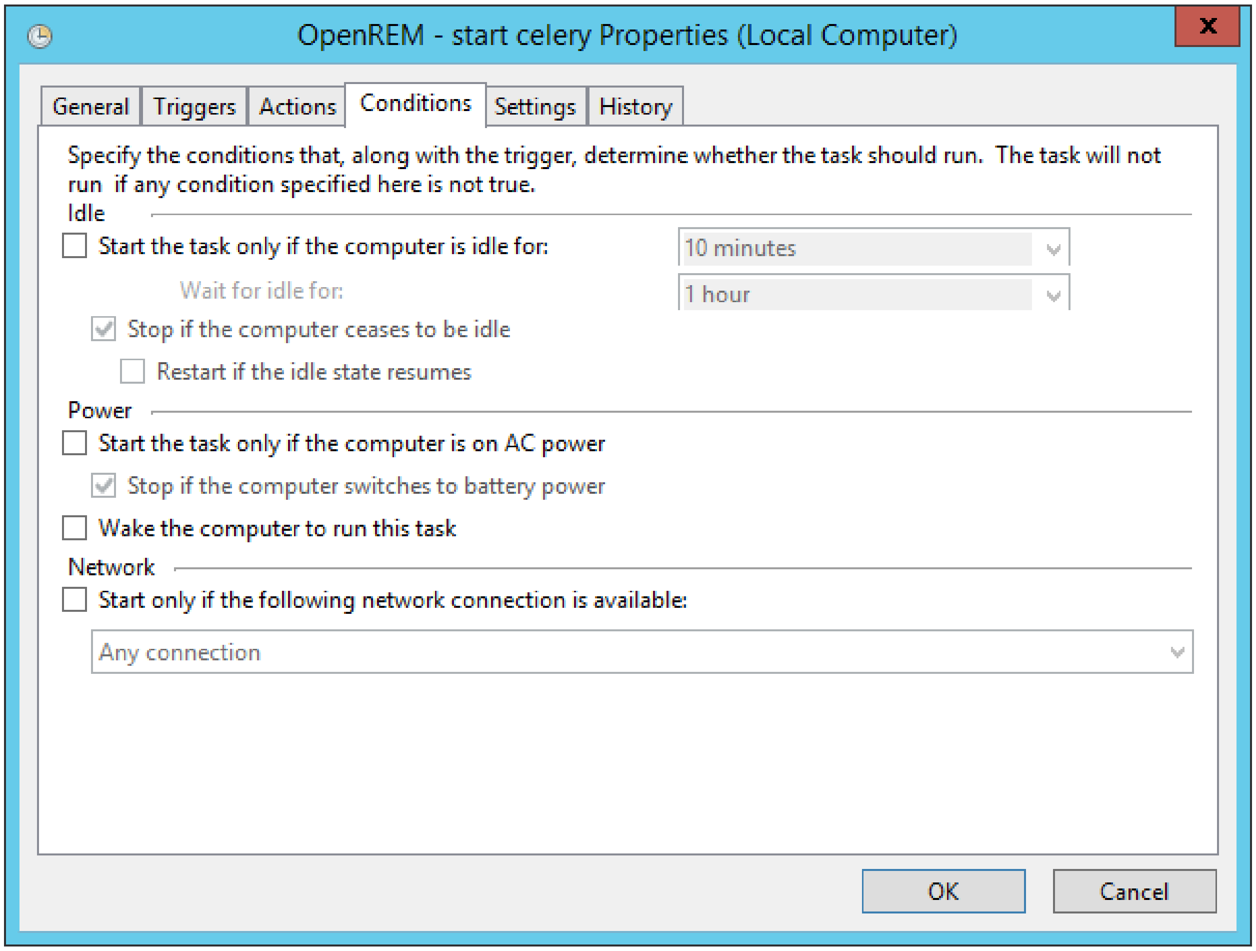
Fig. 4: Figure 4: Action properties

There are no particular conditions set for the task, as shown in figure 5.

Finally, click on the `Settings` tab (figure 6). Check the `Allow task to be run on demand` box, and also the `If the running task does not end when requested, force it to stop` box. Choose `Stop the existing instance` from the `If the task is already running, then the following rule applies:` list. Then click the `OK` button to add the task to the scheduler library.

For Flower

Repeat the above steps for the Flower batch file, but only configure the Flower task to trigger on system start-up: there should be no need to schedule re-starts of Flower.


OpenREM - start celery Properties (Local Computer)

Specify the conditions that, along with the trigger, determine whether the task should run. The task will not run if any condition specified here is not true.

Idle

- [ ] Start the task only if the computer is idle for: 10 minutes
- Wait for idle for: 1 hour
- [✓] Stop if the computer ceases to be idle
- [ ] Restart if the idle state resumes

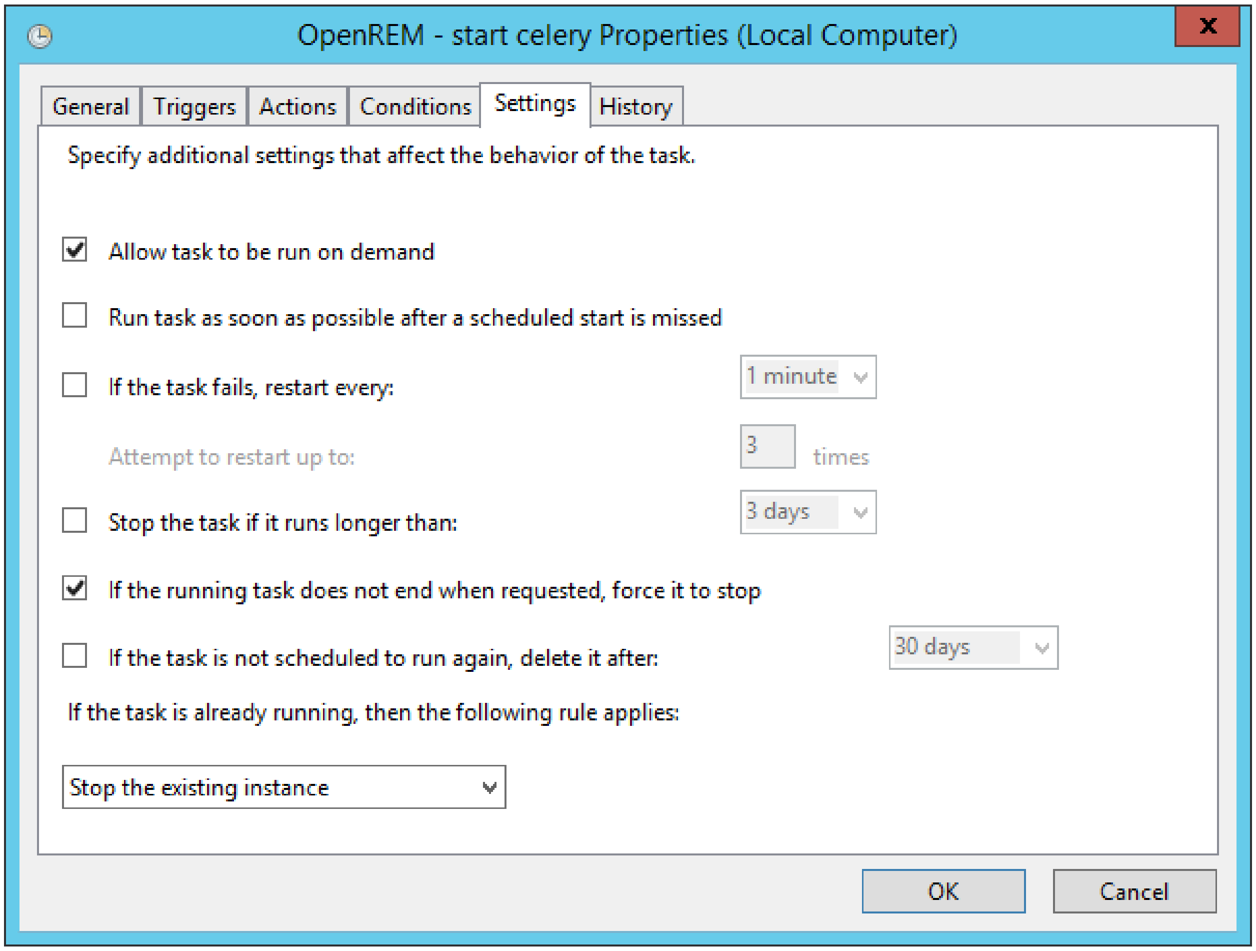
Power

- [ ] Start the task only if the computer is on AC power
- [✓] Stop if the computer switches to battery power
- [ ] Wake the computer to run this task

Network

- [ ] Start only if the following network connection is available: Any connection

Fig. 5: Figure 5: Condition properties



Specify additional settings that affect the behavior of the task.

- [✓] Allow task to be run on demand
- [ ] Run task as soon as possible after a scheduled start is missed
- [ ] If the task fails, restart every: 1 minute
- Attempt to restart up to: 3 times
- [ ] Stop the task if it runs longer than: 3 days
- [✓] If the running task does not end when requested, force it to stop
- [ ] If the task is not scheduled to run again, delete it after: 30 days

If the task is already running, then the following rule applies:

Stop the existing instance

Fig. 6: Figure 6: Task settings

CHAPTER 3

Configuration and administration

See also: `local_settings.py` *Configuration*

3.1 Home page options

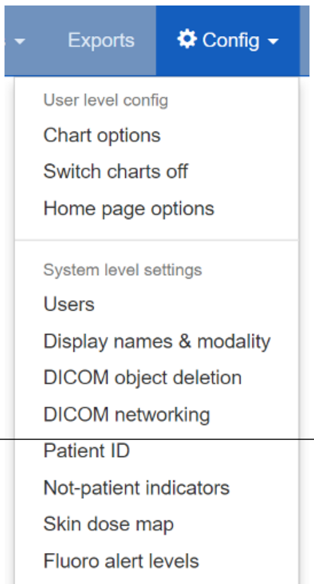
New in 0.8.2

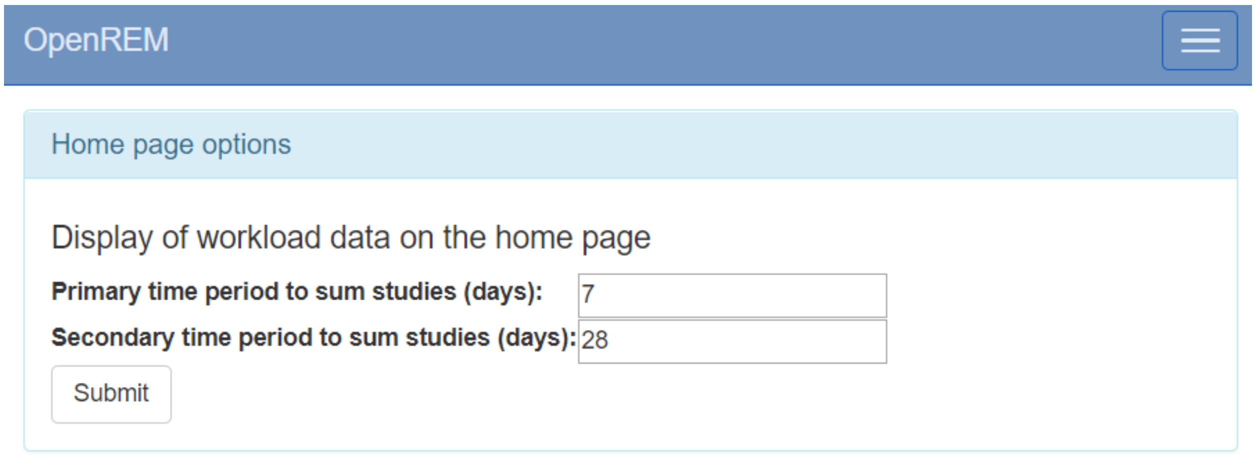
Contents

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).





OpenREM

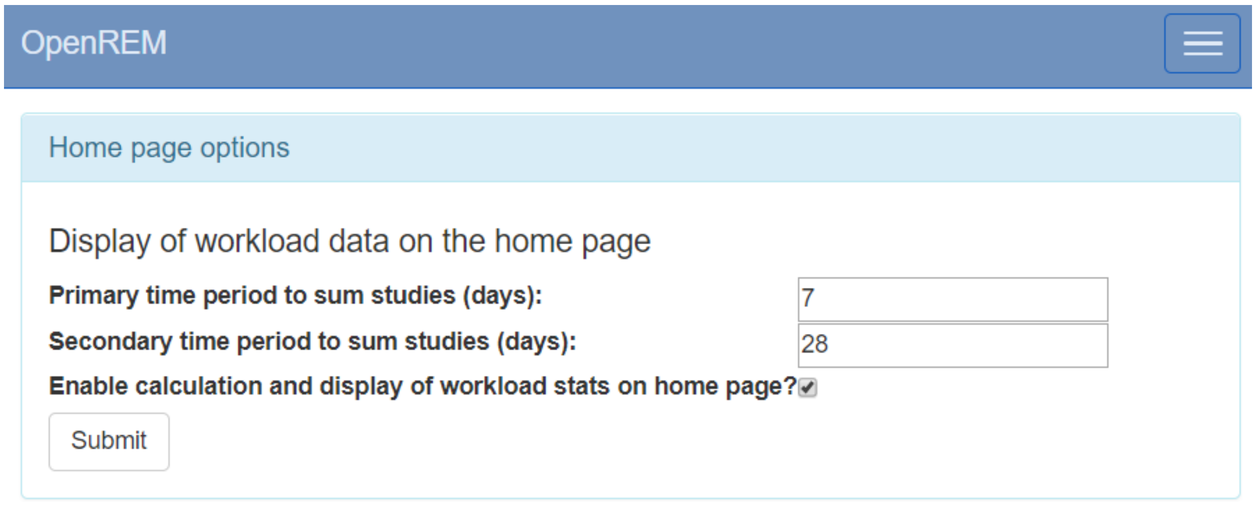
Home page options

Display of workload data on the home page

Primary time period to sum studies (days):	7
Secondary time period to sum studies (days):	28

Submit

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



OpenREM

Home page options

Display of workload data on the home page

Primary time period to sum studies (days):	7
Secondary time period to sum studies (days):	28
Enable calculation and display of workload stats on home page?	✓

Submit

Fig. 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*)

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

3.2.1 Configure what is deleted

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

This will open the configuration page:

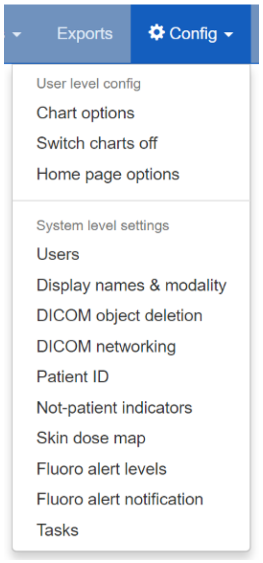
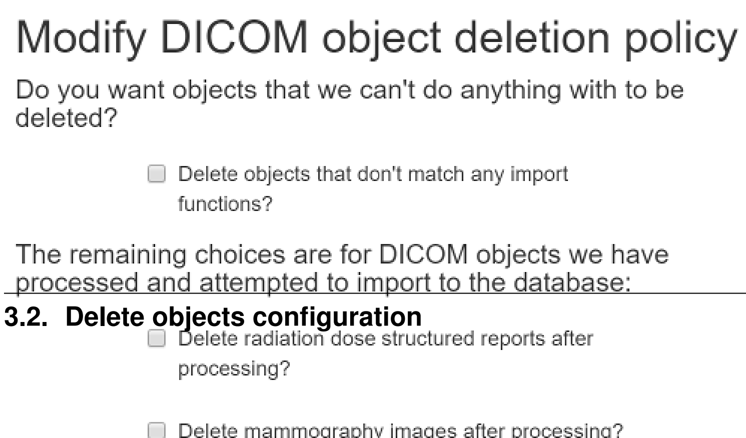


Fig. 4: The `Config` menu

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?

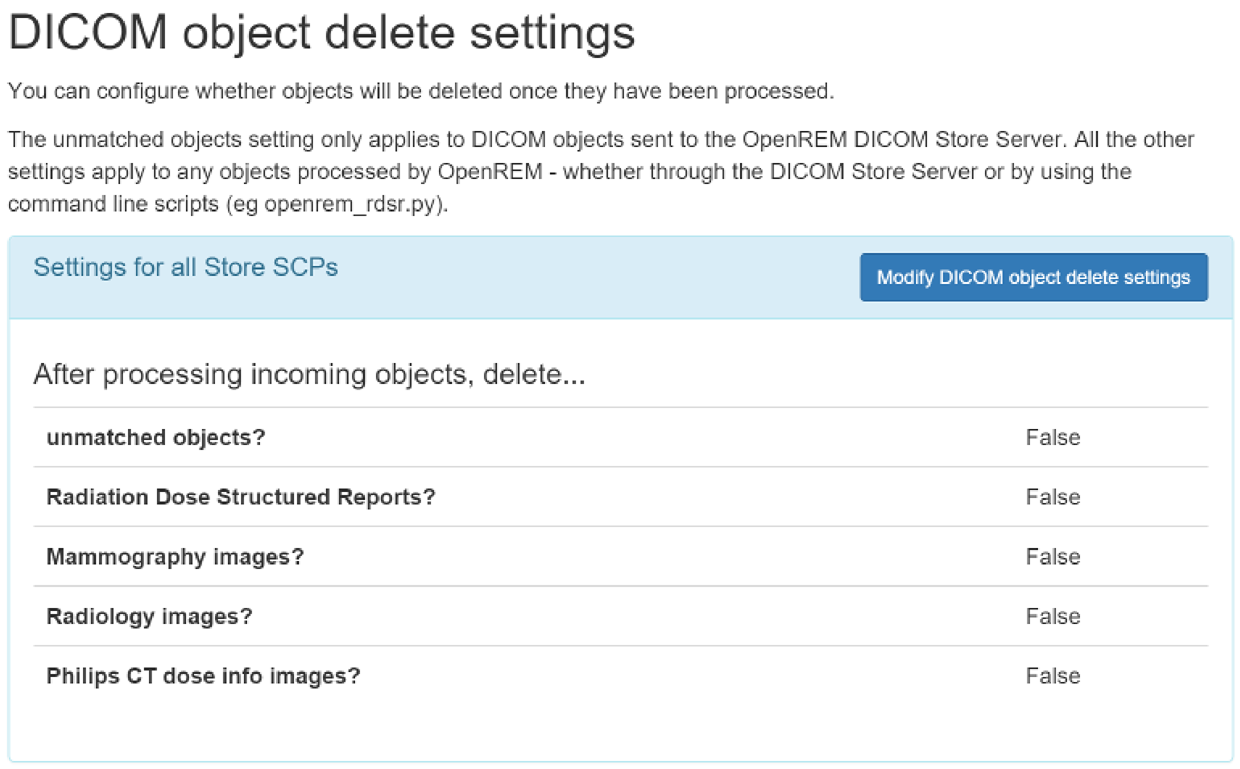
The remaining choices are for DICOM objects we have processed and attempted to import to the database:

☐ Delete radiation dose structured reports after processing?

☐ Delete mammography 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:


DICOM object delete settings

You can configure whether objects will be deleted once they have been processed.

The unmatched objects setting only applies to DICOM objects sent to the OpenREM DICOM Store Server. All the other settings apply to any objects processed by OpenREM - whether through the DICOM Store Server or by using the command line scripts (eg openrem_rdsr.py).

Settings for all Store SCPs

After processing incoming objects, delete...

unmatched objects?	False
Radiation Dose Structured Reports?	False
Mammography images?	False
Radiology images?	False
Philips CT dose info images?	False

Fig. 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

Functionality changed in 0.8.0

Contents

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*

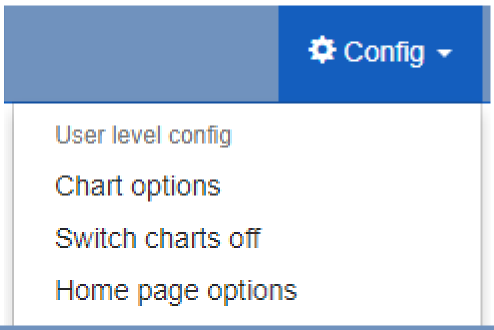
3.3.2 User defined modality field

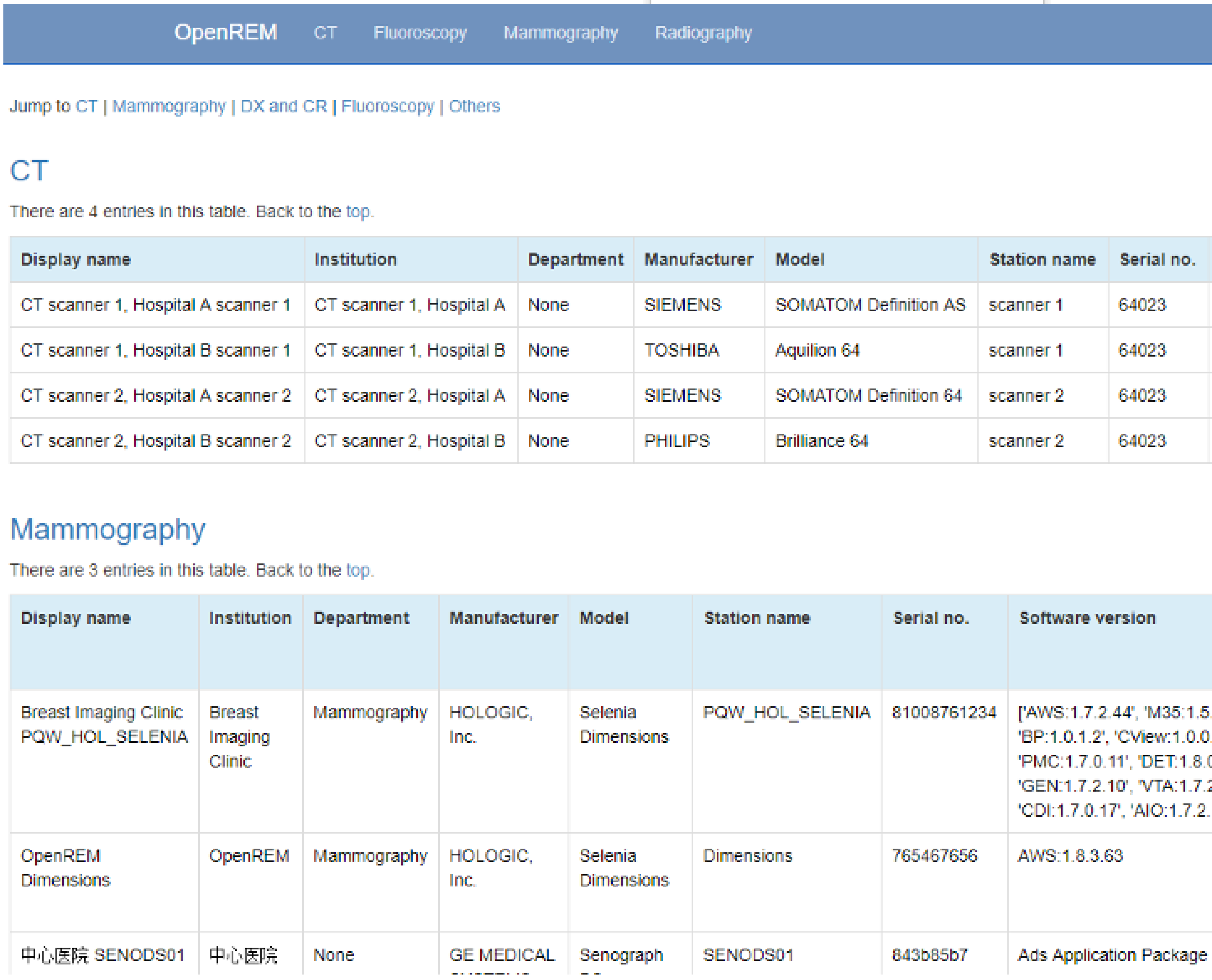
OpenREM determines the modality type of a system based on the information in the DICOM radiation dose structured report. However sometimes this mechanism fails because vendors use templates meant for RF also for DX systems. Therefore it is possible from version 0.8.0 to set a modality type for each system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





Jump to CT | Mammography | DX and CR | Fluoroscopy | Others

CT

There are 4 entries in this table. Back to the top.

Display name	Institution	Department	Manufacturer	Model	Station name	Serial no.
CT scanner 1, Hospital A scanner 1	CT scanner 1, Hospital A	None	SIEMENS	SOMATOM Definition AS	scanner 1	64023
CT scanner 1, Hospital B scanner 1	CT scanner 1, Hospital B	None	TOSHIBA	Aquilion 64	scanner 1	64023
CT scanner 2, Hospital A scanner 2	CT scanner 2, Hospital A	None	SIEMENS	SOMATOM Definition 64	scanner 2	64023
CT scanner 2, Hospital B scanner 2	CT scanner 2, Hospital B	None	PHILIPS	Brilliance 64	scanner 2	64023

Mammography

There are 3 entries in this table. Back to the top.

Display name	Institution	Department	Manufacturer	Model	Station name	Serial no.	Software version
Breast Imaging Clinic PQW_HOL_SELENIA	Breast Imaging Clinic	Mammography	HOLOGIC, Inc.	Selenia Dimensions	PQW_HOL_SELENIA	81008761234	['AWS:1.7.2.44', 'M35:1.5 'BP:1.0.1.2', 'CView:1.0.0 'PMC:1.7.0.11', 'DET:1.8.0 'GEN:1.7.2.10', 'VTA:1.7.2 'CDI:1.7.0.17', 'AIO:1.7.2.
OpenREM Dimensions	OpenREM	Mammography	HOLOGIC, Inc.	Selenia Dimensions	Dimensions	765467656	AWS:1.8.3.63
中心医院 SENODS01	中心医院	None	GE MEDICAL	Senograph	SENODS01	843b85b7	Ads Application Package

Fig. 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:

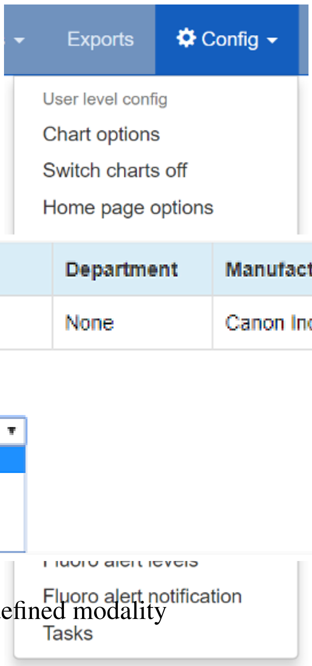
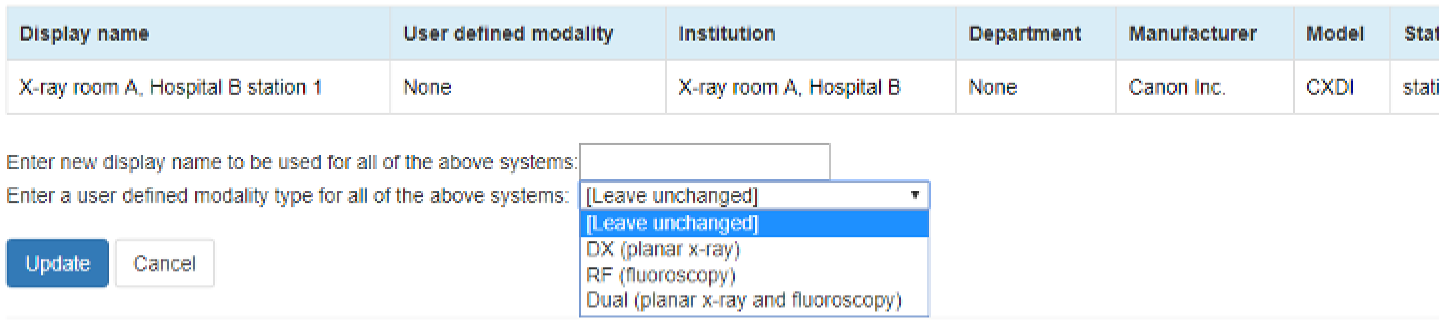


Fig. 9: The `Config` menu (admin)

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:



Display name	User defined modality	Institution	Department	Manufacturer	Model	Stat
X-ray room A, Hospital B station 1	None	X-ray room A, Hospital B	None	Canon Inc.	CXDI	stati

Enter new display name to be used for all of the above systems:

Enter a user defined modality type for all of the above systems: [Leave unchanged]

Fig. 10: Example of the page for updating a display name and user defined modality

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.

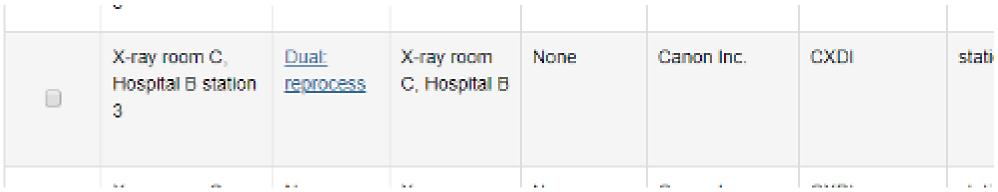
If you update the user defined modality, the modality type for already imported studies will also be set to the user defined modality type. Only changes from modality DX (planar X-ray) to RF (fluoroscopy) and vice versa are possible.

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:



☐	X-ray room C, Hospital B station 3	Dual: reprocess	X-ray room C, Hospital B	None	Canon Inc.	CXDI

Fig. 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.

3.4 Not-patient indicator settings

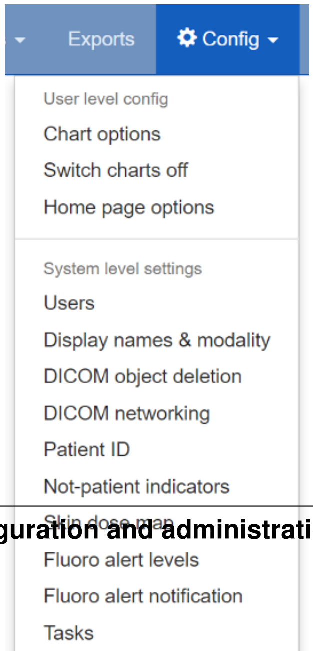
The standard configuration for OpenREM is to not store any patient identifiable information. Therefore it can be difficult to distinguish between real patients and test or quality assurance exposures.

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 hard-coded. 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

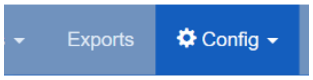
In addition, a key identifier for the exam that is normally not considered patient identifiable was stored:

- Study accession number

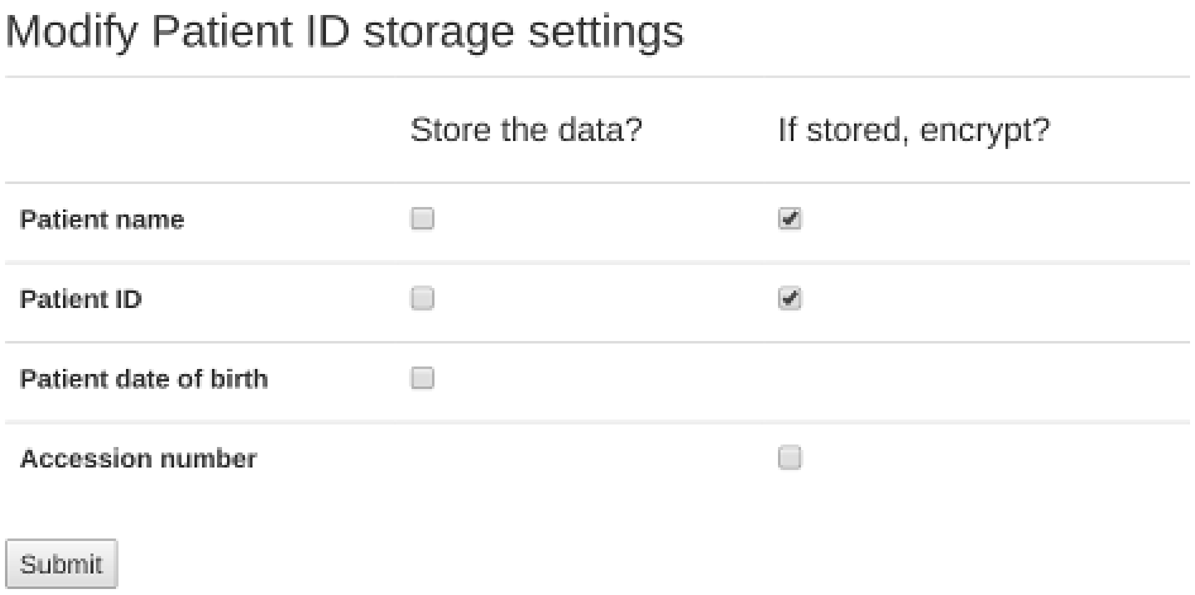
It has become apparent that there are reasons where people need to store patient identifiable data to make the most of OpenREM, so this is now configurable from version 0.7 onwards.

3.5.1 Configure what is stored

On the Config menu, select `Patient ID`:



The initial settings are as follows:



Modify Patient ID storage settings

	Store the data?	If stored, encrypt?
Patient name	☐	☑
Patient ID	☐	☑
Patient date of birth	☐	
Accession number		☐

Submit

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.

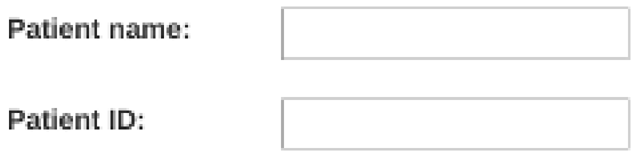
If *exactly* the same name or ID (including spelling, spacing, case etc) occurs more than once, then the same hash will be generated.

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:

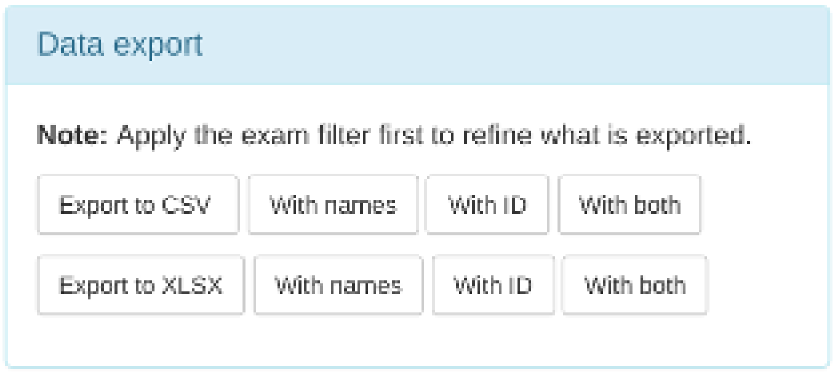


If the values in the database are *not* encrypted, then partial search terms can be used as a case-insensitive 'contains' query will be applied.

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:



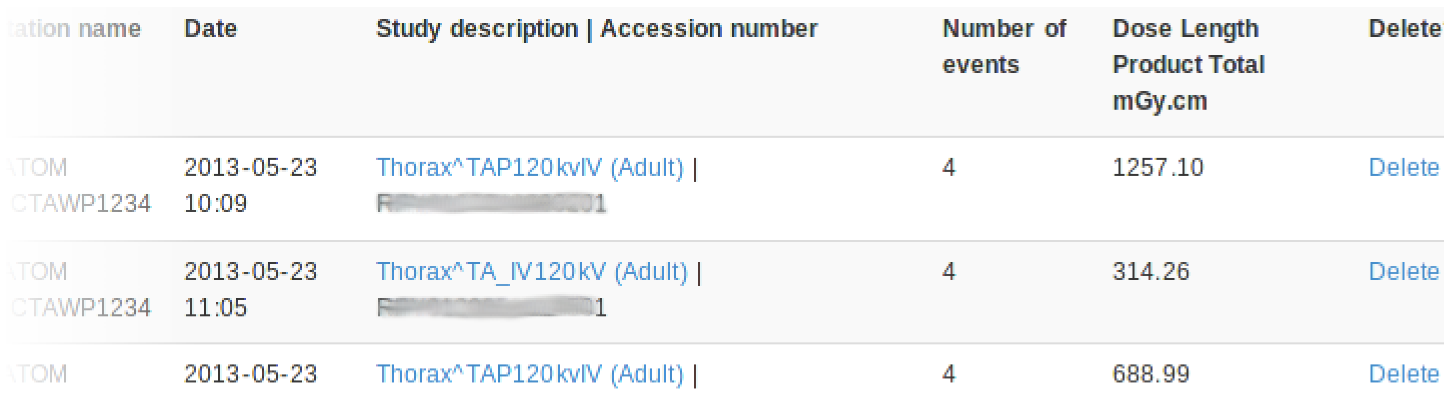
If the IDs or names are encrypted, then these columns will contain the hash rather than the original values. However, it will be possible to see if more than one study belongs to one patient as the values should be the same for both. Due to the nature of the algorithm however, a single change in the 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^TAP120kvIV (Adult) | [redacted]	4	1257.10	Delete
TOM CTAWP1234	2013-05-23 11:05	Thorax^TA_IV120kV (Adult) | [redacted]	4	314.26	Delete
TOM	2013-05-23	Thorax^TAP120kvIV (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.

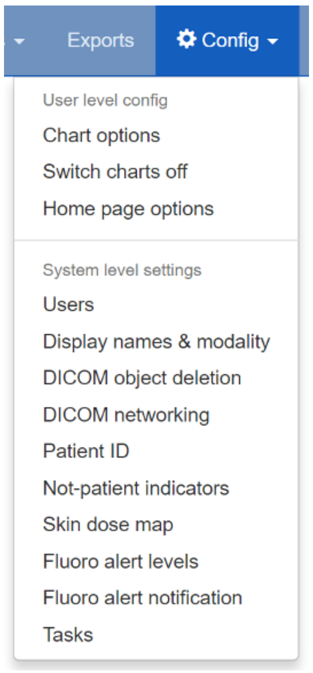
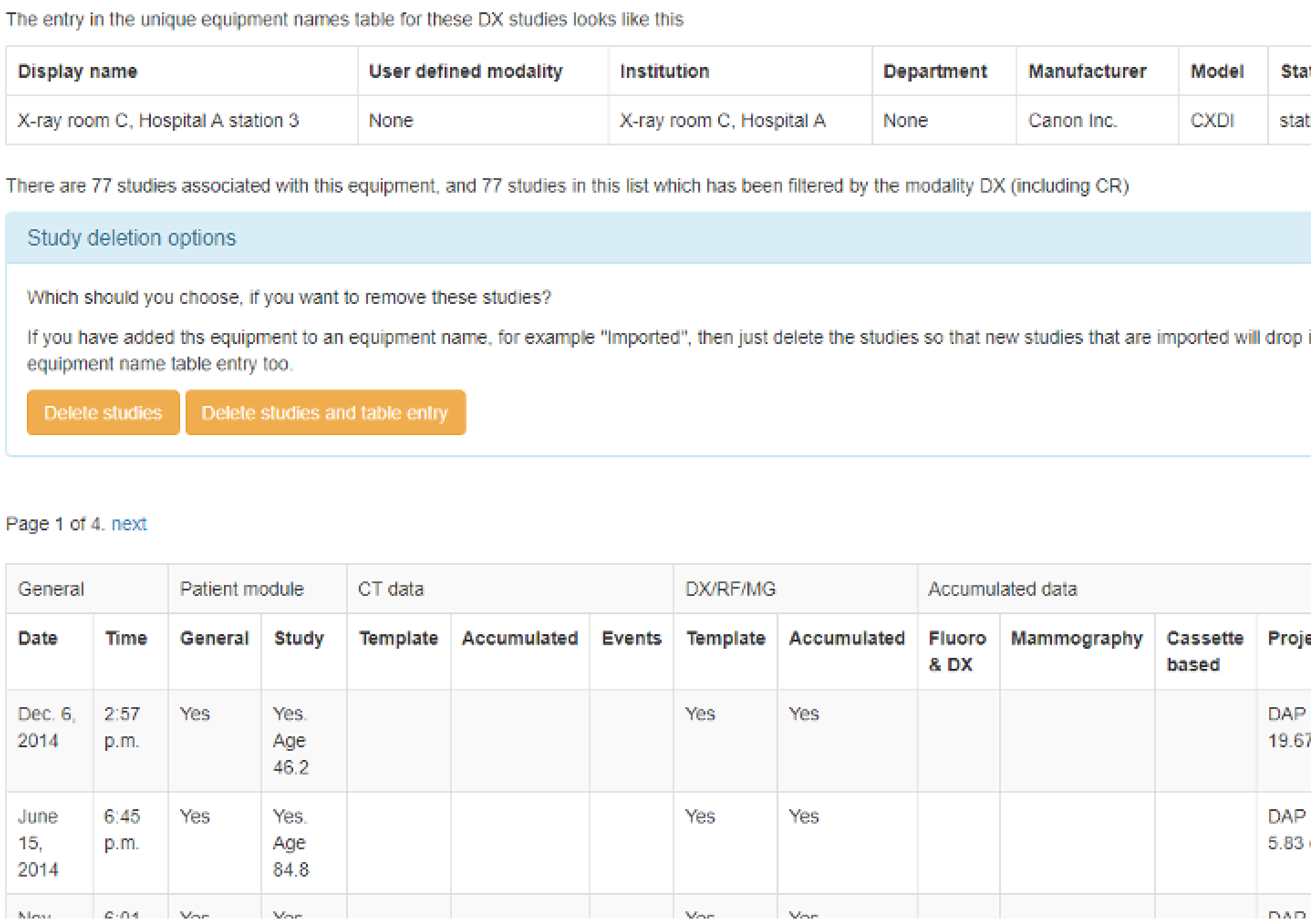


Fig. 13: The `Config` menu (admin)

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.



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

Display name	User defined modality	Institution	Department	Manufacturer	Model	Sta
X-ray room C, Hospital A station 3	None	X-ray room C, Hospital A	None	Canon Inc.	CXDI	stat

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 i 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			
Date	Time	General	Study	Template	Accumulated	Events	Template	Accumulated	Fluoro & DX	Mammography	Cassette based	Proje
Dec. 6, 2014	2:57 p.m.	Yes	Yes. Age 46.2				Yes	Yes				DAP 19.67
June 15, 2014	6:45 p.m.	Yes	Yes. Age 84.8				Yes	Yes				DAP 5.83
Nov.	6:01	Yes	Yes				Yes	Yes				DAP

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

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.

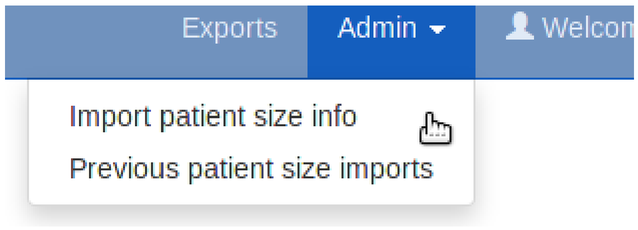
Before release 0.8.2, these studies would appear in the homepage listing as *Error has occurred - import probably unsuccessful*. This has now changed to a link to the review page for that modality with the text *Failed import - review here* for users in the `admingroup` and *Failed import - ask an administrator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

3.7 Adding patient size information from csv using the web interface

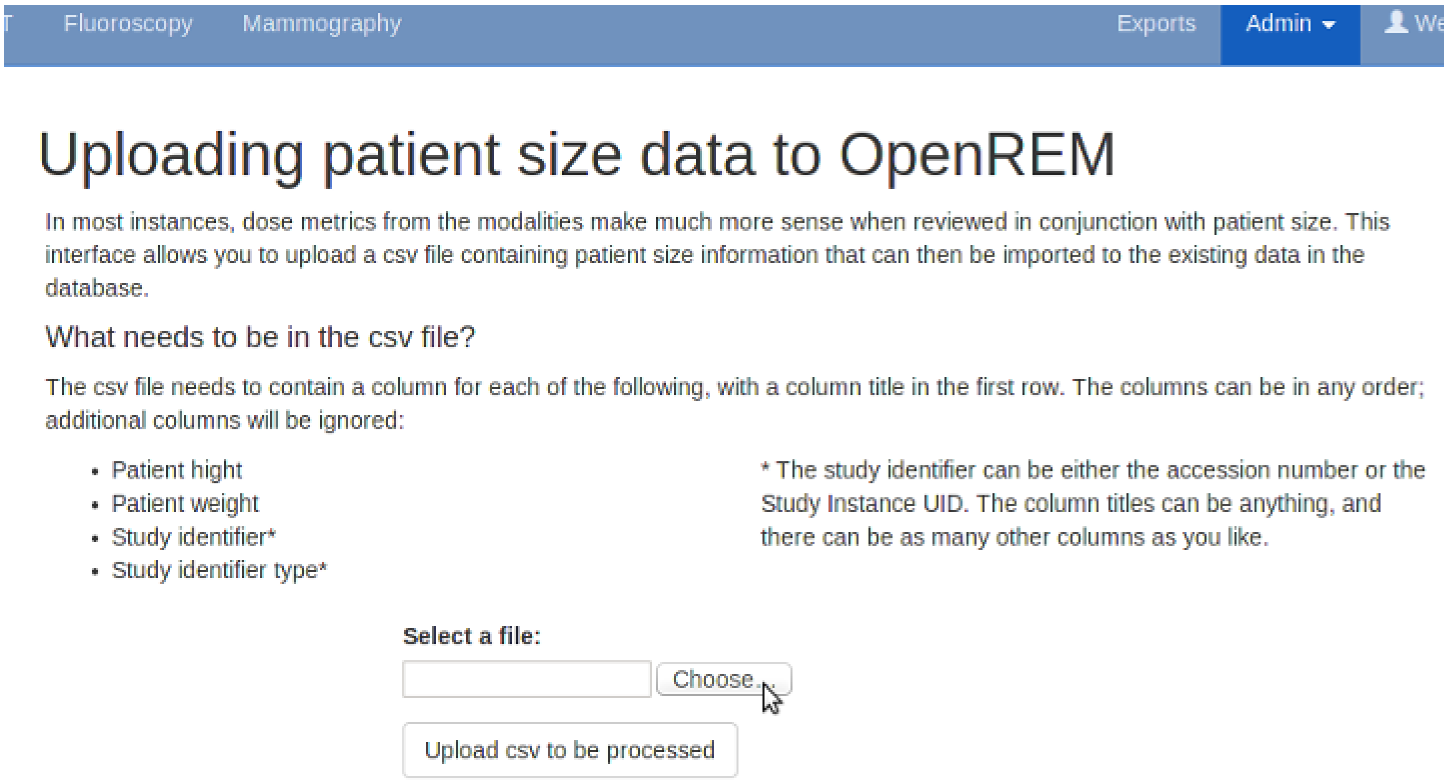
Contents

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:

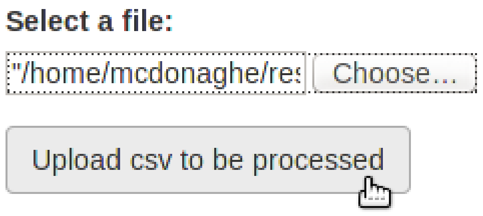
- Patient hight
- Patient weight
- Study identifier*
- Study identifier type*

* 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.

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.

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.



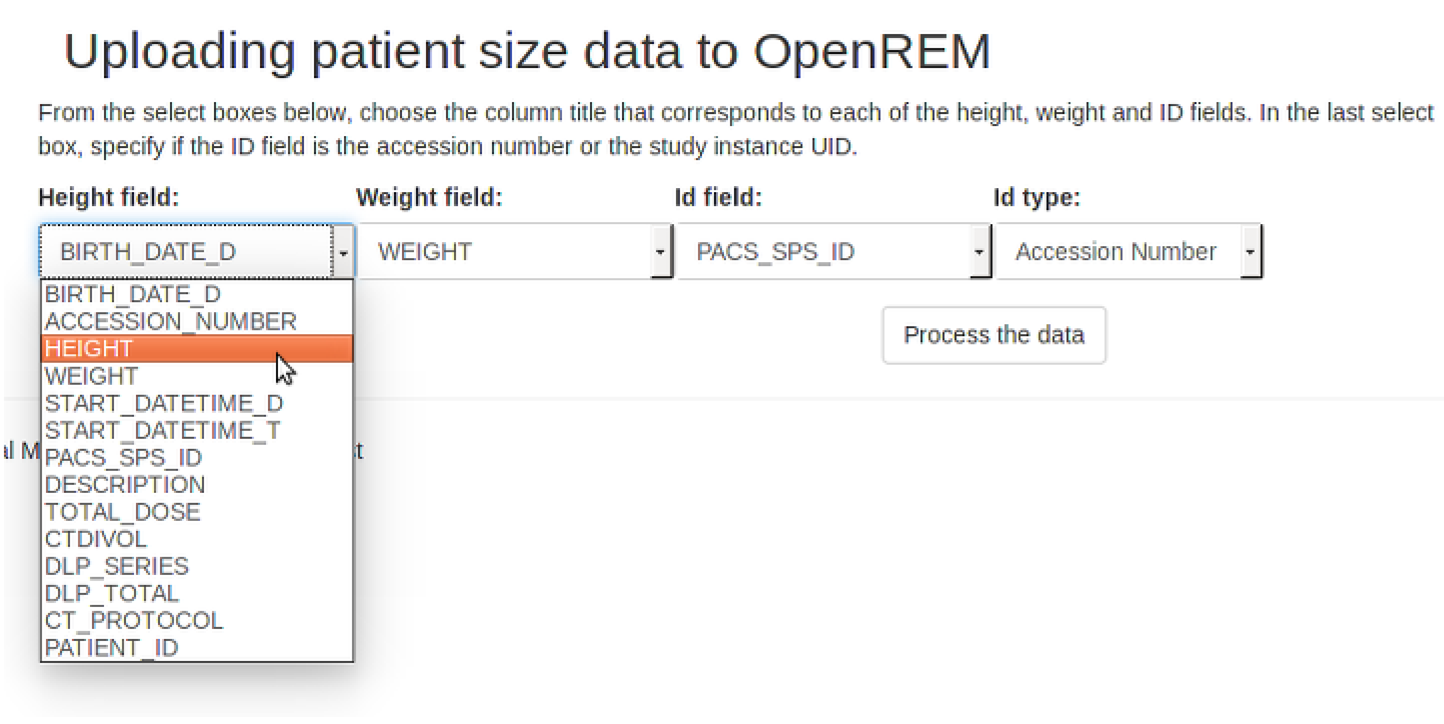
The csv file needs to have at least the required columns. Additional columns will be ignored. If your source of patient size data does not have either the height or the weight column, simply add a new empty column with just the title in the first row.

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.

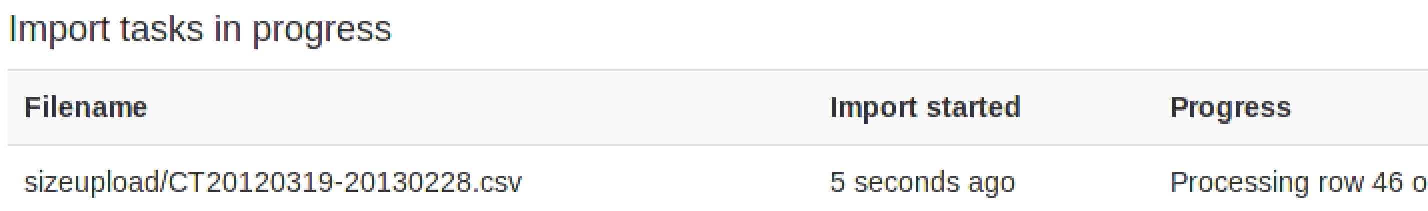


Uploading patient size data to OpenREM

From the select boxes below, choose the column title that corresponds to each of the height, weight and ID fields. In the last select box, specify if the ID field is the accession number or the study instance UID.

Height field:	Weight field:	Id field:	Id type:
BIRTH_DATE_D	WEIGHT	PACS_SPS_ID	Accession Number

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



Import tasks in progress

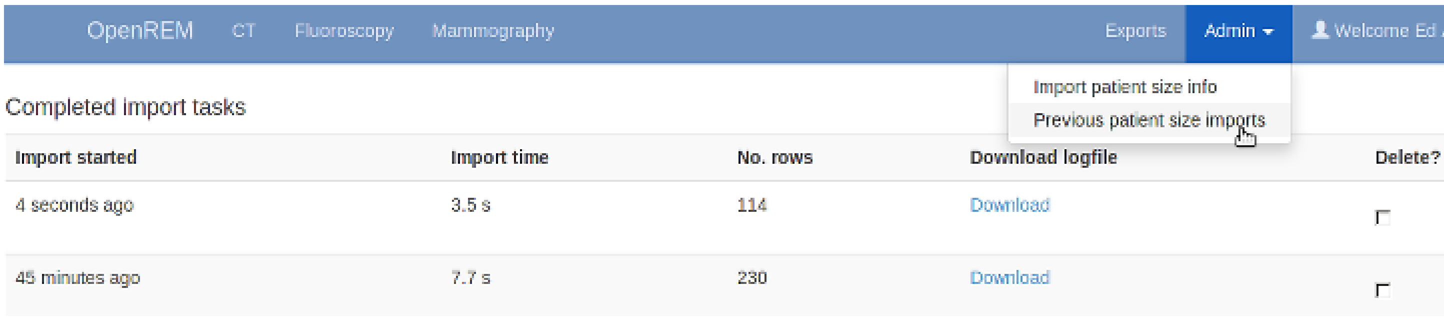
Filename	Import started	Progress
sizeupload/CT20120319-20130228.csv	5 seconds ago	Processing row 46 o

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

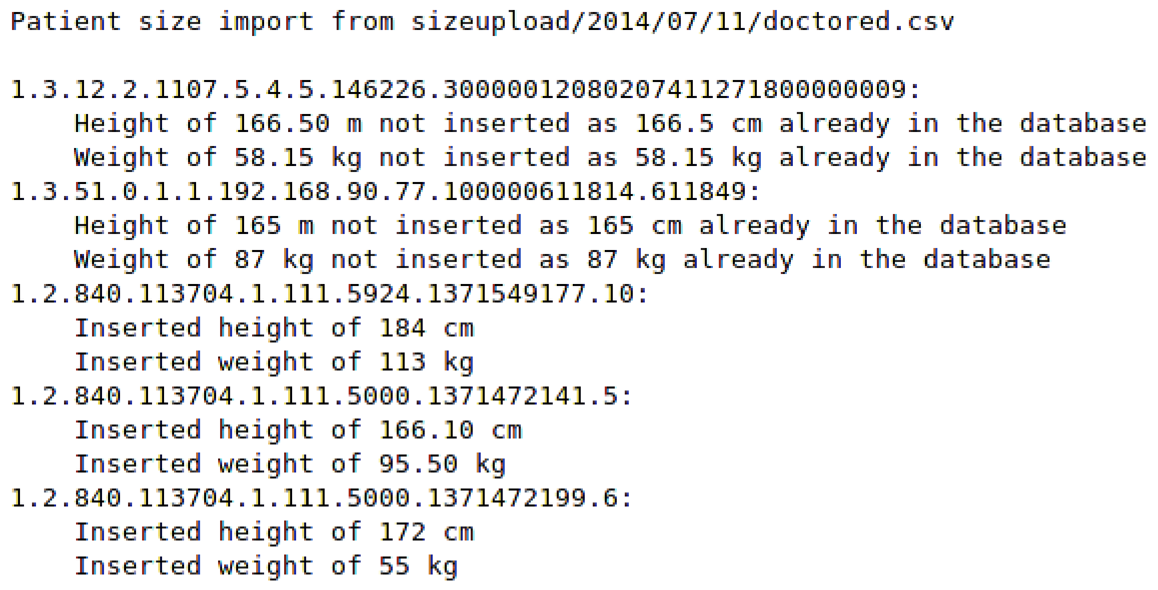
After an import is complete, it is listed in the completed import tasks table. You can also get to this page from the Admin menu:



Completed import tasks

Import started	Import time	No. rows	Download logfile	Delete?
4 seconds ago	3.5 s	114	Download	☐
45 minutes ago	7.7 s	230	Download	☐

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` *New in 0.3.7* 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.

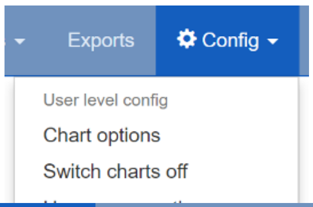
3.9 Fluroscopy high dose alerts

New in 0.8.2

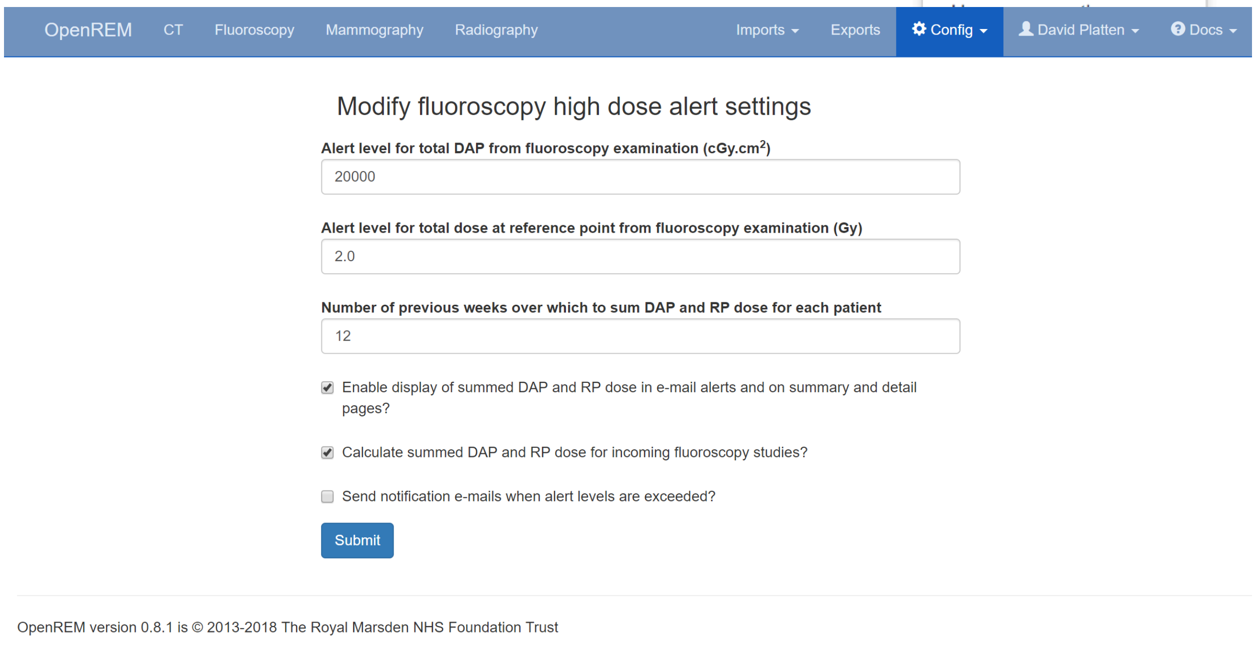
Contents

3.9.1 Alert level configuration

The system highlights fluoroscopy studies that have exceeded defined levels of DAP and total dose at reference point. 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 and 2 Gy total dose at reference point (figure 2).



Modify fluoroscopy high dose alert settings

Alert level for total DAP from fluoroscopy examination (cGy.cm^2)
20000

Alert level for total dose at reference point from fluoroscopy examination (Gy)
2.0

Number of previous weeks over which to sum DAP and RP dose for each patient
12

[✓] Enable display of summed DAP and RP dose in e-mail alerts and on summary and detail pages?

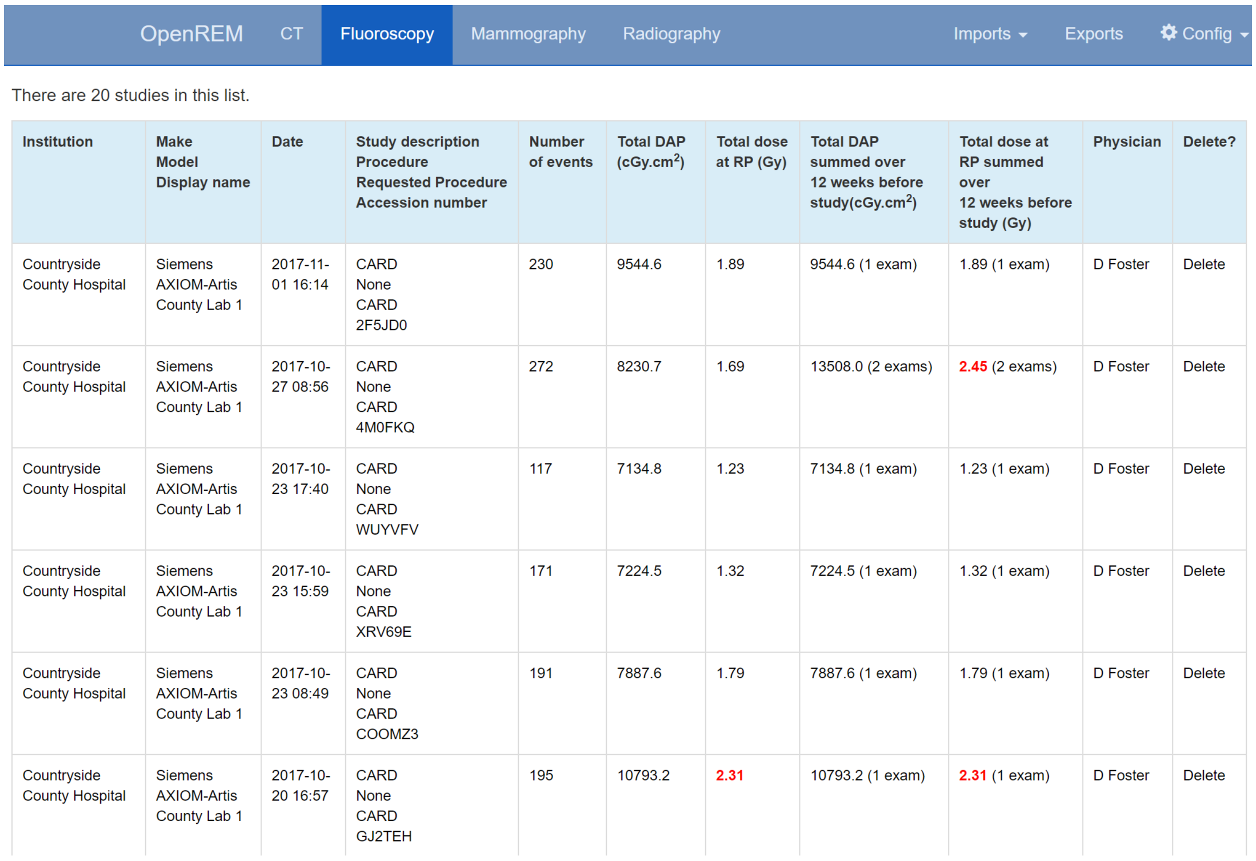
[✓] Calculate summed DAP and RP dose for incoming fluoroscopy studies?

[ ] Send notification e-mails when alert levels are exceeded?

OpenREM version 0.8.1 is © 2013-2018 The Royal Marsden NHS Foundation Trust

Fig. 16: Figure 2: Fluoroscopy high dose alert settings

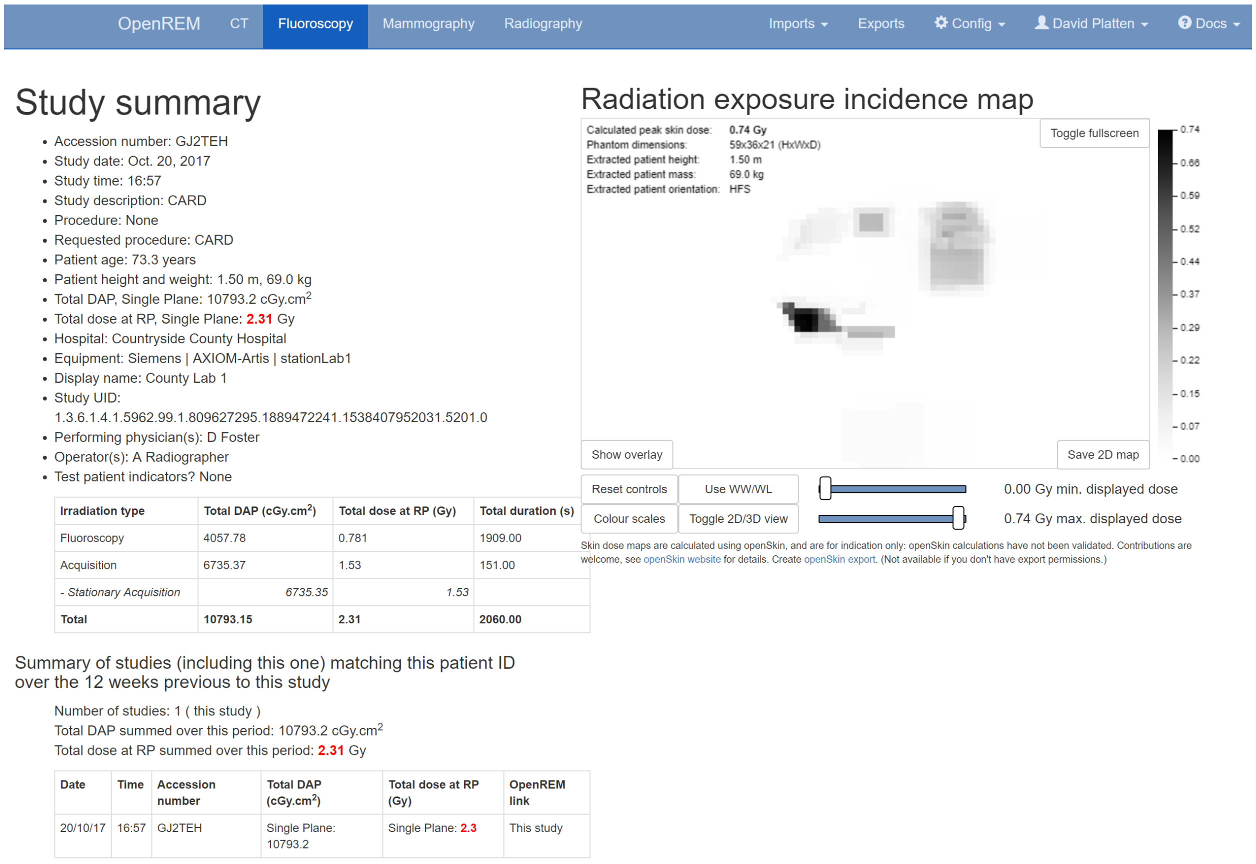
Figures 3 and 4 illustrate how studies that exceed an alert level are highlighted in the filtered and detailed fluoroscopy views.



There are 20 studies in this list.

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

Fig. 17: Figure 3: Filtered view showing the highlighting of some high dose studies



Study summary

- Accession number: GJ2TEH
- Study date: Oct. 20, 2017
- Study time: 16:57
- Study description: CARD
- Procedure: None
- Requested procedure: CARD
- Patient age: 73.3 years
- Patient height and weight: 1.50 m, 69.0 kg
- Total DAP, Single Plane: 10793.2 cGy.cm^2
- Total dose at RP, Single Plane: **2.31** Gy
- Hospital: Countryside County Hospital
- Equipment: Siemens | AXIOM-Artis | stationLab1
- Display name: County Lab 1
- Study UID: 1.3.6.1.4.1.5962.99.1.809627295.1889472241.1538407952031.5201.0
- Performing physician(s): D Foster
- Operator(s): A Radiographer
- Test patient indicators? None

Irradiation type	Total DAP (cGy.cm^2)	Total dose at RP (Gy)	Total duration (s)
Fluoroscopy	4057.78	0.781	1909.00
Acquisition	6735.37	1.53	151.00
- Stationary Acquisition	*6735.35*	*1.53*	
Total	**10793.15**	**2.31**	**2060.00**

Summary of studies (including this one) matching this patient ID over the 12 weeks previous to this study

Number of studies: 1 (this study)
Total DAP summed over this period: 10793.2 cGy.cm^2
Total dose at RP summed over this period: **2.31** Gy

Date	Time	Accession number	Total DAP (cGy.cm^2)	Total dose at RP (Gy)	OpenREM link
20/10/17	16:57	GJ2TEH	Single Plane: 10793.2	Single Plane: **2.3**	This study

Radiation exposure incidence map

Calculated peak skin dose:	**0.74 Gy**
Phantom dimensions:	59x36x21 (HxWxD)
Extracted patient height:	1.50 m
Extracted patient mass:	69.0 kg
Extracted patient orientation:	HFS

0.00 Gy min. displayed dose
0.74 Gy max. displayed dose

Skin dose maps are calculated using openSkin, and are for indication only: openSkin calculations have not been validated. Contributions are welcome, see openSkin website for details. Create openSkin export. (Not available if you don't have export permissions.)

Fig. 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 OpenREM 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 OpenREM administrator, and defaults to 12 weeks (figure 2).

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).

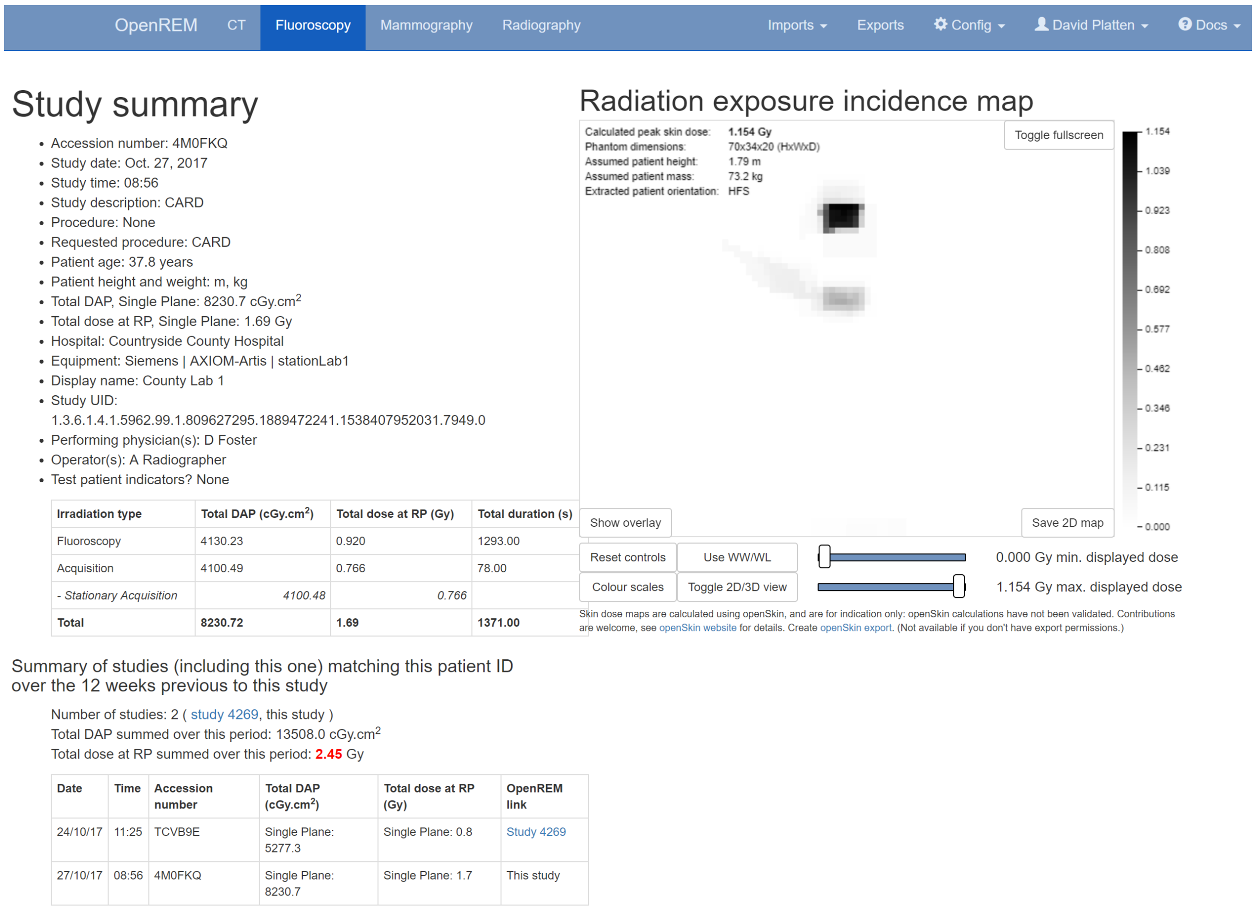
3.9.4 E-mail notifications of high dose alerts

For this feature to function the e-mail section in `local_settings.py` must be correctly completed (see the *E-mail 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.

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.



Study summary

- Accession number: 4M0FKQ
- Study date: Oct. 27, 2017
- Study time: 08:56
- Study description: CARD
- Procedure: None
- Requested procedure: CARD
- Patient age: 37.8 years
- Patient height and weight: m, kg
- Total DAP, Single Plane: 8230.7 cGy.cm^2
- Total dose at RP, Single Plane: 1.69 Gy
- Hospital: Countryside County Hospital
- Equipment: Siemens | AXIOM-Artis | stationLab1
- Display name: County Lab 1
- Study UID: 1.3.6.1.4.1.5962.99.1.809627295.1889472241.1538407952031.7949.0
- Performing physician(s): D Foster
- Operator(s): A Radiographer
- Test patient indicators? None

Irradiation type	Total DAP (cGy.cm^2)	Total dose at RP (Gy)	Total duration (s)
Fluoroscopy	4130.23	0.920	1293.00
Acquisition	4100.49	0.766	78.00
- Stationary Acquisition	*4100.48*	*0.766*	
Total	**8230.72**	**1.69**	**1371.00**

Summary of studies (including this one) matching this patient ID over the 12 weeks previous to this study

Number of studies: 2 (study 4269, this study)
Total DAP summed over this period: 13508.0 cGy.cm^2
Total dose at RP summed over this period: **2.45** Gy

Date	Time	Accession number	Total DAP (cGy.cm^2)	Total dose at RP (Gy)	OpenREM link
24/10/17	11:25	TCVB9E	Single Plane: 5277.3	Single Plane: 0.8	Study 4269
27/10/17	08:56	4M0FKQ	Single Plane: 8230.7	Single Plane: 1.7	This study

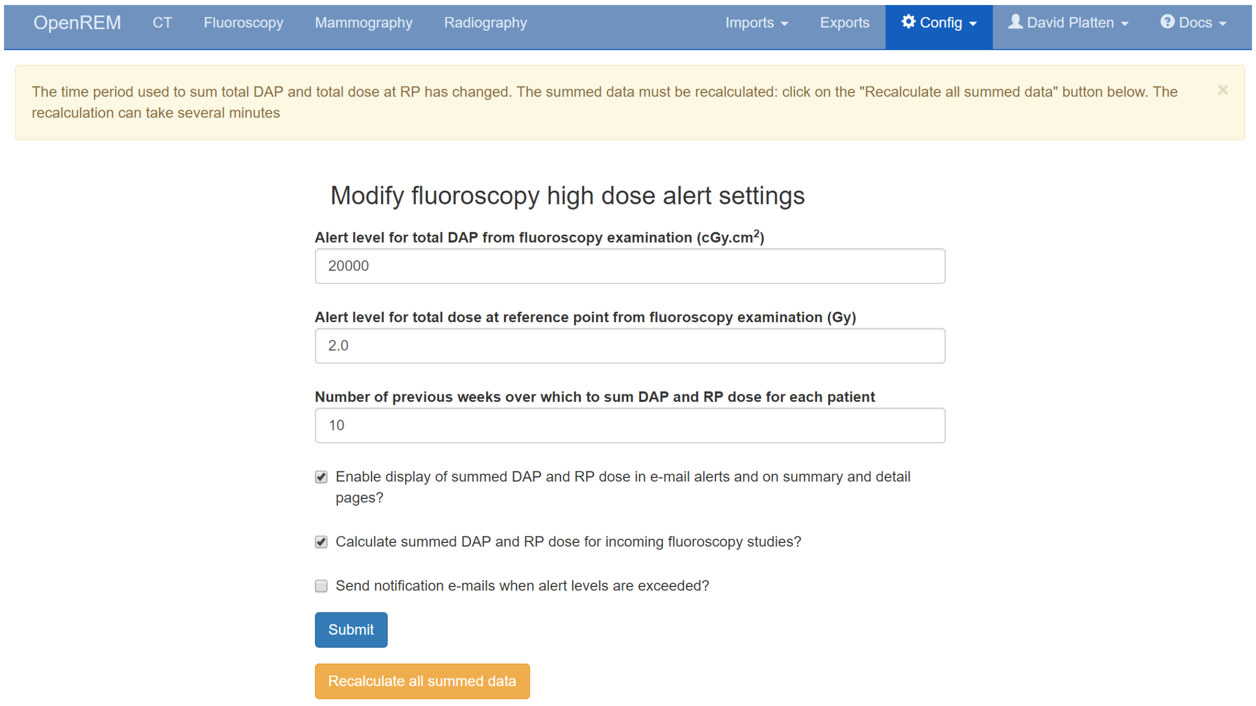
Radiation exposure incidence map

Calculated peak skin dose:	**1.154 Gy**
Phantom dimensions:	70x34x20 (HxWxD)
Assumed patient height:	1.79 m
Assumed patient mass:	73.2 kg
Extracted patient orientation:	HFS

0.000 Gy min. displayed dose
1.154 Gy max. displayed dose

Skin dose maps are calculated using openSkin, and are for indication only: openSkin calculations have not been validated. Contributions are welcome, see openSkin website for details. Create openSkin export. (Not available if you don't have export permissions.)

Fig. 19: Figure 5: Detailed view showing associated studies over a time period



The time period used to sum total DAP and total dose at RP has changed. The summed data must be recalculated: click on the "Recalculate all summed data" button below. The recalculation can take several minutes

Modify fluoroscopy high dose alert settings

Alert level for total DAP from fluoroscopy examination (cGy.cm^2)
20000

Alert level for total dose at reference point from fluoroscopy examination (Gy)
2.0

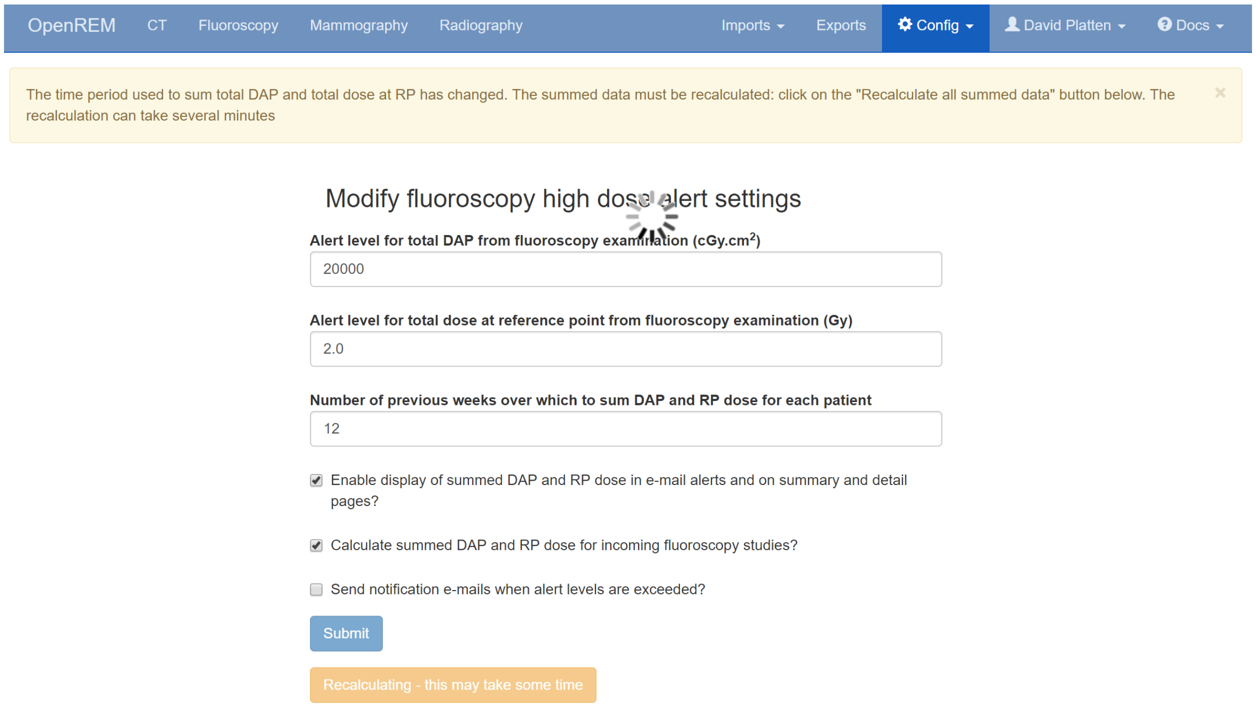
Number of previous weeks over which to sum DAP and RP dose for each patient
10

☑ Enable display of summed DAP and RP dose in e-mail alerts and on summary and detail pages?

☑ Calculate summed DAP and RP dose for incoming fluoroscopy studies?

☐ Send notification e-mails when alert levels are exceeded?

Fig. 20: Figure 6: Prompt to recalculate the summed dose values



The time period used to sum total DAP and total dose at RP has changed. The summed data must be recalculated: click on the "Recalculate all summed data" button below. The recalculation can take several minutes

Modify fluoroscopy high dose alert settings

Alert level for total DAP from fluoroscopy examination (cGy.cm^2)
20000

Alert level for total dose at reference point from fluoroscopy examination (Gy)
2.0

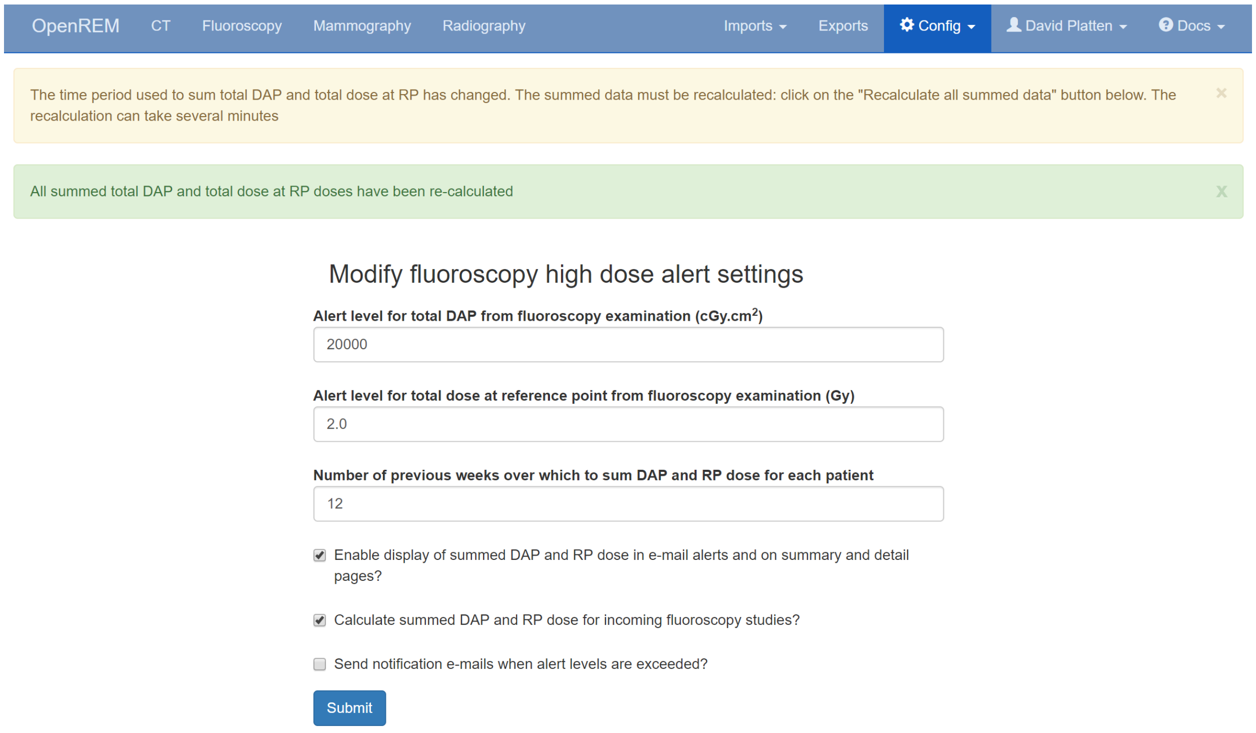
Number of previous weeks over which to sum DAP and RP dose for each patient
12

☑ Enable display of summed DAP and RP dose in e-mail alerts and on summary and detail pages?

☑ Calculate summed DAP and RP dose for incoming fluoroscopy studies?

☐ Send notification e-mails when alert levels are exceeded?

Fig. 21: Figure 7: Prompt to recalculate the summed dose values



The time period used to sum total DAP and total dose at RP has changed. The summed data must be recalculated: click on the "Recalculate all summed data" button below. The recalculation can take several minutes

All summed total DAP and total dose at RP doses have been re-calculated

Modify fluoroscopy high dose alert settings

Alert level for total DAP from fluoroscopy examination (cGy.cm^2)
20000

Alert level for total dose at reference point from fluoroscopy examination (Gy)
2.0

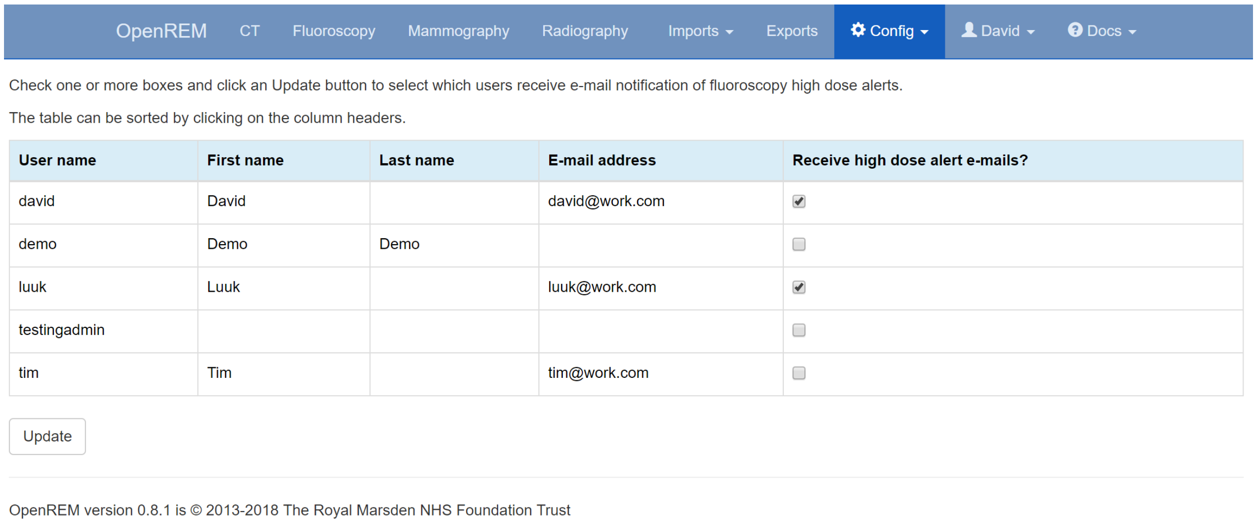
Number of previous weeks over which to sum DAP and RP dose for each patient
12

[✓] Enable display of summed DAP and RP dose in e-mail alerts and on summary and detail pages?

[✓] Calculate summed DAP and RP dose for incoming fluoroscopy studies?

[ ] Send notification e-mails when alert levels are exceeded?

Fig. 22: Figure 8: Message on successful recalculation



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.

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	☐

OpenREM version 0.8.1 is © 2013-2018 The Royal Marsden NHS Foundation Trust

Fig. 23: Figure 9: E-mail user-notification of high-dose alerts

3.10 Task management

New in 0.9

Contents

3.10.1 Enabling RabbitMQ management

Installation instructions were added in 0.9.0. Users upgrading from previous versions should review *Enable RabbitMQ queue management interface*.

3.10.2 Viewing task and service statuses

Users who are logged in with admin rights can use the **Config** menu and choose **Tasks** to see the following:

- The status of the task message broker, RabbitMQ
- The status of the asynchronous task queue, Celery
- The status of the Celery monitoring tool, Flower
- How many tasks are waiting in RabbitMQ for a Celery worker to be available
- A list of the tasks currently being managed by Celery
- A list of previous tasks and their final status

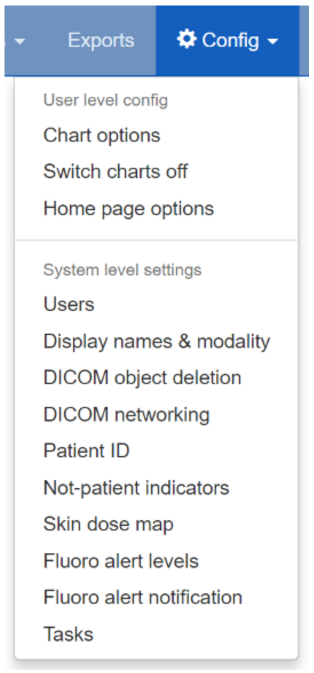
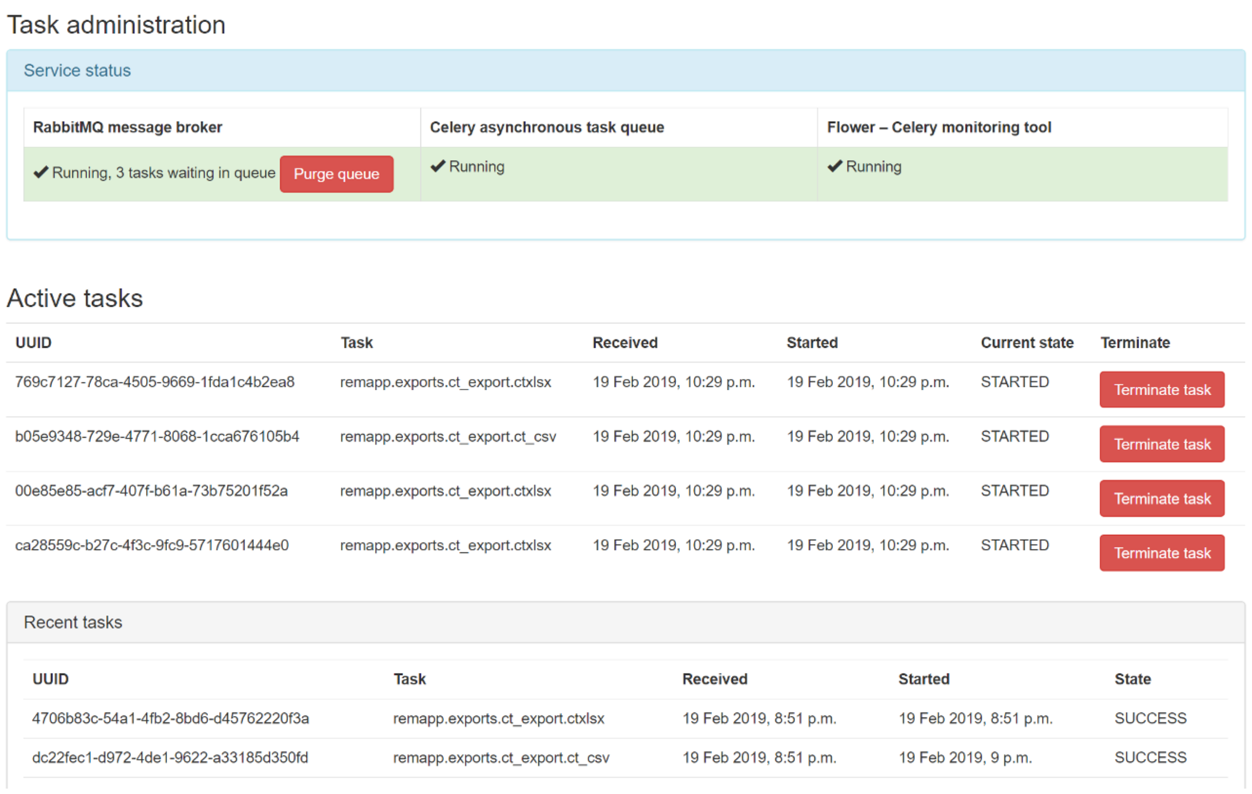


Fig. 24: Figure 1: The `Config` menu (user and admin)

3.10.3 Service statuses

The current status of the services necessary to execute and monitor tasks is displayed in the first section of the page.



Task administration

Service status

RabbitMQ message broker	Celery asynchronous task queue	Flower – Celery monitoring tool
✔ Running, 3 tasks waiting in queue Purge queue	✔ Running	✔ Running

Active tasks

UUID	Task	Received	Started	Current state	Terminate
769c7127-78ca-4505-9669-1fda1c4b2ea8	remapp.exports.ct_export.ctxlsx	19 Feb 2019, 10:29 p.m.	19 Feb 2019, 10:29 p.m.	STARTED	Terminate task
b05e9348-729e-4771-8068-1cca676105b4	remapp.exports.ct_export.ct_csv	19 Feb 2019, 10:29 p.m.	19 Feb 2019, 10:29 p.m.	STARTED	Terminate task
00e85e85-acf7-407f-b61a-73b75201f52a	remapp.exports.ct_export.ctxlsx	19 Feb 2019, 10:29 p.m.	19 Feb 2019, 10:29 p.m.	STARTED	Terminate task
ca28559c-b27c-4f3c-9fc9-5717601444e0	remapp.exports.ct_export.ctxlsx	19 Feb 2019, 10:29 p.m.	19 Feb 2019, 10:29 p.m.	STARTED	Terminate task

Recent tasks

UUID	Task	Received	Started	State
4706b83c-54a1-4fb2-8bd6-d45762220f3a	remapp.exports.ct_export.ctxlsx	19 Feb 2019, 8:51 p.m.	19 Feb 2019, 8:51 p.m.	SUCCESS
dc22fec1-d972-4de1-9622-a33185d350fd	remapp.exports.ct_export.ct_csv	19 Feb 2019, 8:51 p.m.	19 Feb 2019, 9 p.m.	SUCCESS

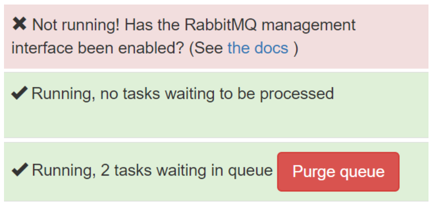
Fig. 25: Figure 2: The task administration page

RabbitMQ message broker

When tasks are created, they are sent to Celery to be processed via a message broker, RabbitMQ. Therefore this service must be running for any of the asynchronous tasks to execute - for example query-retrieve operations and exports. Normal function is indicated with a green status and a tick.

When there are Celery workers available to take tasks, they will be passed through immediately and the service status be green as seen in the middle image of Figure 3. When all the Celery workers are busy any additional tasks will be held with the RabbitMQ broker, and an option to purge the queue is made available as in the bottom image in Figure 3.

Celery asynchronous task queue



✖ Not running! Has the RabbitMQ management interface been enabled? (See the docs)

✔ Running, no tasks waiting to be processed

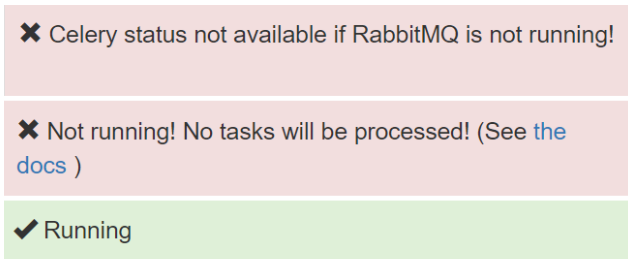
✔ Running, 2 tasks waiting in queue Purge queue

Fig. 26: Figure 3: RabbitMQ statuses: failed; running, no tasks waiting; running, two tasks waiting

Celery is used to manage the tasks that need to be carried out asynchronously. In OpenREM, these tend to be long running tasks such as exports and query-retrieve.

If RabbitMQ is not running, we can't tell if Celery is running; this is indicated by the first panel in Figure 4. If RabbitMQ is running but Celery isn't, this is indicated by the middle panel in Figure 4.

Finally, when Celery is running a green panel with a tick is presented.



✖ Celery status not available if RabbitMQ is not running!

✖ Not running! No tasks will be processed! (See the docs)

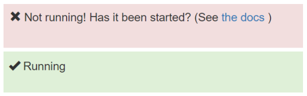
✔ Running

Fig. 27: Figure 4: Celery statuses: no RabbitMQ; not running; running

Flower - Celery monitoring tool

Flower enables monitoring of the Celery queues so we can see what is currently running, terminate if necessary, and maintain a log of previous tasks and how they finished. Flower logs are reset on restart of the service.

If Flower is not running, the Flower status panel will look like the top half of Figure 5. If it is running, the panel will look like the bottom half of Figure 5.



✖ Not running! Has it been started? (See the docs)

✔ Running

Fig. 28: Figure 5: Flower statuses: not running; running

3.10.4 Terminating running tasks

Active tasks are listed after the service status section, as seen in Figure 2. The number of active tasks is limited by the number of workers you have configured - see *Celery concurrency* for details.

It is possible to terminate any running tasks by clicking the red button. There is no confirmation step.

CHAPTER 4

Importing data to OpenREM

4.1 From local DICOM files

If you have RDSRs, DX or MG 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.

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 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.

To get started, you can make use of the in-built DICOM store that can be configured from within OpenREM:

4.2.1 DICOM Network Configuration

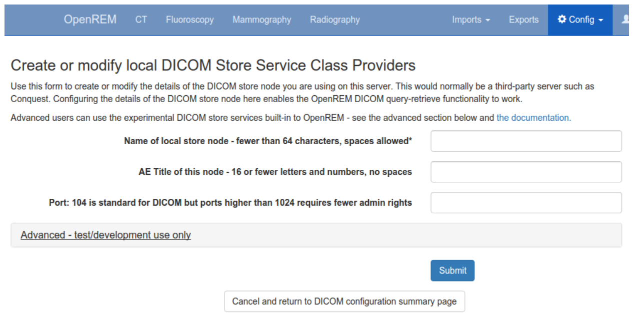
Configuring DICOM store nodes in OpenREM

You need to configure one or more DICOM Store nodes (Store Service Class Provider, or Store SCP) if you want either of the following:

- OpenREM to provide DICOM store functionality
- OpenREM to be able to query retrieve a third-party system (PACS or modality), using the OpenREM Store SCP or a third party one, such as Conquest

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



Create or modify local DICOM Store Service Class Providers

Use this form to create or modify the details of the DICOM store node you are using on this server. This would normally be a third-party server such as Conquest. Configuring the details of the DICOM store node here enables the OpenREM DICOM query-retrieve functionality to work.

Advanced users can use the experimental DICOM store services built-in to OpenREM - see the advanced section below and the documentation.

Name of local store node - fewer than 64 characters, spaces allowed*	
AE Title of this node - 16 or fewer letters and numbers, no spaces	
Port: 104 is standard for DICOM but ports higher than 1024 requires fewer admin rights	

Advanced - test/development use only

Submit

Cancel and return to DICOM configuration summary page

Fig. 1: Figure 1: DICOM Store SCP configuration

Native DICOM store node with direct import

> **Warning:** Native DICOM store functionality has not proved to be stable over long periods. Therefore we cannot recommend that you use this feature in a production environment. However, please do test it and help us to improve it if you are able to!

> **Warning:** If you use supervisord or similar on Linux, then you might not be able to use the web interface or possibly the auto-start service as new threads spawned for the Store SCP tend to get killed. This wouldn't prevent you starting the SCP in a shell. See Issue #337

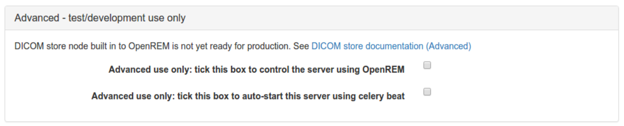
An OpenREM DICOM Store SCP (service class provider) enables modalities or PACS to send DICOM structured reports and images directly to OpenREM where they are imported into the database.

The Store SCP service receives the data, checks whether it is one of the objects that OpenREM can extract data from, and starts an import task if applicable.

The object is then left in the `dicom_in` folder in the `media` folder, or it is deleted, depending on the policy set in *Delete objects configuration*.

For native DICOM store nodes, you need to open the `Advanced - test/development use only` section (see figure 2):

- Control the server using OpenREM: this checkbox will enable OpenREM to create and control the node
- Auto-start the server using celery beat: if checked, and if *Celery periodic tasks: beat* is running, then OpenREM will attempt to start the store node whenever it finds it not to be running.


Advanced - test/development use only

DICOM store node built in to OpenREM is not yet ready for production. See DICOM store documentation (Advanced)

Advanced use only: tick this box to control the server using OpenREM ☐

Advanced use only: tick this box to auto-start this server using celery beat ☐

Fig. 2: Figure 2: DICOM Store SCP advanced configuration

Third-party DICOM store node for scripted import to OpenREM

If you are using Conquest or another third-party Store SCP to collect DICOM data, simply fill in the basic details as above without configuring the settings in the `Advanced` section. This will enable you to request remote hosts send data to your Store SCP in the *retrieve* part of the query-retrieve operation.

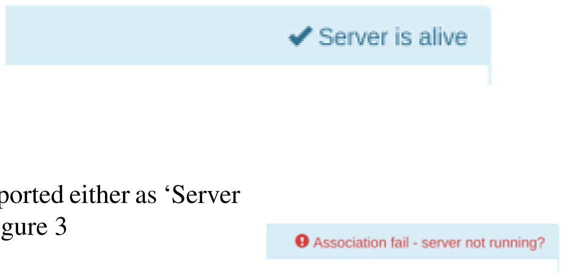
See *Conquest DICOM store node on Ubuntu 16.04* and *Running Conquest on Windows as a service* for more information about using Conquest with OpenREM

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


✓ Server is alive

Association fail - server not running?

Fig. 3: Figure 3: DICOM Store SCP status - Alive and Association failed

Controlling native Store SCP nodes

If a native Store SCP node is not running, then a `Start server` button will be presented at the bottom right. If it is running, this buttin will change to `Stop server`, and the `Delete` button will become inactive.

If the node is configured to be auto-started, and if *Celery periodic tasks: beat* is running, then each minute if the server is not started Celery will try to start the node. If you intend to stop the node for some reason, modify the configuration so that auto-start is not selected, then stop the server.

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 Third-party DICOM Stores

The DICOM store built in to OpenREM hasn't proved to be stable over the longer term with the current implementation and library that it depends on. This will be rectified in a future version, but for now we recommend you use a third-party DICOM store. Previous releases have recommended the Conquest DICOM server which is very good for this task. However, due to difficulties with installation on some platforms, we are now recommending the Orthanc DICOM server instead:

Orthanc Store configuration

Create Lua file

Open the following link in a new tab and copy and paste the content into a text editor. Save this as a new file called `openrem_orthanc_config.lua`. This can be saved anywhere, provided that Orthanc is able to access it:

•

Edit the top two sections

```
-------------------------------------------------------------------------------------
-- OpenREM python environment and other settings

-- Set this to the path and name of the python executable used by OpenREM
local python_executable = 'D:\\Server_Apps\\python27\\python.exe'

-- Set this to the path of the python scripts folder used by OpenREM
local python_scripts_path = 'D:\\Server_Apps\\python27\\Scripts\\'

-- Set this to the path where you want Orthanc to temporarily store DICOM files
local temp_path = 'E:\\conquest\\dicom\\'

-- Set this to 'mkdir' on Windows, or 'mkdir -p' on Linux
local mkdir_cmd = 'mkdir'
-- local mkdir_cmd = 'mkdir -p'

-- Set this to '\\'' on Windows, or '/' on Linux
local dir_sep = '\\'

-- 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

-- Set this to the path where you want to keep physics-related DICOM images
local physics_to_keep_folder = 'E:\\conquest\\dicom\\physics\\'

-- Set this to the path and name of your zip utility, and include any switches that
-- are needed to create an archive and include all files in a supplied folder
-- (used with physics-related images)
local zip_executable = 'D:\\Server_Apps\\7zip\\7za.exe a -r'
-- For Linux use the path to your zip command with the -r switch, such as:
-- local zip_executable = '/usr/bin/zip -r'

-- Set this to the path and name of your remove folder command, including switches
```

(continues on next page)

(continued from previous page)

```
-- for it to be quiet (used with physics-related images)
local rmdir_cmd = 'rmdir /s/q'
-- You can use the command 'rm -r' if you are using Linux:
-- local rmdir_cmd = 'rm -r'
-------------------------------------------------------------------------------------



-------------------------------------------------------------------------------------
-- User-defined lists that determine how Orthanc deals with certain studies

-- 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 lists: they will not be imported into OpenREM.
local manufacturers_to_ignore = {'Agfa', 'Agfa-Gevaert', 'Agfa-Gevaert AG', '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'}

-- 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'},
}
-------------------------------------------------------------------------------------
```

Guide to customising Orthanc configuration

python_executable Set this to the path and name of the python executable used by OpenREM:

```
# Linux, no virtualenv example:
local python_executable = '/usr/bin/python'
# Linux, using virtualenv example:
local python_executable = '/home/username/veopenrem/bin/python'
# Windows, not using virtualenv example:
local python_executable = 'C:\\Python27\\python.exe'
# Windows, using virtualenv example:
local python_executable = 'C:\\path\\to\\virtualenv\\Scripts\\python.exe'
```

python_scripts_path Set this to the path of the python scripts folder used by OpenREM:

```
# Linux, no virtualenv example:
local python_scripts_path = '/usr/local/bin/'
```

(continues on next page)

(continued from previous page)

```
# Linux, using virtualenv example:
local python_scripts_path = '/home/username/veopenrem/bin/'
# Windows, not using virtualenv example:
local python_scripts_path = 'C:\\Python27\\Scripts\\'
# Windows, using virtualenv example:
local python_scripts_path = 'C:\\path\\to\\virtualenv\\Scripts\\'
```

temp_path Set this to the path where you want Orthanc to temporarily store DICOM files. Note: the folder must exist and Orthanc must be able to write to it. On Ubuntu Linux the user is `orthanc`:

```
# Linux example:
local temp_path = '/tmp/orthanc/'
# To create the directory:
mkdir /tmp/orthanc
sudo chown orthanc /tmp/orthanc/
# Windows example:
local temp_path = 'C:\\Temp\\orthanc\\'
```

- Using Orthanc to collect Physics QA images:

  use_physics_filtering set this to `false` if you don't want to use this facility. If this is false, the other physics image related values don't matter. If it is `true`, then ensure that 7zip or similar (Windows) or zip (Linux) are installed, and:

  physics_to_keep_folder *Optional* Set this to the path where you want to keep physics-related DICOM images:

  ```
  local physics_to_keep_folder = 'E:\\conquest\\dicom\\physics\\'
  ```

  physics_to_keep 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:

  ```
  local manufacturers_to_ignore = {'Faxitron X-Ray LLC', 'Gendex-KaVo'}
  local model_names_to_ignore = {'CR 85', 'CR 75'}
  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'}
  ```

- Attempting to get dose data from CT studies with no RDSR using the OpenREMToshiba CT extractor

  use_toshiba_ct_extractor set this to `false` if you haven't installed the additional *Resources for creating RDSR for older Toshiba CT scanners* or do not wish to use this function. Otherwise:

  toshiba_extractor_systems A list of CT make and model pairs that you want to use 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. These will only be considered if an RDSR is not found with the study, otherwise that will be used in preference. The format is `{{'manufacturer', 'model'}, {'manufacturer two'}, {'model two'}}` etc. They will be matched against the names presented in the DICOM headers:

  ```
  local toshiba_extractor_systems = {
          {'Toshiba', 'Aquilion'},
          {'GE Medical Systems', 'Discovery STE'},
  }
  ```

Configure Orthanc to make use of the openrem_orthanc_config.lua file

Edit `orthanc.json` which can be found in:

- Ubuntu linux: `/etc/orthanc/`
- Windows: `C:\Program Files\Orthanc Server\Configuration\`

Find and edit the section below:

Linux:

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

Windows (note the double back-slash):

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

Check permissions

Linux

- orthanc user needs to be able to write to the OpenREM logs
- orthanc user needs to be able to write to the temp directory we specified

Windows

- Orthanc will be running as a local admin user, so should be able to function without any special consideration

Restart Orthanc

Ubuntu linux:

```
sudo service orthanc force-reload
```

Windows:

- Run `Services.msc` as an administrator
- Right-hand click on the Orthanc entry and select `Restart`

Conquest Store configuration

The instructions for installing Conquest were included in the *Install a DICOM Store service* docs.

Now OpenREM is installed, you need to configure `dicom.ini` and setup the import scripts

Import scripts

Creating bash scripts on linux

Create a bash script for each of RDSR, mammo, DX and Philips CT dose images, as required. They should have content something like the following. The examples that follow assume the files have been saved in the folder /etc/conquest-dicom-server but you can save them where you like and change the dicom.ini commands accordingly.

These scripts have a line in them to activate the virtual environment; this is done in the line . /var/dose/venv/bin/activate – you should change the path to your virtualenv or remove it if you have installed without using a virtualenv.

Eash script also has a line to delete the object after it has been imported – OpenREM can also do this by configuration, but the file will be written by the _conquest user, and OpenREM will not be running as that user. Therefore it is easier to have conquest delete the file. If you don't want them to be deleted, remove or comment out that line (add a # character to the start of the line).

- Radiation Dose Structured Reports
- Use which ever editor you are comfortable with – a good choice might be nano. For example:

```
sudo nano /etc/conquest-dicom-server/openrem-rdsr.sh
```

```
#!/bin/sh
#
# usage: ./openrem-rdsr.sh rdsrfilepath
#

# Get the name of the RDSR as variable 'rdsr'
rdsr="$1"

# Setup the python virtual environment - change to suit your path or remove if
# you are not using virtualenv
. /var/dose/venv/bin/activate

# Import RDSR into OpenREM
openrem_rdsr.py ${rdsr}

# Delete RDSR file - remove or comment (#) this line if you want the file to remain
rm ${rdsr}
```

Save and exit, then set the script to be executable:

```
sudo chmod +x /etc/conquest-dicom-server/openrem-rdsr.sh
```

And repeat for the other modality scripts below:

- Mammography images

```
sudo nano /etc/conquest-dicom-server/openrem-mg.sh
```

```
#!/bin/sh
#
```

(continues on next page)

(continued from previous page)

```
# usage: ./openrem-mg.sh mammofilepath
#

mamim="$1"

. /var/dose/venv/bin/activate

openrem_mg.py ${mamim}

rm ${mamim}
```

```
sudo chmod +x /etc/conquest-dicom-server/openrem-mg.sh
```

- Radiography images (DX, and CR that might be DX)

```
sudo nano /etc/conquest-dicom-server/openrem-dx.sh
```

```
#!/bin/sh
#
# usage: ./openrem-dx.sh dxfilepath
#

dxim="$1"

. /var/dose/venv/bin/activate

openrem_dx.py ${dxim}

rm ${dxim}
```

```
sudo chmod +x /etc/conquest-dicom-server/openrem-dx.sh
```

- Philips CT dose info images for Philips CT systems with no RDSR

```
sudo nano /etc/conquest-dicom-server/openrem-ctphilips.sh
```

```
#!/bin/sh
#
# usage: ./openrem-ctphilips.sh philipsctpath
#

philipsim="$1"

. /var/dose/venv/bin/activate

openrem_ctphilips.py ${philipsim}

rm ${philipsim}
```

```
sudo chmod +x /etc/conquest-dicom-server/openrem-ctphilips.sh
```

Creating batch scripts on windows

Create and save a bash script for each of RDSR, mammo, DX and Philips CT dose images, as required. They should have content something like the following.

These scripts assume there is no virtualenv in use. If you are using one, simply refer to the python executable and OpenREM script in your virtualenv rather than the system wide version.

- Radiation Dose Structured Reports

```
C:\Python27\python.exe C:\Python27\scripts\openrem_rdsr.py %1
del %1
```

- Mammography images

```
C:\Python27\python.exe C:\Python27\scripts\openrem_my.py %1
del %1
```

- Radiography images (DX, and CR that might be DX)

```
C:\Python27\python.exe C:\Python27\scripts\openrem_dx.py %1
del %1
```

- Philips CT dose info images for Philips CT systems with no RDSR

```
C:\Python27\python.exe C:\Python27\scripts\openrem_ctphilips.py %1
del %1
```

dicom.ini configuration

Example Windows Conquest dicom.ini file

Below is an example `dicom.ini` file, including comments describing the function of some sections. The file calls various lua scripts - see the Conquest import configuration document for an example - *Advanced Lua Conquest configuration (Windows)*.

The example `dicom.ini` file:

```
# This file contains configuration information for the DICOM server
# Do not edit unless you know what you are doing

[sscscp]
MicroPACS                = sscscp

# Network configuration: server name (AE title) and TCP/IP port number. You may wish
↪to add this OpenREM DICOM
# node to some of your imaging modalities, or to your PACS, so that you can send data
↪to OpenREM from these systems.
# You'll need to know the AE title, port number and IP address of this server when
↪doing this. Port 104 is commonly
# used for DICOM traffic. You may need to configure your server firewall to allow
↪network traffic on this port.
MyACRNema                = OPENREM
```

(continues on next page)

(continued from previous page)

```
TCPPort                    = 104

# Host, database, username and password for database. "localhost" means the server␣
↪that Conquest is running on.
# The SQLServer is blank to prevent the incoming DICOM objects from being added to␣
↪the Conquest database - this
# helps to avoid storing patient-identifiable data that you don't need to keep.
SQLHost                    = localhost
SQLServer                  =
Username                   =
Password                   =
SqLite                     = 1
DoubleBackSlashToDB        = 0
UseEscapeStringConstants = 0

# Configure server
ImportExportDragAndDrop  = 1
ZipTime                    = 05:
UIDPrefix                  = 1.2.826.0.1.3680043.2.135.736310.50024482
EnableComputedFields     = 1

# This option determins the folder structure used by Conquest when saving incoming␣
↪DICOM objects on the server.
# Option 4 saves as ID\seriesuid_series#_image#_timecounter.dcm. At my Trust the ID␣
↪is the patient NHS number.
FileNameSyntax             = 4

# Configuration of compression for incoming images and archival ("ul" saves images␣
↪using little endian explicit
# encoding).
DroppedFileCompression   = ul
IncomingCompression      = ul
ArchiveCompression       = ul

# For debug information
PACSName                   = OPENREM
OperatorConsole            = 127.0.0.1
DebugLevel                 = 0

# Configuration of disk(s) to store incoming DICOM objects.
MAGDeviceFullThreshHold  = 30
MAGDevices                 = 1
MAGDevice0                 = E:\conquest\dicom\


# Importing incoming data in to OpenREM
# The lua scripts that are called by some of these importers must be located in the␣
↪same folder as this dicom.ini
# file.

# DICOM Radiation Dose Structured Report (RDSR) objects
ImportConverter0 = ifequal "%V0008,0016","1.2.840.10008.5.1.4.1.1.88.67"; {save to␣
↪E:\conquest\dicom\sr\%o.dcm; openrem_import_rdsr.lua(E:\conquest\dicom\sr\%o.dcm::
↪%V0018,1000); destroy;}

# Mammography objects (modality MG)
ImportConverter1 = ifequal "%m", "MG"; { save to E:\conquest\dicom\mammo\%o.dcm;␣
↪openrem_import_mg.lua("E:\conquest\dicom\mammo\%o.dcm"); destroy; }
```

(continues on next page)

(continued from previous page)

```
# Digital radiography (modality DX)
ImportConverter2 = ifequal "%m", "DX"; { save to E:\conquest\dicom\dx\%o.dcm; openrem_
↪import_cr_or_dx.lua(E:\conquest\dicom\dx\%o.dcm::%V0008,0070::%V0008,1090::%V0008,
↪1010::%V0018,1020::%V0008,0020::%V0010,0010::%V0010,0020); destroy; }

# Computed radiography (modality CR). Note: some digital radiography systems send␣
↪their images as "CR" rather than "DX"
ImportConverter3 = ifequal "%m", "CR"; { save to E:\conquest\dicom\cr\%o.dcm; openrem_
↪import_cr_or_dx.lua(E:\conquest\dicom\cr\%o.dcm::%V0008,0070::%V0008,1090::%V0008,
↪1010::%V0018,1020::%V0008,0020::%V0010,0010::%V0010,0020); destroy; }

# Import converter for CT images. Conquest is configured to save images using the NHS␣
↪number as the folder name, so
# I think it makes sense to process the images by patient, rather than by study (
↪"process patient after"... rather
# than "process study after"...). If "process study" was used then the openrem_import_
↪ct.lua script may be run
# multiple times on the same folder if the folder contains more than one study for␣
↪that patient.
ImportConverter4 = ifequal "%V0008,1090","Brilliance 64"; { ifequal "%V0008,0016","1.
↪2.840.10008.5.1.4.1.1.7"; { save to E:\conquest\dicom\sr\%o.dcm; openrem_import_
↪ctphilips.lua("E:\conquest\dicom\sr\%o.dcm"); }; destroy; }
ImportConverter5 = ifequal "%V0008,1090","Brilliance 16P"; { ifequal "%V0008,0016","1.
↪2.840.10008.5.1.4.1.1.7"; { save to E:\conquest\dicom\sr\%o.dcm; openrem_import_
↪ctphilips.lua("E:\conquest\dicom\sr\%o.dcm"); }; destroy; }
ImportConverter6 = ifnotequal "%V0008,1090", "Brilliance 64"; { ifequal "%m", "CT"; {␣
↪process patient after 0 by openrem_import_ct.lua %p::%V0008,0070::%V0008,1090::
↪%V0018,1020::%V0008,1010::%V0010,0010::%V0010,0020::%V0008,0020::%V0018,1000; }; }

# Import converter for Presentation State objects - delete them
ImportConverter7 = ifequal "%m", "PR"; { destroy; }

# Import converter for Key Object Selection objects - delete them
ImportConverter8 = ifequal "%m", "KO"; { destroy; }

# Import converter for OT modality objects - delete them
ImportConverter9 = ifequal "%m", "OT"; { destroy; }

# Import converter for PT modality objects (PET) - delete them
ImportConverter10 = ifequal "%m", "PT"; { destroy; }

# Import converter for NM modality objects - delete them
ImportConverter11 = ifequal "%m", "NM"; { destroy; }

# Import converter for "Comprehensive SR Storage" type files - delete them
ImportConverter12 = ifequal "%V0008,0016","1.2.840.10008.5.1.4.1.1.88.33"; {destroy;}

# Import converter for "Basic Text SR Storage" type files - delete them
ImportConverter13 = ifequal "%V0008,0016","=BasicTextSRStorage"; {destroy;}

# Import converter for US modality objects - delete them
ImportConverter14 = ifequal "%m", "US"; { destroy; }

# Import converter for XA modality objects - delete them
ImportConverter15 = ifequal "%m", "XA"; { destroy; }
```

(continues on next page)

(continued from previous page)

```
# Import converter for PX modality objects (panoramic x-ray) - delete them
ImportConverter16 = ifequal "%m", "PX"; { destroy; }

# Import converter for PX modality objects (panoramic x-ray) - delete them
ImportConverter17 = ifequal "%m", "PX"; { destroy; }

# Import converter for XRayAngiographicImageStorage images (graphical dose reports␣
↪sent by the cath lab) - delete them
ImportConverter18 = ifequal "%V0008,0016","=XRayAngiographicImageStorage"; { destroy;␣
↪}

# Import converter for XRayAngiographicImageStorage images (graphical dose reports␣
↪sent by the cath lab) - delete them
ImportConverter19 = ifequal "%V0008,0016","1.2.840.10008.5.1.4.1.1.12.1"; { save to␣
↪E:\conquest\dicom\cath_lab_protocols\%o.dcm; openrem_zip_angiostorage.
↪lua(E:\conquest\dicom\cath_lab_protocols::E:\conquest\dicom\cath_lab_protocols\%o.
↪dcm::%V0008,0020::%V0008,0030::%V0008,0050); destroy; }

# Enhanced SR Storage objects
ImportConverter20 = ifequal "%V0008,0016","1.2.840.10008.5.1.4.1.1.88.22"; {save to␣
↪E:\conquest\dicom\sr\%o.dcm; openrem_import_rdsr.lua("E:\conquest\dicom\sr\%o.dcm");
↪ destroy;}

# Import converter for PX modality objects (panoramic x-ray) - delete them as the␣
↪systems we have contain no useful
# dose data information.
ImportConverter21 = ifequal "%m", "MR"; { destroy; }
```

Advanced Lua Conquest configuration (Windows)

Configuring Conquest DICOM server to automatically forward data to OpenREM

Simple rules using the Conquest dicom.ini file

The Conquest DICOM server can be configured to automatically run tasks when it receives specific types of DICOM object. For example, a script can be run when a DX image is received that will extract dose information into OpenREM; Conquest will then delete the original image.

These actions are set up in the `dicom.ini` file, located in the root of the Conquest installation folder (an example `dicom.ini` file is available here: *Example Windows Conquest dicom.ini file*).

An example import converter:

```
ImportModality1   = MG
ImportConverter1  = save to C:\conquest\dosedata\mammo\%o.dcm; system␣
↪C:\conquest\openrem-mam-launch.bat C:\conquest\dosedata\mammo\%o.dcm; destroy
```

`ImportModality1 = MG` tells Conquest that modality 1 is MG. The commands listed in the `ImportConverter1` line are then run on all incoming MG images.

The `ImportConverter` instructions are separated by semicolons; the above example has three commands:

- `save to C:\conquest\dosedata\mammo\%o.dcm` saves the incoming MG image to the specified folder with a file name set to the SOP instance UID contained in the image

- `system C:\conquest\openrem-mam-launch.bat C:\conquest\dosedata\mammo\%o.dcm` runs a DOS batch file, using the newly saved file as the argument. On my system this batch file runs the OpenREM `openrem_mg.py` import script
- `destroy` tells Conquest to delete the image that it has just received.

My system had three further import sections for DX, CR, and structured dose report DICOM objects:

```
# Import of DX images
ImportModality2   = DX
ImportConverter2  = save to C:\conquest\dosedata\dx\%o.dcm; system␣
↪C:\conquest\openrem-dx-launch.bat C:\conquest\dosedata\dx\%o.dcm; destroy

# Import of CR images
ImportModality3   = CR
ImportConverter3  = save to C:\conquest\dosedata\dx\%o.dcm; system␣
↪C:\conquest\openrem-dx-launch.bat C:\conquest\dosedata\dx\%o.dcm; destroy

# Import of structured dose reports (this checks the DICOM tag 0008,0016 to see if it␣
↪matches the value for a dose report)
ImportConverter4  = ifequal "%V0008,0016","1.2.840.10008.5.1.4.1.1.88.67"; {save to␣
↪C:\conquest\dosedata\sr\%o.dcm; system C:\conquest\openrem-sr-launch.bat
↪"C:\conquest\dosedata\sr\%o.dcm"; destroy}
```

However, I have since moved to calling lua scripts from Conquest, as described in the next section.

Advanced Conquest DICOM object handling using lua scripts

Conquest can be configured to use lua scripts to handle incoming DICOM objects. This enables more flexibility than the examples provided in the section above. For example, you may wish to keep all incoming images that contain the word `physics` in the patient name or id fields. You may also wish to direct images from different makes and models of CT scanner to different OpenREM import scripts. I use this technique to forward studies from some Toshiba Aquilion CX and CXL scanners to an importer that creates a DICOM RDSR object from the Toshiba dose summary images and information stored in the image tags. These particular scanners are not capable of producing their own RDSR objects directly. I use the same script to delete images from CT scanners that I cannot extract data from.

A lua script designed to handle any objects with modality `CT` can be called from `dicom.ini` in the following way:

```
ImportConverter6 = ifequal "%m", "CT"; { process patient after 0 by openrem_import_ct.
↪lua %p::%V0008,0070::%V0008,1090::%V0018,1020::%V0008,1010::%V0010,0010::%V0010,
↪0020::%V0008,0020::%V0018,1000; }
```

The above line will run a script called `openrem_import_ct.lua` once the complete patient data has been received by Conquest and a delay of 10 minutes has elapsed (the default). The following information is passed on to the script:

- `%p`, the path to a folder of DICOM objects
- `%V0008,0070`, manufacturer
- `%V0008,1090`, model
- `%V0018,1020`, software version
- `%V0008,1010`, station name
- `%V0010,0010`, patient name
- `%V0010,0020`, patient id
- `%V0008,0020`, study date

- `%V0018,1000`, device serial number

An example `openrem_import_ct.lua` script is shown below. It receives the parameters passed to it as single string. The individual parameters are recovered by splitting up the string using the `::` substring as a delimiter. The script keeps any images that contain `physics` in the patient name or id fields; it looks for any Philips dose info objects and imports these with the appropriate routine; it deletes images from scanners that cannot be used; it runs the Toshiba RDSR creation importer on images from older Toshiba CT scanners:

```
require "openrem_string_split"

-- It is assumed that 'openrem_rdsr.py' and 'openrem_ctphilips.py' are in the Python␣
↪scripts folder, and that this
-- and the Python executable location are included in the system path.

-- It is assumed that this code has been called from an ImportConverter within a␣
↪Conquest dicom.ini file, and passed
-- the following:
--    path to a folder of DICOM objects
--    the manufacturer (0008,0070)
--    model (0008,1090)
--    software version (0018,1020)
--    station name (0008,1010)
--    patient name (0010,0010)
--    patient id (0010,0020)
--    study date (0008,0020)
--    device serial number (0018,1000)

local physics_to_keep = {'physics'}
local physics_folder = 'E:\\conquest\\dicom\\physics_images\\'

local split_input_text = split(command_line, '::')
local study_folder_path = split_input_text[1]
local manufacturer = split_input_text[2]
local model_name = split_input_text[3]
local software_version = split_input_text[4]
local station_name = split_input_text[5]

local patient_name, patient_id, study_date, device_serial_number

if split_input_text[6] == nil then
  patient_name = ''
else
  patient_name = string.lower(split_input_text[6])
end

if split_input_text[7] == nil then
  patient_id = ''
else
  patient_id = string.lower(split_input_text[7])
end

if split_input_text[8] == nil then
  study_date = 'blank'
else
  study_date = split_input_text[8]
end

if split_input_text[9] == nil then
```

(continues on next page)

(continued from previous page)

```
  device_serial_number = 'blank'
else
  device_serial_number = split_input_text[9]
end

print(study_folder_path)

-- If any of the entries in physics_to_keep are present in the patient name or ID␣
↪then the image is assumed to be a
-- physics test, and is kept.
for i = 1, #physics_to_keep do
  if string.match(patient_name, physics_to_keep[i]) or string.match(patient_id,␣
↪physics_to_keep[i]) then
    print('Keeping the image: patient name is ' .. patient_name)
    print('and patient ID is ' .. patient_id)
    print('Trying to create folder ' .. physics_folder .. '\\' .. study_date)
    system('c:\\Windows\\system32\\cmd.exe /C mkdir ' .. physics_folder .. '\\' ..␣
↪study_date)
    print('Trying to copy to the following folder: ' .. study_folder_path .. ' ' ..␣
↪physics_folder .. '\\' .. study_date .. '\\')
    system('c:\\Windows\\system32\\cmd.exe /C copy ' .. study_folder_path .. '\\*.* '␣
↪.. physics_folder .. '\\' .. study_date .. '\\')
    return
  end
end

if (manufacturer == 'Philips' and model_name == 'Brilliance 64') then
  print('It is a Philips Brilliance 64')
  -- Look for a dose summary image and import it
  local files = assert(io.popen('dir /b ' .. study_folder_path))
  local output = files:read('*all')
  local file_list = split(output, '\n')

  for k, v in pairs(file_list) do
    current_file = study_folder_path .. '\\' .. v -- The fully qualified file name␣
↪and path (Windows-specific)
    readdicom(current_file)
    if Data.SOPClassUID == '1.2.840.10008.5.1.4.1.1.7' then
      system('D:\\Server_Apps\\python27\\python.exe d:\\Server_
↪Apps\\python27\\Scripts\\openrem_ctphilips.py ' .. current_file)
      print('The system command to import a Philips CT dose image has been executed␣
↪on: ' .. current_file)
    end
  end

  -- Delete the study from disk
  print('Complete. Deleting study images.')
  system('C:\\Windows\\system32\\cmd.exe /C rmdir /S /Q ' .. study_folder_path)
  return
end

-- Check for images from a Toshiba CT simulator - images are of no use - need RDSR
if (manufacturer == 'TOSHIBA' and station_name == 'AQ16LB_SCAN') then
  print('It is a Toshiba Aquilion LB study. Cannot make use of these images -␣
↪deleting them.')
  system('C:\\Windows\\system32\\cmd.exe /C rmdir /S /Q ' .. study_folder_path)
  print('The system command has been executed to delete the images from the server')
```

(continues on next page)

(continued from previous page)

```
  return
end

-- Toshiba Aquilion CX and CXL scanners - try and create an RDSR from the data
if (manufacturer == 'TOSHIBA' and model_name == 'Aquilion') then
  print('It is a Toshiba Aquilion. Running openrem_rdsr_toshiba_ct_from_dose_images.
↪py script: ' .. study_folder_path)
  system('d:\\Server_Apps\\python27\\python.exe d:\\Server_
↪Apps\\python27\\Scripts\\openrem_rdsr_toshiba_ct_from_dose_images.py ' .. study_
↪folder_path)
  print('The system command has been executed to create the rdsr and import it: ' ..␣
↪study_folder_path)
  -- The openrem_rdsr_toshiba_ct_from_dose_images.py routine deletes the study from␣
↪disk once the
  -- RDSR has been produced and imported in to OpenREM.
  return
end

-- Old Toshiba Asteion
if (manufacturer == 'TOSHIBA' and model_name == 'Asteion') then
  print('It is a Toshiba Asteion. Cannot make use of these images - deleting them: ' .
↪. study_folder_path)
  system('C:\\Windows\\system32\\cmd.exe /C rmdir /S /Q ' .. study_folder_path)
  print('The system command has been executed to delete the images from the server')
  return
end

-- Old Picker PQS
if (manufacturer == 'Picker International, Inc.' and model_name == 'PQS') then
  print('It is a Picker PQS. Cannot make use of these images - deleting them: ' ..␣
↪study_folder_path)
  system('C:\\Windows\\system32\\cmd.exe /C rmdir /S /Q ' .. study_folder_path)
  print('The system command has been executed to delete the images from the server')
  return
end

-- Image from a Vitrea workstation
if (manufacturer == 'Vital Images, Inc' and model_name == 'Vitrea 2') then
  print('It is a Vitrea 2. Cannot make use of these images - deleting them: ' ..␣
↪study_folder_path)
  system('C:\\Windows\\system32\\cmd.exe /C rmdir /S /Q ' .. study_folder_path)
  print('The system command has been executed to delete the images from the server')
  return
end
```

The above script depends on `openrem_string_split`:

```
function split(str, pat)
   local t = {}  -- NOTE: use {n = 0} in Lua-5.0
   local fpat = "(.-)" .. pat
   local last_end = 1
   local s, e, cap = str:find(fpat, 1)
   while s do
      if s ~= 1 or cap ~= "" then
     table.insert(t,cap)
      end
      last_end = e+1
```

(continues on next page)

(continued from previous page)

```
        s, e, cap = str:find(fpat, last_end)
    end
    if last_end <= #str then
        cap = str:sub(last_end)
        table.insert(t, cap)
    end
    return t
end
```

Preventing Conquest from adding incoming DICOM objects to the Conquest database

You may wish to prevent Conquest from adding patient data from incoming DICOM objects to the Conquest database, such as patient names and IDs. To do this set the SQLServer to a blank in the Conquest `dicom.ini` file:

```
# Host, database, username and password for database
SQLHost = localhost
# The SQLServer is blank below to prevent the incoming objects from being added to
↪the Conquest database.
SQLServer =
```

Setting the compression for Conquest incoming DICOM images and archives

Setting the following options to `ul` within `dicom.ini` will make Conquest store DICOM objects using little endian explicit encoding:

```
# Configuration of compression for incoming images and archival
DroppedFileCompression    = ul
IncomingCompression       = ul
ArchiveCompression        = ul
```

For my system the `ul` above matches the compression that is set for Conquest's known DICOM providers in the file `acrnema.map`, such as the Trust PACS and imaging modalities that have been set up to send data directly to Conquest.

Running Conquest on Windows as a service

Note: Content contributed by DJ Platten, with edit by ET McDonagh

This guide assumes Conquest has already been installed and runs ok. These instructions are based on Windows XP.

Run as a service

1. Make sure conquest isn't running.
2. Open a file browser and navigate to your conquest folder.
3. Right-hand click on the "ConquestDICOMServer.exe" file and choose "Run as. . . "

4. Enter the username and password of a Windows user with administrator rights.
5. Once conquest is running, click on the "Install server as NT service" on the "Configuration" tab.
6. Close the conquest Window.
7. Log in to Windows as a user with administrator rights.
8. Go to "Control panel" -> "Administrative Tools" -> "Services".
9. There will be a service with the same name as conquest's AE title. Right-hand click the mouse on this service and select "Properties".
10. On the "Log On" tab check the box that says "Allow service to interact with the desktop".
11. Click "Apply" then "OK".
12. Right-hand click on the service again and click "Restart".

The "Allow service to interact with the desktop" seems to be necessary for the batch to run that puts the dose report into OpenREM.

Firewall settings

Windows is able to change its firewall settings after you think everything is working ok! Assuming you have control of the firewall, add three port exceptions to the Windows firewall on the server computer: ports 80 and 443 for Apache, and whichever port that was chosen for conquest (104 is *the* port allocated to DICOM, but you may have used a higher port above 1024 for permissions reasons).

The firewall instructions at portforward.com were found to be a useful guide for this.

You ony need one of these - if you already have one installed it is probably easiest to stick to it.

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.

Warning: 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 *Conquest Store configuration* documents first to make sure you have created a DICOM Store node 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.

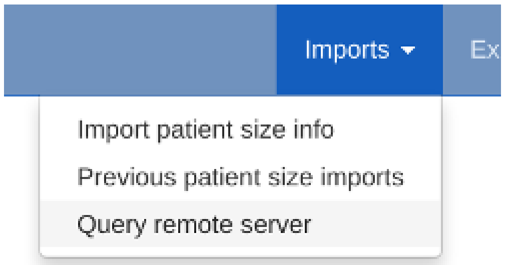


Fig. 4: Figure 1: Import Query-Retrieve menu

- 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.

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!

When you have finished the query parameters, click `Submit`

Review and retrieve

The progress of the query is reported on the right hand side. If nothing happens, ask the administrator to check if the celery queue is running.

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.

Query-retrieve using the command line interface

In a command window/shell, `python openrem_qr.py -h` should present you with the following output:

```
usage: openrem_qr.py [-h] [-ct] [-mg] [-fl] [-dx] [-f yyyy-mm-dd]
                     [-t yyyy-mm-dd] [-sd yyyy-mm-dd] [-tf hhmm] [-tt hhmm]
                     [-e string] [-i string] [-sne string] [-sni string]
                     [-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

optional arguments:
  -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. 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
  -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
```

(continues on next page)

(continued from previous page)

As an example, if you wanted to query the PACS for DX images on the 5th and 6th April 2010 with any study descriptions including `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:

```
python 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:

```
#!/bin/bash

. /var/dose/veopenrem/bin/activate  # activate virtualenv if you are using one,␣
↪modify or delete this line

ONEHOURAGO=$(date -d "1 hour ago" "+%Y-%m-%d")

python 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.

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
python D:\Server_Apps\python27\Scripts\openrem_qr.py 2 1 -ct -f $dateString -t
↪$dateString
```

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

New to OpenREM 0.9.0

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:

```
python 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 or station name filters have been added, and if the `StudyDescription` and/or `StationName` tags are returned by the remote server, the study response list is filtered accordingly.

– 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

– Another attempt is made to exclude or only-include if station name filters have been set

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.

Troubleshooting: openrem_qr.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 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_qr.py 2 1 -dx -f 2016-05-04 -t 2016-05-04 -e "imported"
```

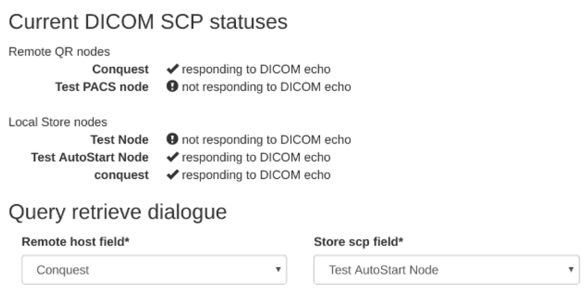
```
[04/May/2016 11:30:02] INFO [remapp.netdicom.qrscu:580] qrscu script called
[04/May/2016 11:30:02] INFO [remapp.netdicom.qrscu:595] Modalities are ['DX']
[04/May/2016 11:30:02] INFO [remapp.netdicom.qrscu:601] Date from: 2016-05-04
[04/May/2016 11:30:02] INFO [remapp.netdicom.qrscu:604] Date until: 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
[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
```

(continues on next page)

(continued from previous page)

```
[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
```

If you are using an OpenREM native storage node, then you might also like to review *Troubleshooting: openrem_store.log*



Current DICOM SCP statuses

Remote QR nodes

Conquest	✓ responding to DICOM echo
Test PACS node	❗ not responding to DICOM echo

Local Store nodes

Test Node	❗ not responding to DICOM echo
Test AutoStart Node	✓ responding to DICOM echo
conquest	✓ responding to DICOM echo

Query retrieve dialogue

Remote host field*: Conquest

Store scp field*: Test AutoStart Node

Fig. 5: Figure 2: Local and remote QR statuses

CHAPTER 5

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 or radiographic 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

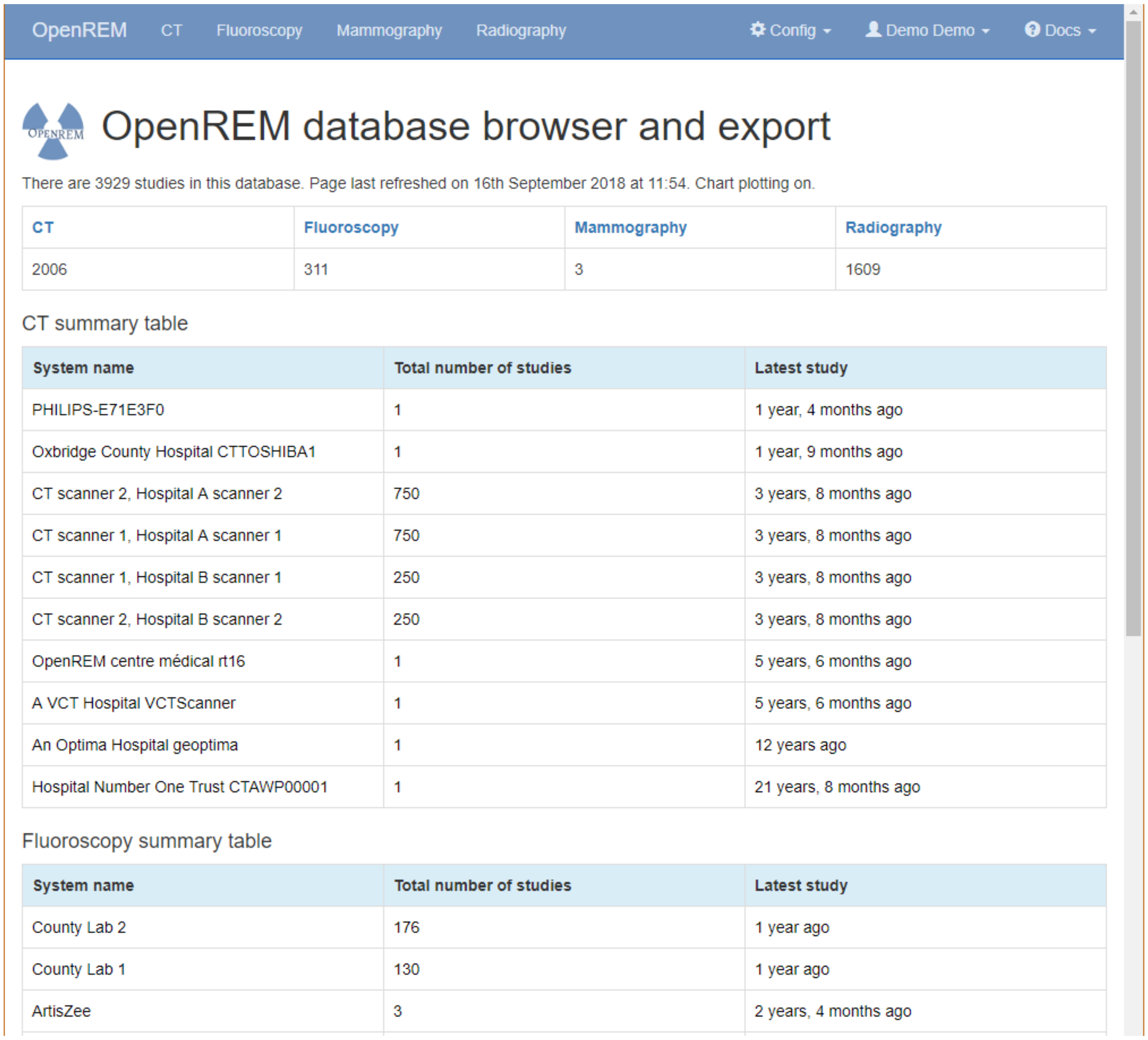
This image shows the CT studies view, available to any logged in user, filtered by entering terms in the boxes on the right hand side to show just the studies where the modality manufacturer name includes the term 'Siemens':

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.

5.2.1 CT: acquisition type restriction

The CT study filters includes the ability to restrict the displayed studies by acquisition type, such as `Spiral` or `Axial`. By default none of these are selected, and studies containing all acquisition types are shown. Ticking one of the acquisition type restrictions and clicking submit will filter the displayed studies to only include those which have



OpenREM database browser and export

There are 3929 studies in this database. Page last refreshed on 16th September 2018 at 11:54. Chart plotting on.

CT	Fluoroscopy	Mammography	Radiography
2006	311	3	1609

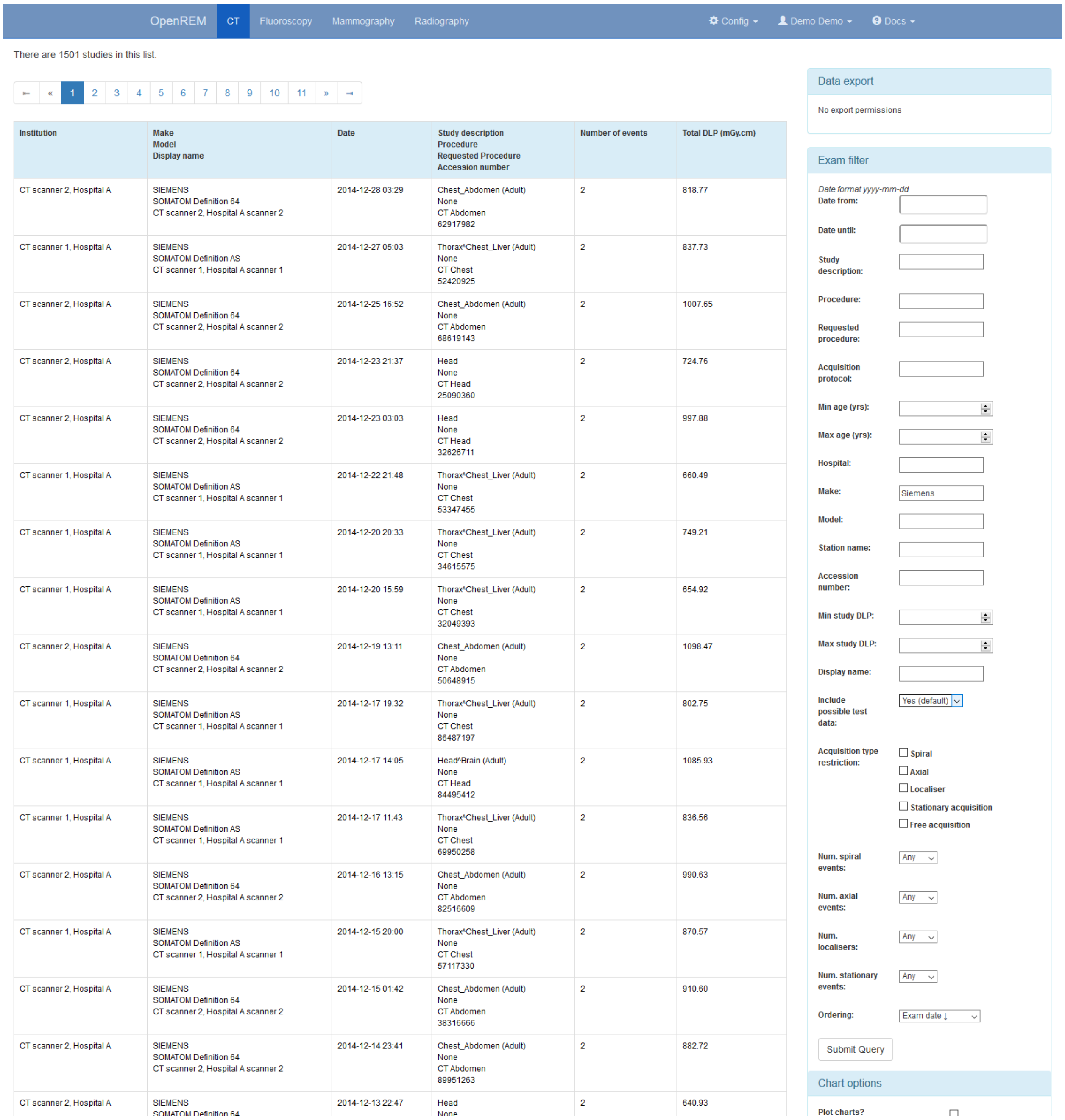
CT summary table

System name	Total number of studies	Latest study
PHILIPS-E71E3F0	1	1 year, 4 months ago
Oxbridge County Hospital CTTOSHIBA1	1	1 year, 9 months ago
CT scanner 2, Hospital A scanner 2	750	3 years, 8 months ago
CT scanner 1, Hospital A scanner 1	750	3 years, 8 months ago
CT scanner 1, Hospital B scanner 1	250	3 years, 8 months ago
CT scanner 2, Hospital B scanner 2	250	3 years, 8 months ago
OpenREM centre médical rt16	1	5 years, 6 months ago
A VCT Hospital VCTScanner	1	5 years, 6 months ago
An Optima Hospital geoptima	1	12 years ago
Hospital Number One Trust CTAWP00001	1	21 years, 8 months ago

Fluoroscopy summary table

System name	Total number of studies	Latest study
County Lab 2	176	1 year ago
County Lab 1	130	1 year ago
ArtisZee	3	2 years, 4 months ago

Fig. 1: OpenREM homepage screenshot



There are 1501 studies in this list.

⇤ « 1 2 3 4 5 6 7 8 9 10 11 » ⇥

Institution	Make Model Display name	Date	Study description Procedure Requested Procedure Accession number	Number of events	Total DLP (mGy.cm)
CT scanner 2, Hospital A	SIEMENS SOMATOM Definition 64 CT scanner 2, Hospital A scanner 2	2014-12-28 03:29	Chest_Abdomen (Adult) None CT Abdomen 62917982	2	818.77
CT scanner 1, Hospital A	SIEMENS SOMATOM Definition AS CT scanner 1, Hospital A scanner 1	2014-12-27 05:03	Thorax^Chest_Liver (Adult) None CT Chest 52420925	2	837.73
CT scanner 2, Hospital A	SIEMENS SOMATOM Definition 64 CT scanner 2, Hospital A scanner 2	2014-12-25 16:52	Chest_Abdomen (Adult) None CT Abdomen 68619143	2	1007.65
CT scanner 2, Hospital A	SIEMENS SOMATOM Definition 64 CT scanner 2, Hospital A scanner 2	2014-12-23 21:37	Head None CT Head 25090360	2	724.76
CT scanner 2, Hospital A	SIEMENS SOMATOM Definition 64 CT scanner 2, Hospital A scanner 2	2014-12-23 03:03	Head None CT Head 32626711	2	997.88
CT scanner 1, Hospital A	SIEMENS SOMATOM Definition AS CT scanner 1, Hospital A scanner 1	2014-12-22 21:48	Thorax^Chest_Liver (Adult) None CT Chest 53347455	2	660.49
CT scanner 1, Hospital A	SIEMENS SOMATOM Definition AS CT scanner 1, Hospital A scanner 1	2014-12-20 20:33	Thorax^Chest_Liver (Adult) None CT Chest 34615575	2	749.21
CT scanner 1, Hospital A	SIEMENS SOMATOM Definition AS CT scanner 1, Hospital A scanner 1	2014-12-20 15:59	Thorax^Chest_Liver (Adult) None CT Chest 32049393	2	654.92
CT scanner 2, Hospital A	SIEMENS SOMATOM Definition 64 CT scanner 2, Hospital A scanner 2	2014-12-19 13:11	Chest_Abdomen (Adult) None CT Abdomen 50648915	2	1098.47
CT scanner 1, Hospital A	SIEMENS SOMATOM Definition AS CT scanner 1, Hospital A scanner 1	2014-12-17 19:32	Thorax^Chest_Liver (Adult) None CT Chest 86487197	2	802.75
CT scanner 1, Hospital A	SIEMENS SOMATOM Definition AS CT scanner 1, Hospital A scanner 1	2014-12-17 14:05	Head^Brain (Adult) None CT Head 84495412	2	1085.93
CT scanner 1, Hospital A	SIEMENS SOMATOM Definition AS CT scanner 1, Hospital A scanner 1	2014-12-17 11:43	Thorax^Chest_Liver (Adult) None CT Chest 69950258	2	836.56
CT scanner 2, Hospital A	SIEMENS SOMATOM Definition 64 CT scanner 2, Hospital A scanner 2	2014-12-16 13:15	Chest_Abdomen (Adult) None CT Abdomen 82516609	2	990.63
CT scanner 1, Hospital A	SIEMENS SOMATOM Definition AS CT scanner 1, Hospital A scanner 1	2014-12-15 20:00	Thorax^Chest_Liver (Adult) None CT Chest 57117330	2	870.57
CT scanner 2, Hospital A	SIEMENS SOMATOM Definition 64 CT scanner 2, Hospital A scanner 2	2014-12-15 01:42	Chest_Abdomen (Adult) None CT Abdomen 38316666	2	910.60
CT scanner 2, Hospital A	SIEMENS SOMATOM Definition 64 CT scanner 2, Hospital A scanner 2	2014-12-14 23:41	Chest_Abdomen (Adult) None CT Abdomen 89951263	2	882.72
CT scanner 2, Hospital A	SIEMENS SOMATOM Definition 64	2014-12-13 22:47	Head None	2	640.93

Data export

No export permissions

Exam filter

Date format yyyy-mm-dd

Date from:	
Date until:	
Study description:	
Procedure:	
Requested procedure:	
Acquisition protocol:	
Min age (yrs):	
Max age (yrs):	
Hospital:	
Make:	Siemens
Model:	
Station name:	
Accession number:	
Min study DLP:	
Max study DLP:	
Display name:	
Include possible test data:	Yes (default)
Acquisition type restriction:	☐ Spiral ☐ Axial ☐ Localiser ☐ Stationary acquisition ☐ Free acquisition
Num. spiral events:	Any
Num. axial events:	Any
Num. localisers:	Any
Num. stationary events:	Any
Ordering:	Exam date ↓

Submit Query

Chart options

Plot charts? ☐

Fig. 2: Filtering CT studies

at least one acquisition of the selected type. Ticking more than one acquisition type restriction will show studies that contain at least one acquisition of any of the selected acquisition types.

Acquisition-level charts will only show acquisition protocols that correspond to the chosen acquisition type restrictions. Acquisitions of any other type present within the study data will not be shown on these charts.

Study- and request-level charts will show studies that have at least one acquisition of a type matching the selected restrictions: these studies and requests may contain acquisition types other than those chosen in the restrictions.

For example, if all studies contain a mixture of `Axial` and `Spiral` acquisitions and the user ticks the `Axial` acquisition type restriction then all studies will still be displayed in the filtered list. However, any acquisition-level charts will only show `Axial` acquisitions: the `Spiral` acquisitions present in the studies will not be shown in the chart. Any study- or request-level chart will show all the data as every one of them includes at least one `Axial` acquisition.

5.2.2 CT: specifying number of event types

In addition (or as an alternative) to specifying that studies must have at least one of one of the acquisition type restrictions, if any are selected, it is also possible to filter for studies that have specific numbers of each acquisition 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:

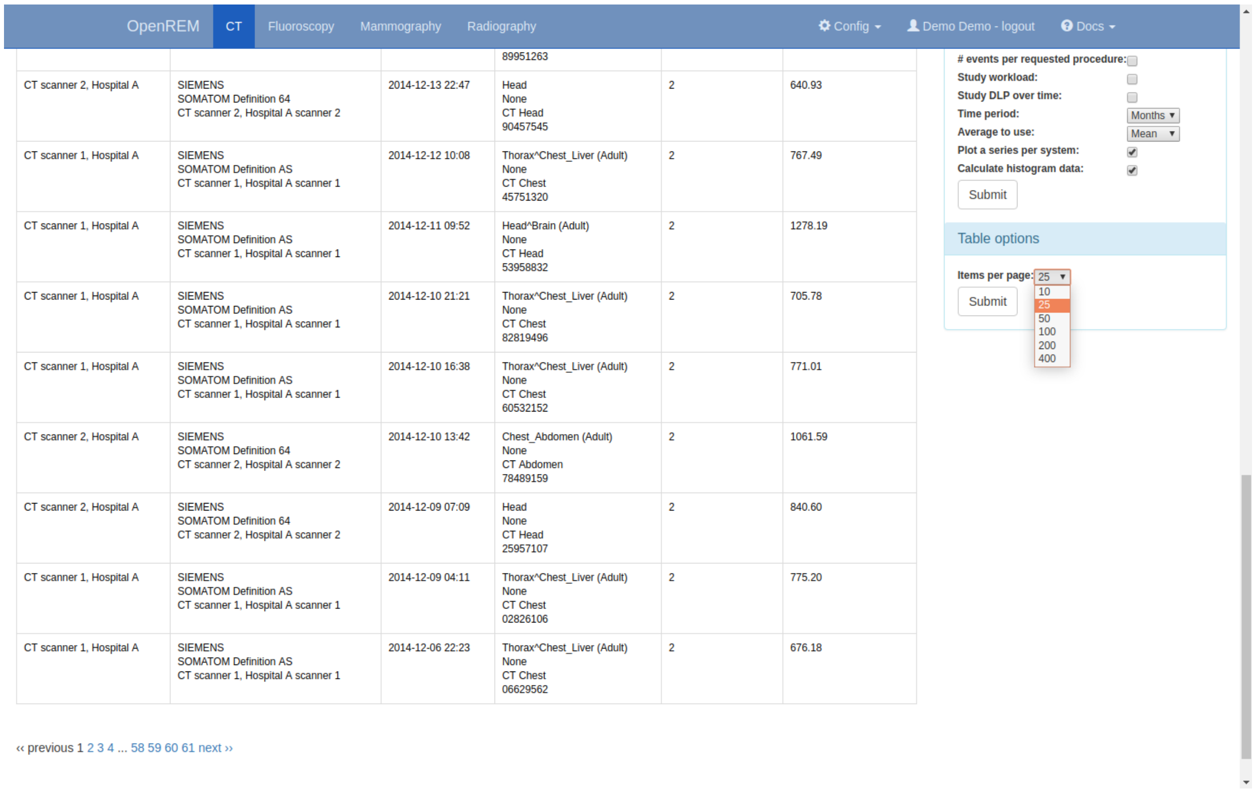
5.4 Viewing study details

By clicking on the study description link (in blue), you can see more details for an individual 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.

Note: For fluoroscopy the table showing details of each exposure can be sorted by clicking on the table headings.



			89951263		
CT scanner 2, Hospital A	SIEMENS SOMATOM Definition 64 CT scanner 2, Hospital A scanner 2	2014-12-13 22:47	Head None CT Head 90457545	2	640.93
CT scanner 1, Hospital A	SIEMENS SOMATOM Definition AS CT scanner 1, Hospital A scanner 1	2014-12-12 10:08	Thorax^Chest_Liver (Adult) None CT Chest 45751320	2	767.49
CT scanner 1, Hospital A	SIEMENS SOMATOM Definition AS CT scanner 1, Hospital A scanner 1	2014-12-11 09:52	Head^Brain (Adult) None CT Head 53958832	2	1278.19
CT scanner 1, Hospital A	SIEMENS SOMATOM Definition AS CT scanner 1, Hospital A scanner 1	2014-12-10 21:21	Thorax^Chest_Liver (Adult) None CT Chest 82819496	2	705.78
CT scanner 1, Hospital A	SIEMENS SOMATOM Definition AS CT scanner 1, Hospital A scanner 1	2014-12-10 16:38	Thorax^Chest_Liver (Adult) None CT Chest 60532152	2	771.01
CT scanner 2, Hospital A	SIEMENS SOMATOM Definition 64 CT scanner 2, Hospital A scanner 2	2014-12-10 13:42	Chest_Abdomen (Adult) None CT Abdomen 78489159	2	1061.59
CT scanner 2, Hospital A	SIEMENS SOMATOM Definition 64 CT scanner 2, Hospital A scanner 2	2014-12-09 07:09	Head None CT Head 25957107	2	840.60
CT scanner 1, Hospital A	SIEMENS SOMATOM Definition AS CT scanner 1, Hospital A scanner 1	2014-12-09 04:11	Thorax^Chest_Liver (Adult) None CT Chest 02826106	2	775.20
CT scanner 1, Hospital A	SIEMENS SOMATOM Definition AS CT scanner 1, Hospital A scanner 1	2014-12-06 22:23	Thorax^Chest_Liver (Adult) None CT Chest 06629562	2	676.18

‹‹ previous 1 2 3 4 ... 58 59 60 61 next ››

# events per requested procedure:	☐
Study workload:	☐
Study DLP over time:	☐
Time period:	Months
Average to use:	Mean
Plot a series per system:	☑
Calculate histogram data:	☑

Submit

Table options

Items per page: 25 (10, [✓ 25], 50, 100, 200, 400)

Submit

Fig. 3: Setting the number of studies per page

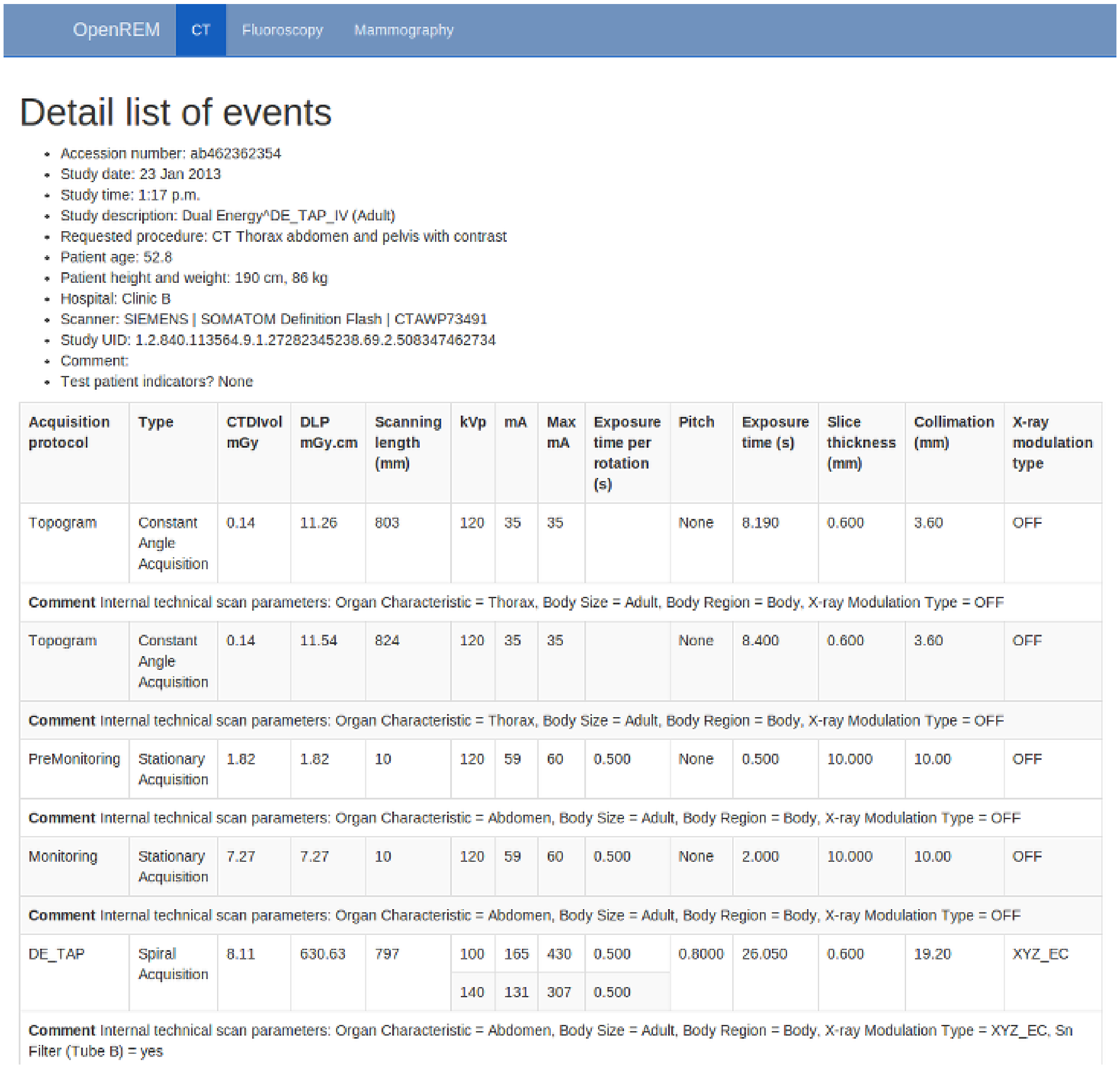
5.4.1 A note on time data for fluoroscopy studies

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.



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													

Fig. 4: Individual CT study

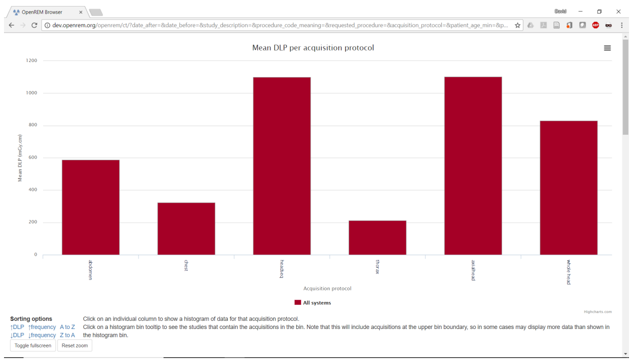
CHAPTER 6

Charts

6.1 Chart types

6.1.1 1. Bar chart of average values across a number of categories

An example of mean DAP per acquisition type is shown in figure 1.


OpenREM Browser
dev.openrem.org/openrem/ct/?date_after=&date_before=&study_description=&procedure_code_meaning=&requested_procedure=&acquisition_protocol=&patient_age_min=&p...
Mean DLP per acquisition protocol
Mean DLP (mGy.cm)
1200
1000
800
600
400
200
0
abdomen
chest
headseq
thorax
axialhead
whole head
Acquisition protocol
All systems
Highcharts.com
Toggle fullscreen
Reset zoom

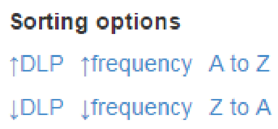
Sorting options
↑DLP ↑frequency A to Z
↓DLP ↓frequency Z to A

Click on an individual column to show a histogram of data for that acquisition protocol.
Click on a histogram bin tooltip to see the studies that contain the acquisitions in the bin. Note that this will include acquisitions at the upper bin boundary, so in some cases may display more data than shown in the histogram bin.

Fig. 1: Figure 1: Bar chart of mean DLP per acquisition

Below each bar chart there are options to sort the order of the data. This can be ascending or descending by average value, size of data sample, or alphabetically (figure 2).

Clicking on an entry in the bar chart legend toggles the display of the corresponding series on the chart.



Sorting options
↑DLP ↑frequency A to Z
↓DLP ↓frequency Z to A

Fig. 2: Figure 2: Bar chart sorting options

If you have histogram calculation switched on then clicking on an individual data point on a bar chart will take you to a histogram of the data for that point so that you can see the shape of the value's distribution (figure 3).

Bar charts can be plotted with a series per x-ray system (figure 4). This can be toggled using the *Plot a series per system* checkbox in the *Chart options*.

Clicking the left-hand mouse button on the chart background and dragging left or right selects part of the se-

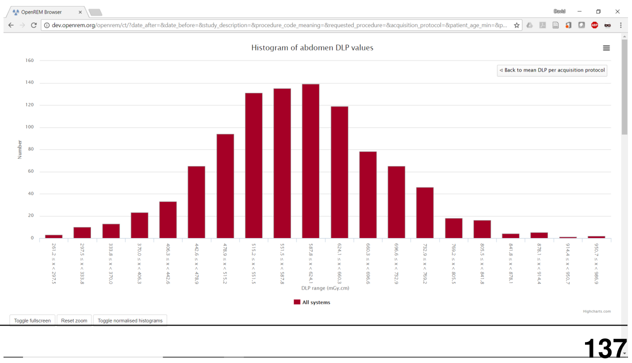


Fig. 3: Figure 3: Histogram of abdomen DLP values

ries. Releasing the mouse button zooms in on this selection. A `Reset zoom` button appears when zoomed in: clicking this resets the chart so that the full series can be seen again. The zoom feature works on both the main series and the histograms. The zooming can be useful when there is a category on the chart that has a very low value compared to others. Zooming in on this category will enable the low values to be seen, as the chart rescales the y-axis after the zoom.

Clicking on the *Toggle data table* button toggles the display of an HTML table containing the data from the current chart. This button is hidden if you are viewing the chart in full screen mode. An example showing a data table is shown in figure 5.

If the the bar chart that you are viewing shows more than one series then clicking on a category name on the x-axis will take you to a plot that shows multiple histograms: one for each series (figure 6).

If the bar chart that you are viewing shows more than one series then buttons are available to *Hide all series*, *Show all series*, and *Toggle all series*. These provide a quick way to switch which series are being displayed without having to click on individual series in the chart legend.

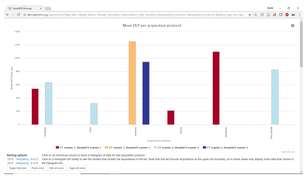
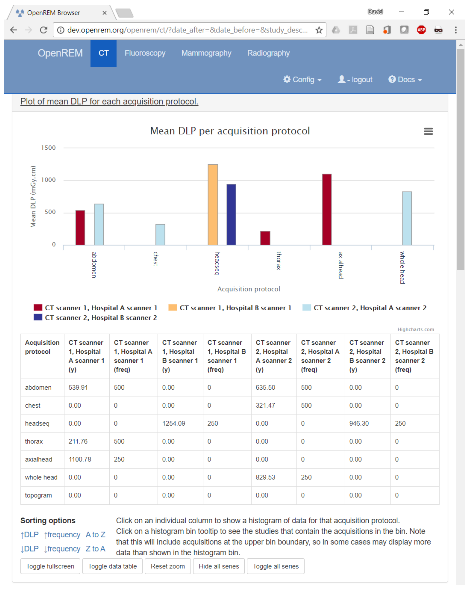


Fig. 4: Figure 4: Bar chart of mean DLP (one system per series)


OpenREM Browser
dev.openrem.org/openrem/ct/?date_after=&date_before=&study_desc...
OpenREM
CT
Fluoroscopy
Mammography
Radiography
Config
- logout
Docs
Plot of mean DLP for each acquisition protocol.
Mean DLP per acquisition protocol
Mean DLP (mGy.cm)
Acquisition protocol
CT scanner 1, Hospital A scanner 1
CT scanner 1, Hospital B scanner 1
CT scanner 2, Hospital A scanner 2
CT scanner 2, Hospital B scanner 2
Highcharts.com

Acquisition protocol	CT scanner 1, Hospital A scanner 1 (y)	CT scanner 1, Hospital A scanner 1 (freq)	CT scanner 1, Hospital B scanner 1 (y)	CT scanner 1, Hospital B scanner 1 (freq)	CT scanner 2, Hospital A scanner 2 (y)	CT scanner 2, Hospital A scanner 2 (freq)	CT scanner 2, Hospital B scanner 2 (y)	CT scanner 2, Hospital B scanner 2 (freq)
abdomen	539.91	500	0.00	0	635.50	500	0.00	0
chest	0.00	0	0.00	0	321.47	500	0.00	0
headseq	0.00	0	1254.09	250	0.00	0	946.30	250
thorax	211.76	500	0.00	0	0.00	0	0.00	0
axialhead	1100.78	250	0.00	0	0.00	0	0.00	0
whole head	0.00	0	0.00	0	829.53	250	0.00	0
topogram	0.00	0	0.00	0	0.00	0	0.00	0

Sorting options

↑DLP ↑frequency A to Z

↓DLP ↓frequency Z to A

Click on an individual column to show a histogram of data for that acquisition protocol. Click on a histogram bin tooltip to see the studies that contain the acquisitions in the bin. Note that this will include acquisitions at the upper bin boundary, so in some cases may display more data than shown in the histogram bin.

Fig. 5: Figure 5: Bar chart with data table displayed

The histogram data can be plotted as absolute values, or be normalised to a value of 1.0 (figure 7). This can be toggled by clicking on the button that is shown below the histogram plots. The normalisation can be useful when trying to compare the shape of several histograms, especially when some histograms have much less data than others.

Each histogram data point includes a text link that appears when the mouse pointer moves over it. Clicking on this link will filter the displayed studies, showing those that correspond to what is contained in the histogram bin.

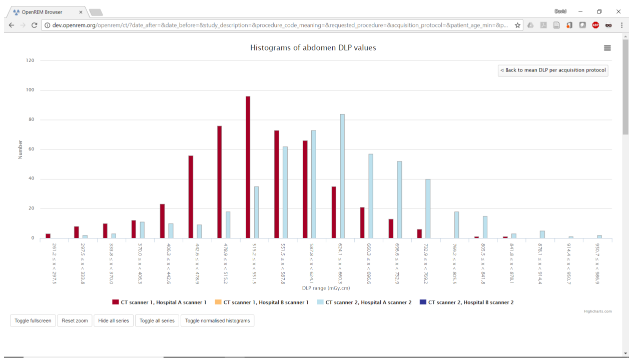

OpenREM Browser
Histograms of abdomen DLP values
< Back to mean DLP per acquisition protocol
Number
DLP range (mGy.cm)
CT scanner 1, Hospital A scanner 1
CT scanner 1, Hospital B scanner 1
CT scanner 2, Hospital A scanner 2
CT scanner 2, Hospital B scanner 2
Toggle fullscreen
Reset zoom
Hide all series
Toggle all series
Toggle normalised histograms

Fig. 6: Figure 6: Histogram of abdomen DLP values, one series per system

Clicking on a legend entry toggles the visibility of the corresponding series.

Only data with non-zero and non-blank dose values are included in the chart data calculations.

6.1.2 2. Pie chart showing the frequency of each item in a category

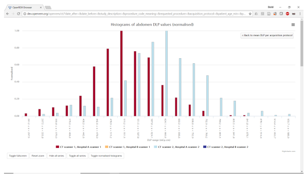


Fig. 7: Figure 7: Normalised histogram of abdomen DLP, one series per system

Figure 8 shows a pie chart of the number of acquisitions made for every acquisition protocol present in the tabulated data.

Clicking on any of the pie chart segments will filter the displayed studies, showing only the studies that correspond to what is contained in that segment. As for the bar charts, this doesn't work perfectly, as the category filtering isn't exact.

Only data with non-zero and non-blank dose values are included in the chart data calculations.

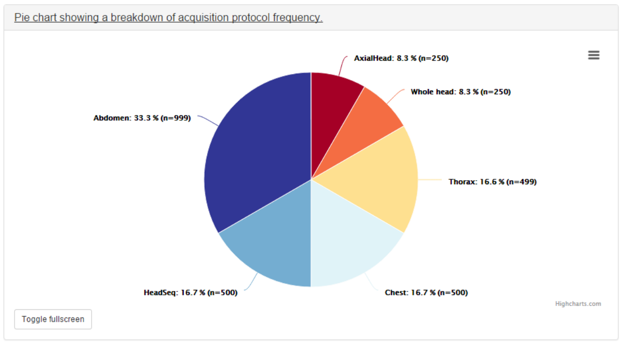


Fig. 8: Figure 8: Pie chart of acquisition frequency

6.1.3 3. Line chart showing how an average value changes over time

A line is plotted for each category, with a point calculated every day, week, month or year. This can be a good way of looking at how things have changed over time. For example, the mean DLP of each study type, calculated with a data point per month is shown in figure 9.

Clicking the left-hand mouse button on the chart and dragging left or right across a range of dates and then releasing the mouse button will zoom in on that selection.

Clicking on a legend entry toggles the visibility of the corresponding series.

Only data with non-zero and non-blank dose values are included in the chart data calculations.

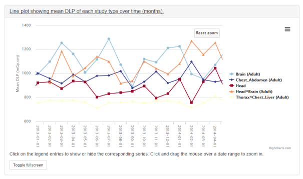
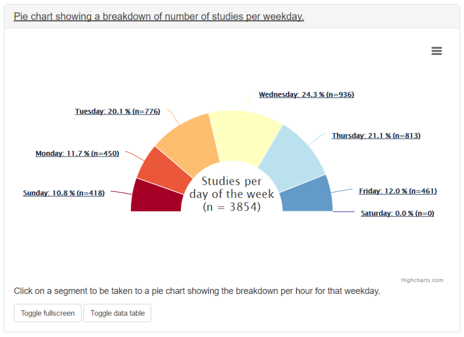


Fig. 9: Figure 9: Line chart of mean DLP per study type over time

6.1.4 4. Pie chart showing the number of events per day of the week

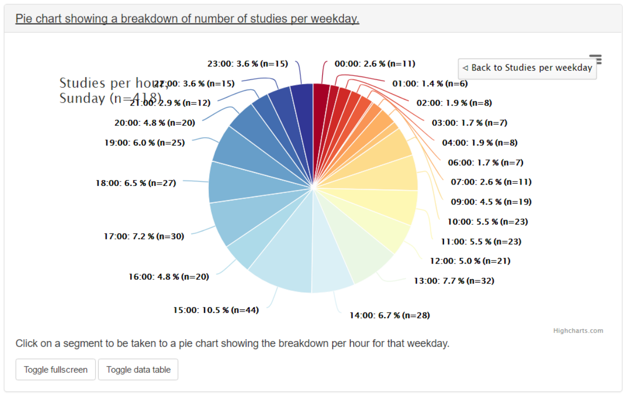


Click on a segment to be taken to a pie chart showing the breakdown per hour for that weekday.

Fig. 10: Figure 10: Pie chart of study workload per day of the week

Each segment represents a day of the week, and shows the number of events that have taken place on that day (figure 10). Clicking on one of the segments will take you to a pie chart that shows the number of events per on that day (figure 11).

All data, including zero blank dose values are included in the data calculations for this chart type.



Click on a segment to be taken to a pie chart showing the breakdown per hour for that weekday.

Fig. 11: Figure 11: Pie chart of study workload per hour in a day

6.1.5 5. Scatter plot showing one value vs. another

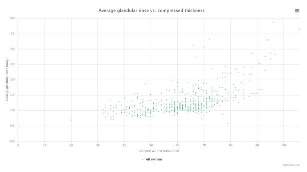


Fig. 12: Figure 12: Scatter plot of average glandular dose vs. compressed thickness

This plot type shows a data point per event (figure 12). The series name and data values are shown when the mouse cursor is positioned over a data point.

These can be plotted with a series per x-ray system (figure 13). This can be toggled using the *Plot a series per system* checkbox in the *Chart options*.

Clicking the left-hand mouse button on the chart and dragging a rectangular region will zoom in on that selection of the chart. A Reset zoom button appears when zoomed in: clicking this resets the chart so that the full series can be seen again.

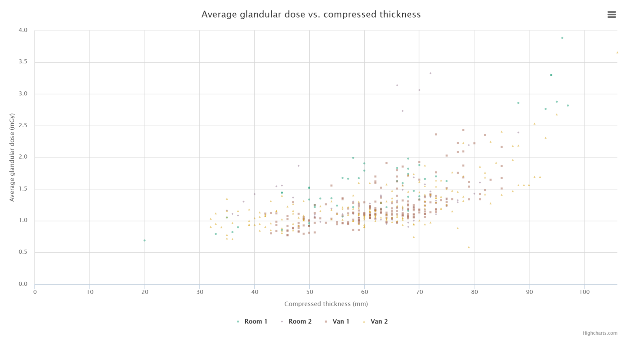


Fig. 13: Figure 13: Scatter plot of average glandular dose vs. compressed thickness; one series per system

Clicking on a system's legend entry toggles the display of the corresponding series on the chart.

Only data with non-zero and non-blank dose values are included in the chart data calculations.

6.2 Exporting chart data

An image file of a chart can be saved using the menu in the top-right hand side of any of the charts. The same menu can be used to save the data used to plot a chart: the data can be downloaded in either csv or xls format.

6.3 Chart options

Chart options can be configured by choosing the `Chart options` item from the `User options` menu on the OpenREM homepage (figure 13).

CT and radiographic plot options can also be set from their respective summary pages.

The first option, `Plot charts?`, determines whether any plots are shown. This also controls whether the data for the plots is calculated by OpenREM.

Switching `Case-insensitive categories` on will force chart categories to be lowercase. This can be helpful if several rooms use the same wording but with different capitalisation for acquisition protocol, study description or requested procedure. With this option switched on then all rooms with the same wording, irrespective of capitalisation, will be shown side-by-side under the same single category. With the option switched off there will be a seperate category for each differently capitalised category.

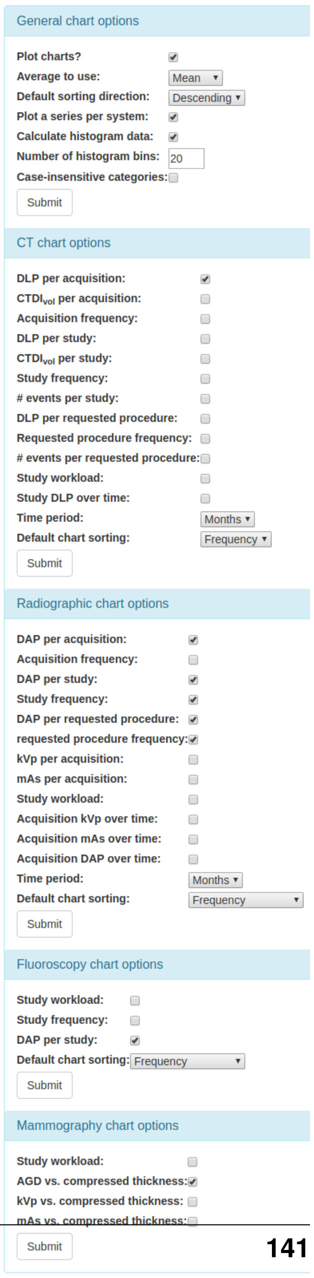
Some plot data is slow to calculate when there is a large amount of data: some users may prefer to leave `Plot charts?` off for performance reasons. `Plot charts?` can be switched on and activated with a click of the `Submit` button after the data has been filtered.

The user can also switch off chart plotting by clicking on the `Switch charts off` link in the `User options` menu in the navigation bar at the top of any OpenREM page, as shown in figure 14.

The user can choose whether the data displayed on the charts is the mean, median or both by using the drop-down `Average to use` selection. Only the bar charts can display both mean and median together. Other charts display just median data when this option is selected.

The charts can be sorted by either bar height, frequency or alphabetically by category. The default sorting direction can be set to ascending or descending using the drop-down list near the top of the `chart options`.

A user's chart options can also be configured by an administrator via OpenREM's user administration page.



General chart options

Plot charts?	✓
Average to use:	Mean
Default sorting direction:	Descending
Plot a series per system:	✓
Calculate histogram data:	✓
Number of histogram bins:	20
Case-insensitive categories:	☐

CT chart options

DLP per acquisition:	✓
$CTDI_{vol}$ per acquisition:	☐
Acquisition frequency:	☐
DLP per study:	☐
$CTDI_{vol}$ per study:	☐
Study frequency:	☐
# events per study:	☐
DLP per requested procedure:	☐
Requested procedure frequency:	☐
# events per requested procedure:	☐
Study workload:	☐
Study DLP over time:	☐
Time period:	Months
Default chart sorting:	Frequency

Radiographic chart options

DAP per acquisition:	✓
Acquisition frequency:	☐
DAP per study:	✓
Study frequency:	✓
DAP per requested procedure:	✓
requested procedure frequency:	✓
kVp per acquisition:	☐
mAs per acquisition:	☐
Study workload:	☐
Acquisition kVp over time:	☐
Acquisition mAs over time:	☐
Acquisition DAP over time:	☐
Time period:	Months
Default chart sorting:	Frequency

Fluoroscopy chart options

Study workload:	☐
Study frequency:	☐
DAP per study:	✓
Default chart sorting:	Frequency

Mammography chart options

Study workload:	☐
AGD vs. compressed thickness:	✓
kVp vs. compressed thickness:	☐
mAs vs. compressed thickness:	☐

Fig. 14: Figure 13: OpenREM chart options

6.4 Chart types - CT

- Bar chart of average DLP for each acquisition protocol (all systems combined)

- Bar chart of average DLP for each acquisition protocol (one series per system)
- Pie chart of the frequency of each acquisition protocol
- Pie chart showing the number of studies carried on each day of the week
- Line chart showing the average DLP of each study name over time
- Bar chart of average $\mathrm{CTDI}_{\mathrm{vol}}$ for each acquisition protocol
- Bar chart of average DLP for each study name
- Pie chart of the frequency of each study name
- Bar chart of average DLP for each requested procedure
- Pie chart of the frequency of each requested procedure

6.5 Chart types - radiography

- Bar chart of average DAP for each acquisition protocol
- Pie chart of the frequency of each acquisition protocol
- Bar chart of average DAP for each study description
- Pie chart of the frequency of each study description
- Bar chart of average number of irradiation events for each study description
- Bar chart of average DAP for each requested procedure
- Pie chart of the frequency of each requested procedure
- Bar chart of average number of irradiation events for each requested procedure
- Bar chart of average kVp for each acquisition protocol
- Bar chart of average mAs for each acquisition protocol
- Pie chart showing the number of studies carried out per weekday
- Line chart of average DAP of each acquisition protocol over time
- Line chart of average mAs of each acquisition protocol over time
- Line chart of average kVp of each acquisition protocol over time

6.6 Chart types - fluoroscopy

- Bar chart of average DAP for each study description
- Pie chart of the frequency of each study description
- Bar chart of average DAP for each requested procedure
- Pie chart of the frequency of each requested procedure
- Pie chart showing the number of studies carried out per weekday

6.7 Chart types - mammography

- Scatter plot of average glandular dose vs. compressed thickness for each acquisition
- Scatter plot of kVp vs. compressed thickness for each acquisition
- Scatter plot of mAs vs. compressed thickness for each acquisition
- Pie chart showing the number of studies carried out per weekday

6.8 Performance notes

6.8.1 All chart types

For any study- or request-based charts, filtering using *Acquisition protocol* forces OpenREM to use a slightly slower method of querying the database for chart data.

6.8.2 Bar charts

Switching off histogram calculation in *Chart options* will speed up bar chart data calculation.

Switching off *Plot a series per system* in the *Chart options* will speed up data calculation.

6.8.3 Scatter plots

Switching off *Plot a series per system* in the *Chart options* will speed up data calculation.

CHAPTER 7

Skin dose maps

7.1 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, so calculating on import may tie up your server for that time, depending on how you have configured the *Celery task queue*
- 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
- 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).

7.1.1 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

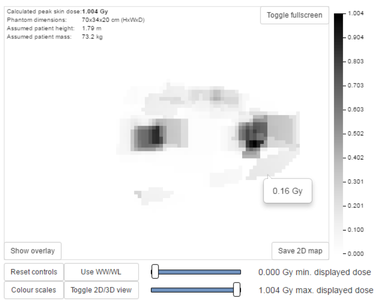

Calculated peak skin dose: **1.004 Gy**
Phantom dimensions: 70x34x20 cm (HxWxD)
Assumed patient height: 1.79 m
Assumed patient mass: 73.2 kg
0.16 Gy
0.000 Gy min. displayed dose
1.004 Gy max. displayed dose

Fig. 1: Figure 1: 2D visualisation of the 3D data

7.1.2 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

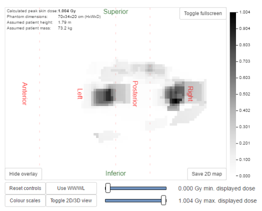

Calculated peak skin dose: **1.004 Gy**
Phantom dimensions: 70x34x20 cm (HxWxD)
Assumed patient height: 1.79 m
Assumed patient mass: 73.2 kg
0.000 Gy min. displayed dose
1.004 Gy max. displayed dose

Fig. 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

7.2 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

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.

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

7.3 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).

7.4 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`

7.4.1 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
```

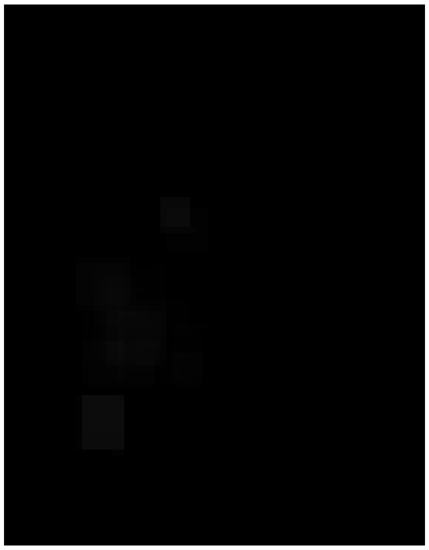
The peak dose is the peak incident dose delivered to any one-cm-square area. If any of these 1 cm^2 areas (referred to as cells) are above 3 Gy, then the number of cells in this category, or the two higher dose categories, are listed in the table accordingly.

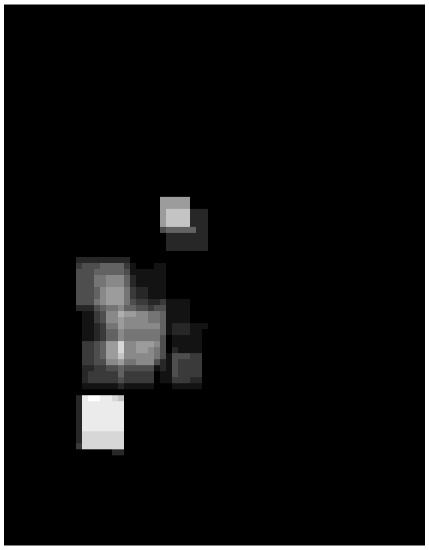
7.4.2 Incidence map image file

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:





7.5 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.

openSkin is yet to be validated independently - if this is something you want to do, please do go ahead and feed back your findings to Jonathan Cole at jacole.

CHAPTER 8

Exporting study information

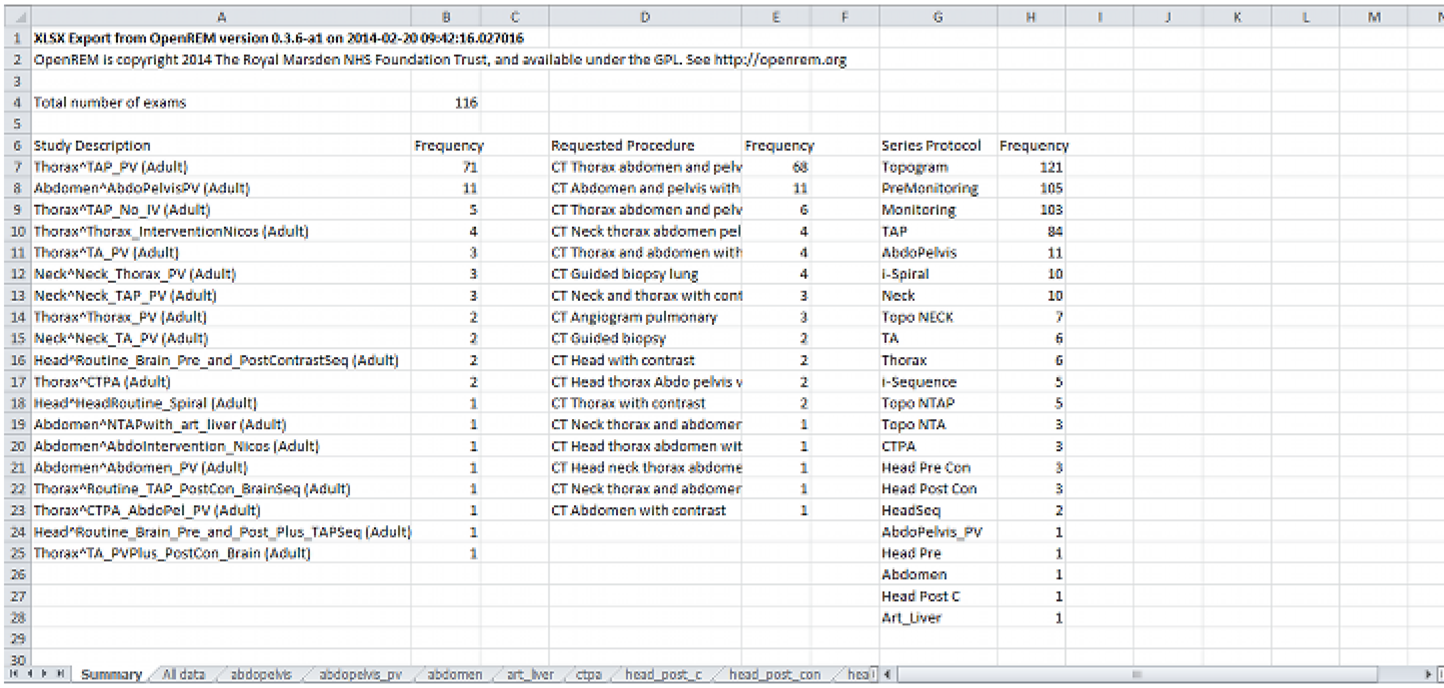
8.1 Exporting to csv and xlsx sheets

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:

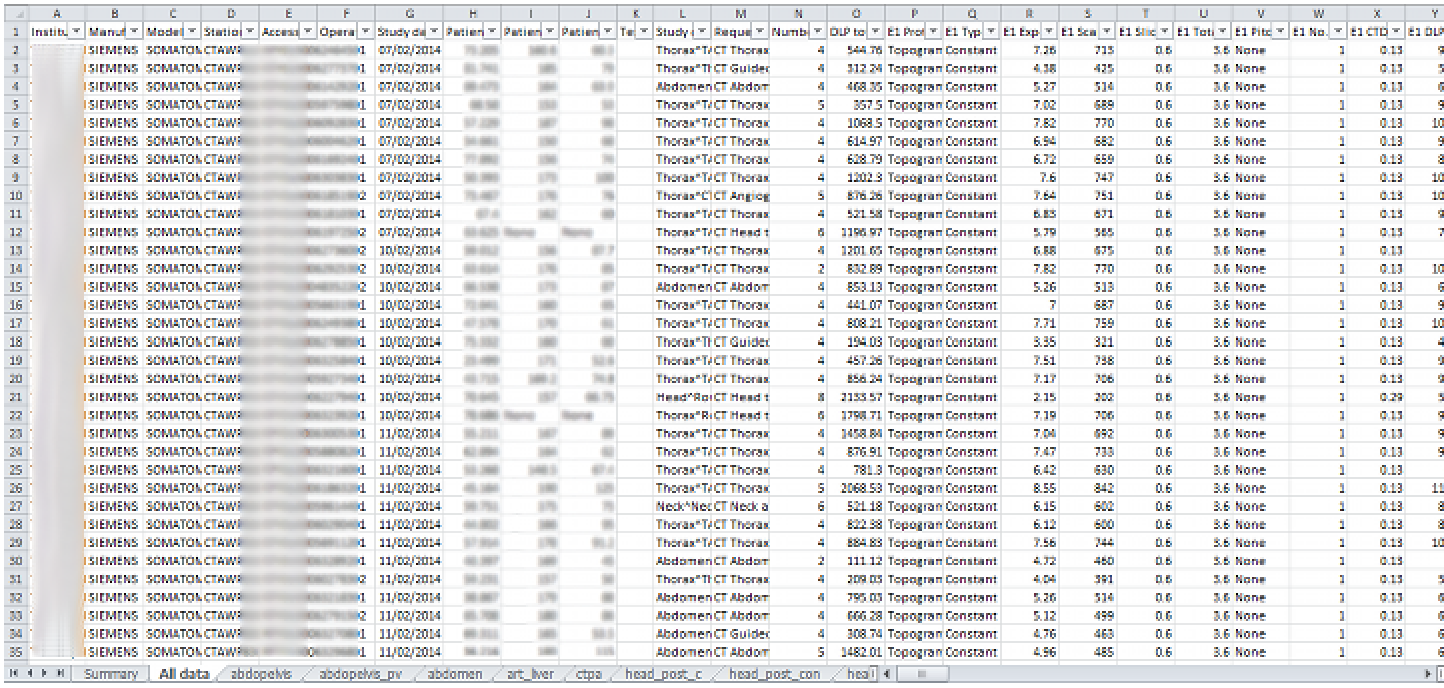


	A	B	C	D	E	F	G	H
1	XLSX Export from OpenREM version 0.3.6-a1 on 2014-02-20 09:42:16.027016							
2	OpenREM is copyright 2014 The Royal Marsden NHS Foundation Trust, and available under the GPL. See http://openrem.org							
3								
4	Total number of exams	116						
5								
6	Study Description	Frequency		Requested Procedure	Frequency		Series Protocol	Frequency
7	Thorax^TAP_PV (Adult)	71		CT Thorax abdomen and pelv	68		Topogram	121
8	Abdomen^AbdoPelvisPV (Adult)	11		CT Abdomen and pelvis with	11		PreMonitoring	105
9	Thorax^TAP_No_IV (Adult)	5		CT Thorax abdomen and pelv	6		Monitoring	103
10	Thorax^Thorax_InterventionNicos (Adult)	4		CT Neck thorax abdomen pel	4		TAP	84
11	Thorax^TA_PV (Adult)	3		CT Thorax and abdomen with	4		AbdoPelvis	11
12	Neck^Neck_Thorax_PV (Adult)	3		CT Guided biopsy lung	4		i-Spiral	10
13	Neck^Neck_TAP_PV (Adult)	3		CT Neck and thorax with cont	3		Neck	10
14	Thorax^Thorax_PV (Adult)	2		CT Angiogram pulmonary	3		Topo NECK	7
15	Neck^Neck_TA_PV (Adult)	2		CT Guided biopsy	2		TA	6
16	Head^Routine_Brain_Pre_and_PostContrastSeq (Adult)	2		CT Head with contrast	2		Thorax	6
17	Thorax^CTPA (Adult)	2		CT Head thorax Abdo pelvis v	2		i-Sequence	5
18	Head^HeadRoutine_Spiral (Adult)	1		CT Thorax with contrast	2		Topo NTAP	5
19	Abdomen^NTAPwith_art_liver (Adult)	1		CT Neck thorax and abdomer	1		Topo NTA	3
20	Abdomen^AbdoIntervention_Nicos (Adult)	1		CT Head thorax abdomen wit	1		CTPA	3
21	Abdomen^Abdomen_PV (Adult)	1		CT Head neck thorax abdome	1		Head Pre Con	3
22	Thorax^Routine_TAP_PostCon_BrainSeq (Adult)	1		CT Neck thorax and abdomer	1		Head Post Con	3
23	Thorax^CTPA_AbdoPel_PV (Adult)	1		CT Abdomen with contrast	1		HeadSeq	2
24	Head^Routine_Brain_Pre_and_Post_Plus_TAPSeq (Adult)	1					AbdoPelvis_PV	1
25	Thorax^TA_PVPlus_PostCon_Brain (Adult)	1					Head Pre	1
26							Abdomen	1
27							Head Post C	1
28							Art_Liver	1
29								
30								

Summary / All data / abdopelvis / abdopelvis_pv / abdomen / art_liver / ctpa / head_post_c / head_post_con / hea

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.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	Institu	Manuf	Model	Statio	Access	Opera	Study da	Patien	Patien	Patien	Te	Study	Reque	Numb	DLP to	E1 Prot	E1 Typ	E1 Exp	E1 Sca	E1 Slic	E1 Tot	E1 Pitc	E1 No.	E1 CTD	E1 DLP
2	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	07/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Thorax	4	544.76	Topogram	Constant	7.26	715	0.6	3.6	None	1	0.13	9
3	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	07/02/2014	[redacted]	[redacted]	[redacted]		Thorax^Th	CT Guidec	4	312.24	Topogram	Constant	4.38	425	0.6	3.6	None	1	0.13	5
4	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	07/02/2014	[redacted]	[redacted]	[redacted]		Abdomen	CT Abdom	4	468.35	Topogram	Constant	5.27	514	0.6	3.6	None	1	0.13	6
5	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	07/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Thorax	5	357.5	Topogram	Constant	7.02	689	0.6	3.6	None	1	0.13	9
6	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	07/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Thorax	4	1068.5	Topogram	Constant	7.82	770	0.6	3.6	None	1	0.13	10
7	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	07/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Thorax	4	614.97	Topogram	Constant	6.94	682	0.6	3.6	None	1	0.13	9
8	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	07/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Thorax	4	628.79	Topogram	Constant	6.72	659	0.6	3.6	None	1	0.13	8
9	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	07/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Thorax	4	1202.3	Topogram	Constant	7.6	747	0.6	3.6	None	1	0.13	10
10	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]2	07/02/2014	[redacted]	[redacted]	[redacted]		Thorax^C1	CT Angiog	5	876.26	Topogram	Constant	7.64	751	0.6	3.6	None	1	0.13	10
11	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	07/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Thorax	4	521.58	Topogram	Constant	6.83	671	0.6	3.6	None	1	0.13	9
12	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]2	07/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Head t	6	1196.97	Topogram	Constant	5.79	565	0.6	3.6	None	1	0.13	7
13	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]2	10/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Thorax	4	1201.65	Topogram	Constant	6.88	675	0.6	3.6	None	1	0.13	
14	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]2	10/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Thorax	2	832.89	Topogram	Constant	7.82	770	0.6	3.6	None	1	0.13	10
15	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]2	10/02/2014	[redacted]	[redacted]	[redacted]		Abdomen	CT Abdom	4	853.13	Topogram	Constant	5.26	513	0.6	3.6	None	1	0.13	6
16	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	10/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Thorax	4	441.07	Topogram	Constant	7	687	0.6	3.6	None	1	0.13	9
17	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	10/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Thorax	4	808.21	Topogram	Constant	7.71	759	0.6	3.6	None	1	0.13	10
18	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	10/02/2014	[redacted]	[redacted]	[redacted]		Thorax^Th	CT Guidec	4	194.03	Topogram	Constant	3.35	321	0.6	3.6	None	1	0.13	4
19	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	10/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Thorax	4	457.26	Topogram	Constant	7.51	738	0.6	3.6	None	1	0.13	9
20	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	10/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Thorax	4	856.24	Topogram	Constant	7.17	706	0.6	3.6	None	1	0.13	9
21	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	10/02/2014	[redacted]	[redacted]	[redacted]		Head^Ro	CT Head t	8	2133.57	Topogram	Constant	2.15	202	0.6	3.6	None	1	0.29	5
22	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	10/02/2014	[redacted]	[redacted]	[redacted]		Thorax^R	CT Head t	6	1798.71	Topogram	Constant	7.19	706	0.6	3.6	None	1	0.13	9
23	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	11/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Thorax	4	1458.84	Topogram	Constant	7.04	692	0.6	3.6	None	1	0.13	9
24	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	11/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Thorax	4	876.91	Topogram	Constant	7.47	733	0.6	3.6	None	1	0.13	9
25	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	11/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Thorax	4	781.3	Topogram	Constant	6.42	630	0.6	3.6	None	1	0.13	
26	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	11/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Thorax	5	2068.53	Topogram	Constant	8.55	842	0.6	3.6	None	1	0.13	11
27	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	11/02/2014	[redacted]	[redacted]	[redacted]		Neck^Nec	CT Neck a	6	521.18	Topogram	Constant	6.15	602	0.6	3.6	None	1	0.13	8
28	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	11/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Thorax	4	822.38	Topogram	Constant	6.12	600	0.6	3.6	None	1	0.13	8
29	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	11/02/2014	[redacted]	[redacted]	[redacted]		Thorax^T/	CT Thorax	4	884.83	Topogram	Constant	7.56	744	0.6	3.6	None	1	0.13	10
30	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	11/02/2014	[redacted]	[redacted]	[redacted]		Abdomen	CT Abdom	2	111.12	Topogram	Constant	4.72	460	0.6	3.6	None	1	0.13	
31	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]2	11/02/2014	[redacted]	[redacted]	[redacted]		Thorax^Th	CT Thorax	4	209.03	Topogram	Constant	4.04	391	0.6	3.6	None	1	0.13	5
32	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	11/02/2014	[redacted]	[redacted]	[redacted]		Abdomen	CT Abdom	4	795.03	Topogram	Constant	5.26	514	0.6	3.6	None	1	0.13	6
33	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]2	11/02/2014	[redacted]	[redacted]	[redacted]		Abdomen	CT Abdom	4	666.28	Topogram	Constant	5.12	499	0.6	3.6	None	1	0.13	6
34	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	11/02/2014	[redacted]	[redacted]	[redacted]		Abdomen	CT Guidec	4	308.74	Topogram	Constant	4.76	463	0.6	3.6	None	1	0.13	6
35	[redacted]	SIEMENS	SOMATOM	CTAW[redacted]	[redacted]	[redacted]1	11/02/2014	[redacted]	[redacted]	[redacted]		Abdomen	CT Abdom	5	1482.01	Topogram	Constant	4.96	485	0.6	3.6	None	1	0.13	6

Summary / All data / abdopelvis / abdopelvis_pv / abdomen / art_liver / ctpa / head_post_c / head_post_con / hea

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.

8.1.1 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)

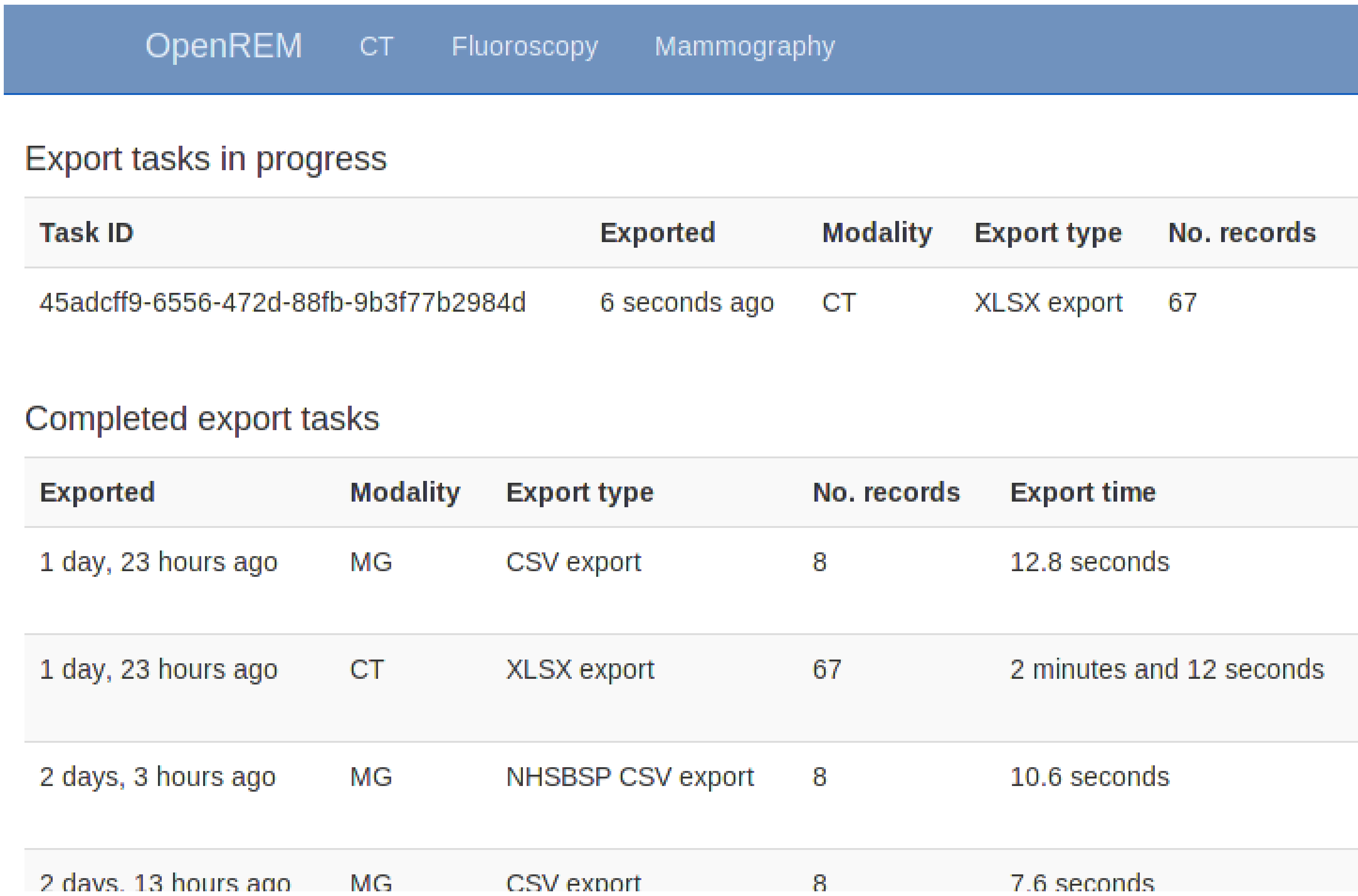
The minimum, maximum and mean of all the remaining factors are presented for each group along with the common factors. Where a factor is not available in the source RDSR, that factor is not considered.

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.

8.1.2 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:



Export tasks in progress

Task ID	Exported	Modality	Export type	No. records
45adcff9-6556-472d-88fb-9b3f77b2984d	6 seconds ago	CT	XLSX export	67

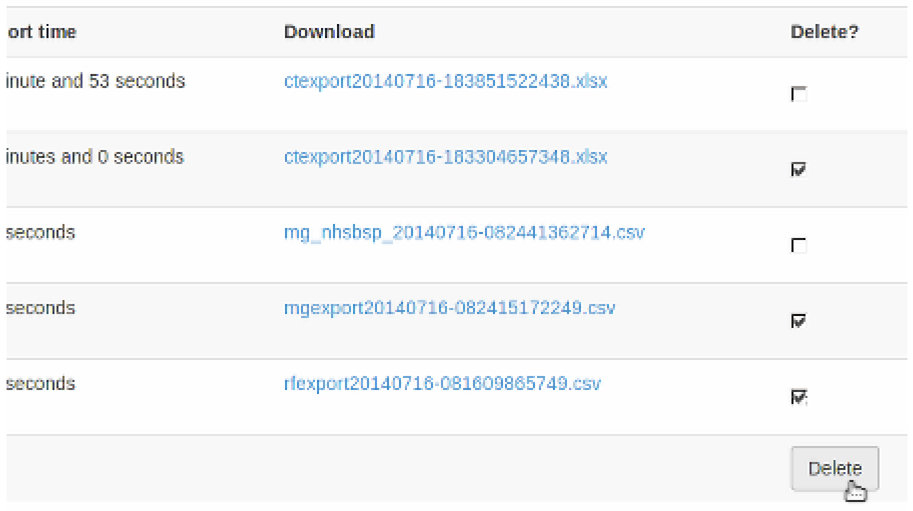
Completed export tasks

Exported	Modality	Export type	No. records	Export time
1 day, 23 hours ago	MG	CSV export	8	12.8 seconds
1 day, 23 hours ago	CT	XLSX export	67	2 minutes and 12 seconds
2 days, 3 hours ago	MG	NHSBSP CSV export	8	10.6 seconds
2 days, 13 hours ago	MG	CSV export	8	7.6 seconds

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:



ort time	Download	Delete?
inute and 53 seconds	ctexport20140716-183851522438.xlsx	☐
inutes and 0 seconds	ctexport20140716-183304657348.xlsx	☑
seconds	mg_nhsbsp_20140716-082441362714.csv	☐
seconds	mgexport20140716-082415172249.csv	☑
seconds	rfexport20140716-081609865749.csv	☑
		Delete

8.2 Specific modality export information

8.2.1 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.

8.2.2 PHE 2019 CT survey

This export is specific to the UK Public Health England (PHE) CT dose audit 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 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 minimum age of 16, and by study description (or combination of factors to specify a particular protocol). The guidance specifies patients between 50 to 90 kg – if you have weight data in OpenREM this filter could be added, but the values form part of the output anyway so it isn't essential. It would however enable filtering out of those studies that don't have weight data if the majority does. If patient weight data isn't available in OpenREM then just ensure the sample is large!

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 onwards of the Patient and Protocol sheet of the PHE data collection spreadsheet. Column AL 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 onwards.

8.3 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.

CHAPTER 9

Troubleshooting

9.1 Server 500 errors

Turn on debug mode

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`
- Change the line:

```
# DEBUG = True
```

- to:

```
DEBUG = True
```

This will render a debug report in the browser - usually revealing the problem.

Once the problem is fixed, change `DEBUG` to `False`, or comment it again using a `#`. If you leave debug mode in place, the system is likely to run out of memory as database queries are cached.

9.2 Unknown encoding errors

9.2.1 Background

OpenREM versions 0.8 and earlier rely on pydicom 0.9.9 which has recently had a new release, version 1.0. A replacement of pynetdicom is also being worked on and will be called pynetdicom3. When pynetdicom3 is released, OpenREM will be revised to work the two new packages, but as they are both backwards-incompatible with the old versions you must keep using the versions specified by the installation instructions until this happens. (The correct version of pydicom is installed automatically when you install OpenREM).

The function that deals with character set encoding in pydicom 0.9.9 has not been edited since 2013 and is missing a lot of encodings which can cause imports and query-retrieve functions to fail if those encodings are encountered.

As an interim work-around, users that encounter this issue can take the following step to work with those files in OpenREM 0.8.

9.2.2 Work-around

Find and edit the pydicom file `charset.py`. It should be at one of the following paths:

- Ubuntu linux: `/usr/local/lib/python2.7/dist-packages/dicom/charset.py`
- Other linux: `/usr/lib/python2.7/site-packages/dicom/charset.py`
- Linux virtualenv: `vitualenvfolder/lib/python2.7/site-packages/dicom/charset.py`
- Windows: `C:\Python27\Lib\site-packages\dicom\charset.py`
- Windows virtualenv: `virtualenvfolder\Lib\site-packages\dicom\charset.py`

Replace the dictionary `python_encoding` that is declared at the start with the following dictionary:

```
# Map DICOM Specific Character Set to python equivalent
python_encoding = {

    # default character set for DICOM
    '': 'iso8859',

    # alias for latin_1 too (iso_ir_6 exists as an alias to 'ascii')
    'ISO_IR 6': 'iso8859',
    'ISO_IR 13': 'shift_jis',

    # these also have iso_ir_1XX aliases in python 2.7
    'ISO_IR 100': 'latin_1',
    'ISO_IR 101': 'iso8859_2',
    'ISO_IR 109': 'iso8859_3',
    'ISO_IR 110': 'iso8859_4',
    'ISO_IR 126': 'iso_ir_126',  # Greek
    'ISO_IR 127': 'iso_ir_127',  # Arabic
    'ISO_IR 138': 'iso_ir_138',  # Hebrew
    'ISO_IR 144': 'iso_ir_144',  # Russian
    'ISO_IR 148': 'iso_ir_148',  # Turkish
    'ISO_IR 166': 'iso_ir_166',  # Thai
    'ISO 2022 IR 6': 'iso8859',  # alias for latin_1 too
```

(continues on next page)

(continued from previous page)

```
    'ISO 2022 IR 13': 'shift_jis',
    'ISO 2022 IR 87': 'iso2022_jp',
    'ISO 2022 IR 100': 'latin_1',
    'ISO 2022 IR 101': 'iso8859_2',
    'ISO 2022 IR 109': 'iso8859_3',
    'ISO 2022 IR 110': 'iso8859_4',
    'ISO 2022 IR 126': 'iso_ir_126',
    'ISO 2022 IR 127': 'iso_ir_127',
    'ISO 2022 IR 138': 'iso_ir_138',
    'ISO 2022 IR 144': 'iso_ir_144',
    'ISO 2022 IR 148': 'iso_ir_148',
    'ISO 2022 IR 149': 'euc_kr',  # needs cleanup via clean_escseq()
    'ISO 2022 IR 159': 'iso-2022-jp',
    'ISO 2022 IR 166': 'iso_ir_166',
    'ISO 2022 IR 58': 'iso_ir_58',
    'ISO_IR 192': 'UTF8',  # from Chinese example, 2008 PS3.5 Annex J p1-4
    'GB18030': 'GB18030',
    'ISO 2022 GBK': 'GBK',  # from DICOM correction CP1234
    'ISO 2022 58': 'GB2312',  # from DICOM correction CP1234
    'GBK': 'GBK',  # from DICOM correction CP1234
}
```

Leave the rest of the file as it was and save. You should now be able to work with DICOM encoded with a much wider range of character sets.

9.3 Invalid tag errors

9.3.1 Background

OpenREM versions 0.8 and earlier rely on pydicom 0.9.9 which has recently had a new release, version 1.0. A replacement of pynetdicom is also being worked on and will be called pynetdicom3. When pynetdicom3 is released, OpenREM will be revised to work the two new packages, but as they are both backwards-incompatible with the old versions you must keep using the versions specified by the installation instructions until this happens. (The correct version of pydicom is installed automatically when you install OpenREM).

The DICOM dictionary (the list of all the tags) supplied with pydicom 0.9.9 is old and doesn't have some of the newer tags that some modalities are starting to include. This can lead to errors when you attempt to import a file with one of these tags, and the file will not be imported.

9.3.2 Error message

The error message will look something like this. The actual tag can be different, this is an example.

Error: KeyError: 'Invalid tag (0018, 9559): "Unknown DICOM tag (0018, 9559) - can't look up VR"'

9.3.3 Work-around

The workaround is to replace the `_dicom_dict.py` with the version in the current master of pydicom.

First, download the current file (right click 'Save link as…' or similar): _dicom_dict.py

Next, find the location of your pydicom install – it should be at one of the following paths:

- Ubuntu linux: `/usr/local/lib/python2.7/dist-packages/dicom/`
- Other linux: `/usr/lib/python2.7/site-packages/dicom/`
- Linux virtualenv: `vitualenvfolder/lib/python2.7/site-packages/dicom/`
- Windows: `C:\Python27\Lib\site-packages\dicom\`
- Windows virtualenv: `virtualenvfolder\Lib\site-packages\dicom\`

Replace the existing `_dicom_dict.py` file with the one you have downloaded.

Try and import the file again - it should now work.

9.4 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. As a result the accumulated AGD code didn't work, so the accumulated AGD cell on the mammography summary sheets remained blank.

The code below allows the laterality to be derived automatically from the acquisition protocol name, if the protocol name has `R` or `L` as the first letter. When this information is derived, the accumulated AGD is also calculated per breast.

9.4.1 Back up your database

It is always best practice to backup your database before running code to edit your database:

- For PostgreSQL you can refer to *Backup the database*
- For a non-production SQLite3 database, simply make a copy of the database file

9.4.2 How to use the code

Create a new file in your Python OpenREM folder (the folder `manage.py` is in). For example it could be called `fix_dbt_laterality.py`:

- Ubuntu linux: `/usr/local/lib/python2.7/dist-packages/openrem/fix_dbt_laterality.py`
- Other linux: `/usr/lib/python2.7/site-packages/openrem/fix_dbt_laterality.py`
- Linux virtualenv: `vitualenvfolder/lib/python2.7/site-packages/openrem/fix_dbt_laterality.py`
- Windows: `C:\Python27\Lib\site-packages\openrem\fix_dbt_laterality.py`
- Windows virtualenv: `virtualenvfolder\Lib\site-packages\openrem\fix_dbt_laterality.py`

Copy and paste into the new file the code from below. You will need to edit the `DISPLAY_NAME` to match the display name you have configured for the Hologic DBT system that needs to be modified. If you are not sure what this is, go to the home page of your OpenREM installation and see how the Hologic unit is listed there. You will not be able to copy and paste from there as if you click on it it will load that page, and likewise on the summary list page. If you click through to the study detail page, you will find the display name listed in the details there. See the *Display names and user-defined modalities* documentation for more information.

If you are working on Linux, you may like to look at the brief tips on using `nano` on the Nginx *Troubleshooting and tips* section.

```
# -*- coding: utf-8 -*-
# Routine to fix missing laterality details for exposures extracted from Hologic DBT␣
↪proprietary projection data objects
# Also retrospectively updates the accumulated average glandular dose fields that␣
↪were previously missing due to lack
# of laterality.
#
# Contains hard coded DISPLAY_NAME that must be changed appropriately to identify the␣
↪Hologic DBT system in your
# database.
# See https://bitbucket.org/openrem/openrem/issues/411

import os
import sys
import django
import logging
from django.core.exceptions import ObjectDoesNotExist

logger = logging.getLogger(__name__)

# setup django/OpenREM
basepath = os.path.dirname(__file__)
projectpath = os.path.abspath(os.path.join(basepath, ))
if projectpath not in sys.path:
    sys.path.insert(1, projectpath)
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'openremproject.settings')
django.setup()

from remapp.models import IrradEventXRayData
from remapp.tools.get_values import get_or_create_cid

# *****************************************************
# Update the DISPLAY_NAME
DISPLAY_NAME = "Display name of my Hologic"
# *****************************************************

def _accumulatedxraydose(proj):
    """
    Borrowed from mam.py
    Creates an AccumXRayDose table entry
    :param proj: Projection x-ray radiation dose database entry
    :return: Nothing
    """
    from remapp.models import AccumXRayDose, AccumMammographyXRayDose
    from remapp.tools.get_values import get_or_create_cid
    accum = AccumXRayDose.objects.create(projection_xray_radiation_dose=proj)
    accum.acquisition_plane = get_or_create_cid('113622', 'Single Plane')
    accum.save()
    accummam = AccumMammographyXRayDose.objects.create(accumulated_xray_dose=accum)
    accummam.accumulated_average_glandular_dose = 0.0
    accummam.save()
```

(continues on next page)

(continued from previous page)

```
def _accumulatedmammo_update(event):  # TID 10005
    """
    Borrowed from mam.py
    Updates the accumulated average glandular dose tables on a per-breast basis
    :param event: Irradiation event database entry
    :return: Nothing
    """
    from remapp.tools.get_values import get_or_create_cid
    from remapp.models import AccumMammographyXRayDose
    try:
        accum = event.projection_xray_radiation_dose.accumxraydose_set.get()
    except ObjectDoesNotExist:
        print("No accumxraydose for event occurring at {0} in study no {1}".
↪format(event.date_time_started,
            event.projection_xray_radiation_dose.general_study_module_attributes.id))
        _accumulatedxraydose(event.projection_xray_radiation_dose)
        accum = event.projection_xray_radiation_dose.accumxraydose_set.get()
    accummams = accum.accummammographyxraydose_set.all()
    event_added = False
    for accummam in accummams:
        if not accummam.laterality:
            if event.laterality.code_meaning == 'Right':
                accummam.laterality = get_or_create_cid('T-04020', 'Right breast')
            elif event.laterality.code_meaning == 'Left':
                accummam.laterality = get_or_create_cid('T-04030', 'Left breast')
            accummam.accumulated_average_glandular_dose += event.
↪irradeventxraysourcedata_set.get(
                ).average_glandular_dose
            accummam.save()
            event_added = True
        elif event.laterality.code_meaning in accummam.laterality.code_meaning:
            accummam.accumulated_average_glandular_dose += event.
↪irradeventxraysourcedata_set.get(
                ).average_glandular_dose
            accummam.save()
            event_added = True
    if not event_added:
        accummam = AccumMammographyXRayDose.objects.create(accumulated_xray_
↪dose=accum)
        if event.laterality.code_meaning == 'Right':
            accummam.laterality = get_or_create_cid('T-04020', 'Right breast')
        elif event.laterality.code_meaning == 'Left':
            accummam.laterality = get_or_create_cid('T-04030', 'Left breast')
        accummam.accumulated_average_glandular_dose = event.irradeventxraysourcedata_
↪set.get().average_glandular_dose
        accummam.save()
    accummam.save()


events = IrradEventXRayData.objects.filter(
    projection_xray_radiation_dose__general_study_module_attributes__
↪generalequipmentmoduleattr__unique_equipment_name__display_name__exact=DISPLAY_NAME)

events_r = events.filter(laterality__code_meaning__exact=u"Right")
events_l = events.filter(laterality__code_meaning__exact=u"Left")
events_n = events.filter(laterality__isnull=True)
```

(continues on next page)

(continued from previous page)

```
print(u"Total events is {0}, of which {1} are Right, {2} are Left and {3} are null␣
↪(remainder {4})".format(
    events.count(), events_r.count(), events_l.count(), events_n.count(),
    events.count() - events_r.count() - events_l.count() - events_n.count()))

for event in events_n:
    if event.acquisition_protocol[0] == u'R':
        event.laterality = get_or_create_cid('G-A100', 'Right')
        event.save()
        _accumulatedmammo_update(event)
    elif event.acquisition_protocol[0] == u'L':
        event.laterality = get_or_create_cid('G-A101', 'Left')
        event.save()
        _accumulatedmammo_update(event)
    else:
        print("Event acquisition protocol is {0} so we couldn't assign it left or␣
↪right. Exam ID is {1}".format(
            event.acquisition_protocol, event.projection_xray_radiation_dose.general_
↪study_module_attributes.id))

events_r = events.filter(laterality__code_meaning__exact=u"Right")
events_l = events.filter(laterality__code_meaning__exact=u"Left")
events_n = events.filter(laterality__isnull=True)
print(u"Post update, total events is {0}, of which {1} are Right, {2} are Left and {3}
↪ are null (remainder {4})".format(
    events.count(), events_r.count(), events_l.count(), events_n.count(),
    events.count() - events_r.count() - events_l.count() - events_n.count()))
```

9.4.3 Run the fix

In a shell/command window, activate your virtualenv if you are using one, and change directory to 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\`

Then:

```
python fix_dbt_laterality.py
```

This should generate the following response, with one message for each event that can’t be assigned laterality due to the acquisition protocol name not starting with `L` or `R`:

```
Total events is 46410, of which 0 are Right, 0 are Left and 46410 are null (remainder␣
↪0)
Event acquisition protocol is Flat Field Combo so we couldn't assign it left or right.
↪ Exam ID is 184466
Event acquisition protocol is Flat Field Combo so we couldn't assign it left or right.
↪ Exam ID is 75963
Event acquisition protocol is Flat Field Combo so we couldn't assign it left or right.
↪ Exam ID is 75919
```

(continues on next page)

(continued from previous page)

```
Event acquisition protocol is Flat Field Combo so we couldn't assign it left or right.
↪ Exam ID is 76004
Event acquisition protocol is Flat Field Combo so we couldn't assign it left or right.
↪ Exam ID is 83784
Event acquisition protocol is Flat Field Combo so we couldn't assign it left or right.
↪ Exam ID is 83784
Event acquisition protocol is Flat Field Combo so we couldn't assign it left or right.
↪ Exam ID is 84912
Event acquisition protocol is Flat Field Combo so we couldn't assign it left or right.
↪ Exam ID is 100765
Event acquisition protocol is Flat Field Combo so we couldn't assign it left or right.
↪ Exam ID is 110471
Event acquisition protocol is Flat Field Combo so we couldn't assign it left or right.
↪ Exam ID is 121500
Event acquisition protocol is Flat Field Combo so we couldn't assign it left or right.
↪ Exam ID is 121588
Event acquisition protocol is Flat Field Combo so we couldn't assign it left or right.
↪ Exam ID is 123462
Event acquisition protocol is Flat Field Combo so we couldn't assign it left or right.
↪ Exam ID is 137145
Event acquisition protocol is ACR Phantom Combo so we couldn't assign it left or␣
↪right. Exam ID is 140563
Event acquisition protocol is Flat Field Tomo so we couldn't assign it left or right.␣
↪Exam ID is 156826
Event acquisition protocol is Flat Field Combo so we couldn't assign it left or right.
↪ Exam ID is 165131
Event acquisition protocol is Flat Field Combo so we couldn't assign it left or right.
↪ Exam ID is 165486
Post update, total events is 46410, of which 23323 are Right, 23070 are Left and 17␣
↪are null (remainder 0)
```

The Exam ID referred to is the database ID, so if you look at a mammography exam in the web interface, you can change the Exam ID in the URL if you want to review that study.

9.4.4 Multiple Hologic units

If you have more than one unit that has studies that need fixing, simply change the `DISPLAY_NAME` and run the code again.

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 RabbitMQ/task management:

- *Task management*

9.5 Log files

Log file location, naming and verbosity were configured in the `local_settings.py` configuration - see the *Log file* configuration docs for details.

If the defaults have not been modified, then there will be three log files in your `MEDIAROOT` folder which you configured at installation. See the install config section on *Location for imports and exports* 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`.

9.6 Starting again!

If for any reason you want to start again with the database, then this is how you might do it:

9.6.1 SLQite3 database

- Delete or rename your existing database file (location will be described in your `local_settings.py` file)
- *Create the database*

9.6.2 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!
a.delete()
a.count()
exit()
```

CHAPTER 10

Developers

Contents:

10.1 Running the test suite

10.1.1 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 *Pre-installation preparations* 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.

Configure OpenREM

Rename and configure `openremproject/local_settings.py.example` and `openremproject/wsgi.py.example` as per the *Installing OpenREM* docs.

Create a database following the same *Installing OpenREM* instructions.

10.1.2 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.

10.1.3 Related tools

Enabling django-debug-toolbar

See *Enabling debug toolbar*

10.1.4 Creating test versions of production systems

If you wish to create a duplicate install to test upgrades etc, refer to *Restore the database* and the preceding text regarding making backups.

10.2 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

10.2.1 Installation

- Activate the virtualenv (assuming you are using one…)
- Install from pip:

```
pip install django-debug-toolbar==1.9.1
```

The version is fixed for now due to the version of Django being used.

10.2.2 Configuration

- Open `openremproject/local_settings.py` and add the lines:

```
MIDDLEWARE_CLASSES += ('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.

10.2.3 Using Django Debug Toolbar

When `DEBUG = True` in `openremproject/local_settings.py` the toolbar should appear.

10.3 DICOM import modules

10.3.1 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 and digital radiography.

10.3.2 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.

10.3.3 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.

10.3.4 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.

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.

10.4 Non-DICOM import modules

10.4.1 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.

10.5 Export from database

10.5.1 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.

`(task)` `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.

Parameters

- **`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

Returns Saves xlsx file into Media directory for user to download

`(task)` `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

Parameters

- **`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

Returns Saves xlsx file into Media directory for user to download

`(task)` `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.

Parameters

- **`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

Returns Saves xlsx file into Media directory for user to download

(task) `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.

Parameters

- **`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

Returns Saves csv file into Media directory for user to download

10.5.2 Single sheet CSV exports

(task) `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.

Parameters

- **`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

Returns Saves csv file into Media directory for user to download

(task) `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.

Parameters

- **`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

Returns Saves csv file into Media directory for user to download

(task) `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.

Parameters

- **`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

Returns Saves csv file into Media directory for user to download

(task) `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.

Parameters

- **`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

Returns Saves csv file into Media directory for user to download

Specialised csv exports - NHSBSP formatted mammography export

(task) `remapp.exports.mg_csv_nhsbsp.`**`mg_csv_nhsbsp`**(*filterdict*, *user=None*)

Export filtered mammography database data to a NHSBSP formatted single-sheet CSV file.

Parameters **`filterdict`** (*`dict`*) – Dictionary of query parameters from the mammo filtered page URL.

Returns None - file is saved to disk and location is stored in database

10.6 Tools and helper modules

10.6.1 OpenREM settings

Administrative module to define the name of the project and to add it to the Python path

10.6.2 Get values

Tiny modules to reduce repetition in the main code when extracting information from DICOM headers using pydicom.

`remapp.tools.get_values.`**`get_value_kw`**`(`*tag*, *dataset*`)`

Get DICOM value by keyword reference.

Parameters

- **`tag`** (*`str.`*) – DICOM keyword, no spaces or plural as per dictionary.
- **`dataset`** (*`dataset`*) – The DICOM dataset containing the tag.

Returns str. – value

`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.

Parameters

- **`tag`** (*`hex`*) – DICOM group and element number as a single hexadecimal number (prefix 0x).
- **`dataset`** (*`dataset`*) – The DICOM dataset containing the tag.

Returns str. – value

`remapp.tools.get_values.`**`get_seq_code_value`**`(`*sequence*, *dataset*`)`

From a DICOM sequence, get the code value.

Parameters

- **`sequence`** (*`DICOM keyword, no spaces or plural as per dictionary.`*) – DICOM sequence name.
- **`dataset`** (*`DICOM dataset`*) – The DICOM dataset containing the sequence.

Returns int. – code value

`remapp.tools.get_values.`**`get_seq_code_meaning`**`(`*sequence*, *dataset*`)`

From a DICOM sequence, get the code meaning.

Parameters

- **`sequence`** (*`DICOM keyword, no spaces or plural as per dictionary.`*) – DICOM sequence name.
- **`dataset`** (*`DICOM dataset`*) – The DICOM dataset containing the sequence.

Returns str. – code meaning

`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.

Parameters

- **`codevalue`** (*`int.`*) – Code value as defined in the DICOM standard part 16
- **`codemeaning`** – Code meaning as defined in the DICOM standard part 16

Returns ContextID entry for code value passed

`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

`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

`remapp.tools.get_values.`**`export_csv_prep`**(*unicode_string*)
Built-in csv module can't deal with unicode strings without specifying an encoding. Hence encode to utf-8 before writing. :param unicode_string: String to encode as utf-8 :return: UTF-8 encoded string with no commas or semicolons.

`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)

`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

10.6.3 Check if UID exists

Small module to check if UID already exists in the database.

`remapp.tools.check_uid.`**`check_uid`**(*uid*, *level='Study'*)
Check if UID already exists in database.

Parameters **`uid`** (`str.`) – Study UID.

Returns 1 if it does exist, 0 otherwise

`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.

Parameters

- **`study`** – GeneralStudyModuleAttr database object
- **`sop_instance_uid`** – SOP Instance UID of object being imported

Returns

10.6.4 DICOM time and date values

Module to convert betweeen DICOM and Python dates and times.

`remapp.tools.dcmdatetime.`**`get_date`**(*tag*, *dataset*)
Get DICOM date string and return Python date.

Parameters

- **`tag`** (`str.`) – DICOM keyword, no spaces or plural as per dictionary.
- **`dataset`** (`dataset`) – The DICOM dataset containing the tag.

Returns Python date value

`remapp.tools.dcmdatetime.`**`get_time`**(*tag*, *dataset*)
Get DICOM time string and return Python time.

Parameters

- **`tag`** (*`str.`*) – DICOM keyword, no spaces or plural as per dictionary.
- **`dataset`** (*`dataset`*) – The DICOM dataset containing the tag.

Returns python time value

`remapp.tools.dcmdatetime.`**`get_date_time`**(*tag*, *dataset*)
Get DICOM date time string and return Python date time.

Parameters

- **`tag`** (*`str.`*) – DICOM keyword, no spaces or plural as per dictionary.
- **`dataset`** (*`dataset`*) – The DICOM dataset containing the tag.

Returns Python date time value

`remapp.tools.dcmdatetime.`**`make_date`**(*dicomdate*)
Given a DICOM date, return a Python date.

Parameters **`dicomdate`** (*`str.`*) – DICOM style date.

Returns Python date value

`remapp.tools.dcmdatetime.`**`make_time`**(*dicomtime*)
Given a DICOM time, return a Python time.

Parameters **`dicomdate`** (*`str.`*) – DICOM style time.

Returns Python time value

`remapp.tools.dcmdatetime.`**`make_date_time`**(*dicomdatetime*)
Given a DICOM date time, return a Python date time.

Parameters **`dicomdate`** (*`str.`*) – DICOM style date time.

Returns Python date time value

`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

`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.

Parameters

- **`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

Returns DICOM formatted date range or single date

`remapp.tools.dcmdatetime.`**`make_dcm_time`**(*python_time*)
Return DICOM formatted time without seconds from python time

Parameters **`python_time`** – Python datetime.time object

Returns string, %H%M

`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)

Parameters

- **`time1`** – time, format 0123, 0000 if None
- **`time2`** – time, format 0123, 2359 if None

Returns time range, string, format 0123-1234

10.6.5 Test for QA or other non-patient related studies

`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.

Parameters **`dataset`** (*`dataset`*) – The DICOM dataset.

Returns str. – xml style string if any trigger values are found.

10.7 Models

10.8 Filtering code

10.9 Views

Will be documented in future release

10.10 Export Views

10.11 Forms

10.12 DICOM networking modules

10.12.1 Query-retrieve module

Query function

Move function

openrem_qr.py script

10.12.2 NetDICOM common functions

`openrem.remapp.netdicom.tools.`**`echoscu`**(*scp_pk=None*, *store_scp=False*, *qr_scp=False*, **args*, ***kwargs*)

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 :param args: :param kwargs: :return: 'AssocFail', Success or ?

`openrem.remapp.netdicom.tools.`**`create_ae`**(*aet*, *port=None*, *sop_scu=None*, *sop_scp=None*, *transfer_syntax=None*)

Function to create an application entity :param aet: string, AE Title :param sop_classes: list of supported SOP classes from netdicom.SOPclass to override default set :param transfer_syntax: list of supported transfer syntax from dicom.UID to override default set :return: application entity object ready to be started

10.13 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.

10.13.1 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`

10.13.2 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`

10.13.3 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']
```

(continues on next page)

(continued from previous page)

```
        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)
```

(continues on next page)

(continued from previous page)

```
    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}
```

(continues on next page)

(continued from previous page)

```
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")
```

(continues on next page)

(continued from previous page)

```
...
...

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,
                    }
```

10.13.4 Additions to `displaychartoptions.html`

A new div needs to be added for the fluoroscopy chart options:

```
<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>
```

10.13.5 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 %}
```

(continues on next page)

(continued from previous page)

```
{% 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 %}
```

10.13.6 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.

10.14 Indices and tables

- genindex
- modindex
- search

CHAPTER 11

Release Notes and Change Log

11.1 Version history change log

11.1.1 OpenREM version history

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 OpenREM 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 TEMPLATE_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)

Note:

- #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)

Note: #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)

Note: 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

11.2 Release notes and upgrade instructions

Each release comes with specific upgrade instructions, so please follow the links below for the appropriate version.

11.2.1 Version specific information

- *Upgrade to OpenREM 0.9.1* (this release)

Upgrade to OpenREM 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*

Upgrading an OpenREM server with no internet access

Follow the instructions found at *Upgrade an offline OpenREM installation*, before returning here to update the database and configuration.

Upgrading from version 0.8.x

Upgrade

- Back up your database
  - For PostgreSQL on linux you can refer to *Backup the database*

    - For PostgreSQL on Windows you can refer to *Backing up a PostgreSQL database (Windows)*
    - 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 the new version of OpenREM:

```
pip install openrem==0.9.0
```

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
```

Update static files

In the same shell/command window as you used above run the following command to clear the static files belonging to your previous OpenREM version and replace them with those belonging to the version you have just installed (assuming you are using a production web server…):

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

Enable the RabbitMQ management interface

To make use of the RabbitMQ queue display and purge control, the management interface needs to be enabled. To do so, follow the instructions at *Enable RabbitMQ queue management interface*.

Enable the Celery management interface, Flower

To make use of the Celery task management, Flower needs to be running. To do so, follow the instructions in *Celery task management: Flower*. For 'one-page Ubuntu' installs, add the Flower related config and create, register and start the systemd service files as described in *Celery and Flower*. If you need to change the default Flower port of 5555 then make sure you do so in `openremproject\local_settings.py` to add/modify the line `FLOWER_PORT = 5555` as well as when you start Flower.

Changes to Celery for Windows

For best performance and reliability when using Celery on Windows, it is recommended to do the following from the openrem folder (activate virtualenv if using one):

```
pip install celery==3.1.25
```

If your command for starting Celery specifies a pool option, for example `-P solo`, remove it so that Celery reverts to using the default `prefork` pool. This will enable multiple tasks to run concurrently and it will be possible to terminate tasks.

If you are a Windows user you may also wish to review *Daemonising Celery and Flower on Windows* as the example control batch files have been updated.

E-mail server settings

If you want selected OpenREM users to be automatically sent fluoroscopy high dose alerts then set the details of the e-mail server to be used in the *E-mail server settings* part of your `local_settings.py` file. 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`

Then change the e-mail section settings to reflect the e-mail server that is to be used:

```
EMAIL_HOST = 'localhost'
EMAIL_PORT = 25
EMAIL_HOST_USER = ''
EMAIL_HOST_PASSWORD = ''
EMAIL_USE_TLS = False
EMAIL_USE_SSL = False
EMAIL_DOSE_ALERT_SENDER = 'your.alert@email.address'
EMAIL_OPENREM_URL = 'http://your.openrem.server'
```

See the *E-mail configuration* documentation for full details.

Restart all the services

Follow the guide at *Start all the services*.

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

If upgrading from 0.7.4, see also *OpenREM Release Notes version 0.8.0*

Upgrading an OpenREM server with no internet access

Follow the instructions found at *Upgrade an offline OpenREM installation*, before returning here to update the database and configuration.

Upgrading from version 0.7.4 or 0.8.0

Upgrade

- Back up your database
  - For PostgreSQL on linux you can refer to *Backup the database*
  - For PostgreSQL on Windows you can refer to backupRestorePostgreSQL
  - 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 the new version of OpenREM:

```
pip install openrem==0.8.1
```

Update the configuration

no changes should be required if upgrading from 0.8.0

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`

Date format

Set the date format for xlsx exports (need to check csv situation). Copy the following code into your `local_settings.py` file if you want to change it from `dd/mm/yyy`:

```
# 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'
```

Time zone and language

Consider setting the timezone and language in `local_settings.py`. See `local_settings.py.example`.

Add additional log file configuration

Warning: If the configuration is not added for the new `openrem_extractor.log` you will find it being created whereever you start the webserver from, and starting the webserver may fail.

Add the new extractor log file configuration to the `local_settings.py` - you can copy the 'Logging configuration' section from `local_settings.py.example` if you haven't made many changes to this section. See the *Log file* settings in the install instructions.

Warning: If you are upgrading from an earlier beta with the Toshiba RDSR creation logs defined, this has changed names and must be modified in `local_settings.py` before the migration below. It should be changed to:

```
LOGGING['loggers']['remapp.extractors.ct_toshiba']['level'] = 'INFO'  # Toshiba␣
↪RDSR creation extractor logs
```

substituting `INFO` for whichever level of logging is desired.

Adding legacy Toshiba CT functionality

No change required if upgrading from 0.8.0

If you need to import data from older Toshiba CT scanners into OpenREM then the following tools need to be available on the same server as OpenREM:

- The Offis DICOM toolkit
- Java
- pixelmed.jar from the PixelMed Java DICOM Toolkit

The paths to these must be set in `local_settings.py` for your system:

```
# Locations of various tools for DICOM RDSR creation from CT images
DCMTK_PATH = 'C:/Apps/dcmtk-3.6.0-win32-i386/bin'
DCMCONV = os.path.join(DCMTK_PATH, 'dcmconv.exe')
DCMMKDIR = os.path.join(DCMTK_PATH, 'dcmmkdir.exe')
JAVA_EXE = 'C:/Apps/doseUtility/windows/jre/bin/java.exe'
JAVA_OPTIONS = '-Xms256m -Xmx512m -Xss1m -cp'
PIXELMED_JAR = 'C:/Apps/doseUtility/pixelmed.jar'
PIXELMED_JAR_OPTIONS = '-Djava.awt.headless=true com.pixelmed.doseocr.OCR -'
```

The example above is for Windows. On linux, if you have installed the Offis DICOM toolkit with `sudo apt install dcmtk` or similar, you can find the path for the configuration above using the command `which dcmconv`. This will be something like `/usr/bin/dcmconv`, so the `DCMTK_PATH` would be `'/usr/bin` and the `DCMCONV` would be `os.path.join(DCMTK_PATH, 'dcmconv')`. Similarly for `DCMMKDIR` and `JAVA_EXE`, which might be `/usr/bin/java`. The pixelmed.jar file should be downloaded from the link above, and you will need to provide the path to where you have saved it.

The list of CT scanners that the extractor works with has been reduced. You can add to this list, but you will need to verify that any systems you configure to use this extractor produce data in OpenREM that you expect.

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
# if changes are detected (not expected between most beta versions)
python manage.py migrate remapp
```

Update static files

In the same shell/command window as you used above run the following command to clear the static files belonging to your previous OpenREM version and replace them with those belonging to the version you have just installed (assuming you are using a production web server. . .):

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

Restart all the services

Follow the guide at *Start all the services*.

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

Upgrading an OpenREM server with no internet access

Follow the instructions found at *Upgrade an offline OpenREM installation*, before returning here to update the database and configuration.

Upgrading from version 0.7.4 or previous 0.8.0 betas

Upgrade

- Back up your database
  - For PostgreSQL on linux you can refer to *Backup the database*
  - For PostgreSQL on Windows you can refer to backupRestorePostgreSQL
  - For a non-production SQLite3 database, simply make a copy of the database file
- Stop any Celery workers
- Consider temporarily disabling your DICOM StoreSCP, or redirecting the data to be processed later

- If you are using a virtualenv, activate it
- Install the new version of OpenREM:

```
pip install openrem==0.8.0
```

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`

Date format

Set the date format for xlsx exports (need to check csv situation). Copy the following code into your `local_settings.py` file if you want to change it from `dd/mm/yyy`:

```
# 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'
```

Time zone and language

Consider setting the timezone and language in `local_settings.py`. See `local_settings.py.example`.

Add additional log file configuration

> **Warning:** If the configuration is not added for the new `openrem_extractor.log` you will find it being created whereever you start the webserver from, and starting the webserver may fail.

Add the new extractor log file configuration to the `local_settings.py` - you can copy the 'Logging configuration' section from `local_settings.py.example` if you haven't made many changes to this section. See the *Log file* settings in the install instructions.

Warning: If you are upgrading from an earlier beta with the Toshiba RDSR creation logs defined, this has changed names and must be modified in `local_settings.py` before the migration below. It should be changed to:

```
LOGGING['loggers']['remapp.extractors.ct_toshiba']['level'] = 'INFO'  # Toshiba␣
↪RDSR creation extractor logs
```

substituting `INFO` for whichever level of logging is desired.

Adding legacy Toshiba CT functionality

If you need to import data from older Toshiba CT scanners into OpenREM then the following tools need to be available on the same server as OpenREM:

- The Offis DICOM toolkit
- Java
- pixelmed.jar from the PixelMed Java DICOM Toolkit

The paths to these must be set in `local_settings.py` for your system:

```
# Locations of various tools for DICOM RDSR creation from CT images
DCMTK_PATH = 'C:/Apps/dcmtk-3.6.0-win32-i386/bin'
DCMCONV = os.path.join(DCMTK_PATH, 'dcmconv.exe')
DCMMKDIR = os.path.join(DCMTK_PATH, 'dcmmkdir.exe')
JAVA_EXE = 'C:/Apps/doseUtility/windows/jre/bin/java.exe'
JAVA_OPTIONS = '-Xms256m -Xmx512m -Xss1m -cp'
PIXELMED_JAR = 'C:/Apps/doseUtility/pixelmed.jar'
PIXELMED_JAR_OPTIONS = '-Djava.awt.headless=true com.pixelmed.doseocr.OCR -'
```

The example above is for Windows. On linux, if you have installed the Offis DICOM toolkit with `sudo apt install dcmtk` or similar, you can find the path for the configuration above using the command `which dcmconv`. This will be something like `/usr/bin/dcmconv`, so the `DCMTK_PATH` would be `'/usr/bin` and the `DCMCONV` would be `os.path.join(DCMTK_PATH, 'dcmconv')`. Similarly for `DCMMKDIR` and `JAVA_EXE`, which might be `/usr/bin/java`. The pixelmed.jar file should be downloaded from the link above, and you will need to provide the path to where you have saved it.

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
# if changes are detected (not expected between most beta versions)
python manage.py migrate remapp
```

Update static files

In the same shell/command window as you used above run the following command to clear the static files belonging to your previous OpenREM version and replace them with those belonging to the version you have just installed (assuming you are using a production web server. . .):

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

Restart all the services

Follow the guide at *Start all the services*.

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

Upgrading an OpenREM server with no internet access

Follow the instructions to *OpenREM Release Notes version 0.7.1* and to *OpenREM Release Notes version 0.7.3* first.

Then use the instructions found at *Upgrade an offline OpenREM installation* again, but this time change the pip commands from `openrem==0.7.1` to `openrem==0.7.4`.

Upgrading from 0.6 series

Follow the instructions to *OpenREM Release Notes version 0.7.1* and to *OpenREM Release Notes version 0.7.3* first.

Finally, return to these instructions to upgrade to 0.7.4.

Upgrading from version 0.7.1

Follow the instructions to *OpenREM Release Notes version 0.7.3* first, then return to these instructions to upgrade to 0.7.4.

Upgrading from version 0.7.3

- Back up your database
  - For PostgreSQL you can refer to *Backup the database*
  - For a non-production SQLite3 database, simply make a copy of the database file
- Stop any Celery workers

- If you are using a virtualenv, activate it
- Install the new version of OpenREM:

```
pip install openrem==0.7.4
```

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\`

Check for any migrations

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

The expected response should is: `No changes detected in app 'remapp'`

Restart all the services

Follow the guide at *Start all the services*.

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

Upgrading an OpenREM server with no internet access

Upgrade using the instructions found at *Upgrade an offline OpenREM installation*, but change the pip commands from `openrem==0.7.1` to `openrem==0.7.3`. If you are still on a 0.6 series install, upgrade to 0.7.1 first.

Upgrading from version 0.7.1

- Back up your database
  - For PostgreSQL you can refer to *Backup the database*
  - For a non-production SQLite3 database, simply make a copy of the database file
- Stop any Celery workers
- If you are using a virtualenv, activate it
- Install the new version of OpenREM:

```
pip install openrem==0.7.3
```

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\`

Check the current status of your migrations

```
python manage.py showmigrations
```

If you are using the PostgreSQL database and installed 0.7.1 as a fresh install, the `remapp` section should look like this:

```
remapp
 [X] 0001_initial
 [X] 0002_0_7_fresh_install_add_median
```

If you installed 0.7.1 as a fresh install and are using a different database – such as MySQL or the built-in test database SQLite3 – the `remapp` section should look like this:

```
remapp
 [X] 0001_initial
```

For both of these scenarios your upgrade is complete and you can *Start all the services*.

If you have an installation that has been upgraded from the 0.6 series, it should have a `remapp` section that looks like this:

```
remapp
 [X] 0001_initial
 [X] 0002_upgrade_0_7_from_0_6
```

For this scenario, please continue and apply the new migration using the instructions below.

If your migrations list is different from these, particularly if there are any migrations listed with an empty `[ ]` check box and you don't know why, please ask a question on the Google group before continuing. Don't forget to tell us what is in the `remapp` section of your `showmigrations` listing and what upgrades you have done so far.

Apply the new migration

Rename the file

```
remapp/migrations/000x_delete_060_acq_field.py.inactive
```

to:

```
remapp/migrations/000x_delete_060_acq_field.py
```

Check that the rename was successful by running `python manage.py showmigrations` again. The new migration should be listed with an empty pair of square brackets.

Now run

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

This should result in an error similar to this:

```
CommandError: Conflicting migrations detected (0002_upgrade_0_7_from_0_6, 000x_delete_
↪060_acq_field in remapp).
To fix them run 'python manage.py makemigrations --merge'
```

Now run

```
python manage.py makemigrations --merge
```

This will then list the merge actions, finishing with the following text:

```
Merging will only work if the operations printed above do not conflict
with each other (working on different fields or models)
Do you want to merge these migration branches? [y/N]
```

Respond with a `y`, then run `python manage.py showmigrations` again. This should result in the following listing:

```
remapp
 [X] 0001_initial
 [ ] 000x_delete_060_acq_field
 [X] 0002_upgrade_0_7_from_0_6
 [ ] 0003_merge
```

Now run the migration:

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

A final `python manage.py showmigrations` should show:

```
remapp
 [X] 0001_initial
 [X] 000x_delete_060_acq_field
 [X] 0002_upgrade_0_7_from_0_6
 [X] 0003_merge
```

Restart all the services

Follow the guide at *Start all the services*.

Import all the failed studies since 0.6 series upgrade

Re-import any fluoroscopy, radiography or mammography data that has not imported since the upgrade from the 0.6 series. This relates to issue #415 on the Bitbucket issue tracker.

If you have any studies complaining

```
remapp.models.DoesNotExist: ProjectionXRayRadiationDose matching query does not exist.
```

You should check to see if the study you are importing has been partially imported before the database was fixed. If it has, you might need to delete it using the delete function in the web interface. You will only see the delete function if you have admin privileges - see *Configure the settings* for details.

Upgrading from 0.6 series

Follow the instructions to *OpenREM Release Notes version 0.7.1* first, then return to these instructions to upgrade to 0.7.3.

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
- Imports
  - 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

Upgrading an OpenREM server with no internet access

Follow the instructions found at *Upgrade an offline OpenREM installation*, before returning here to update the database and configuration.

Upgrading from version 0.6.0

- Back up your database
  - For PostgreSQL you can refer to *Backup the database*
  - For a non-production SQLite3 database, simply make a copy of the database file
- Stop any Celery workers
- The 0.7.0 upgrade must be made from a 0.6.0 (or later) database, and a schema migration is required:

```
pip install openrem==0.7.1
```

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\`

Delete all numbered migration files in openrem's `remapp/migrations` folder, **leaving the 0002 files ending in .inactive**

- If there is no file named `__init__.py` in the `remapp/migrations` folder, please create it.
- If you have accidentally deleted the 0002 files ending in `.inactive`, you can get a new copy from the bitbucket repository.

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

Now rename the file

```
remapp/migrations/0002_upgrade_0_7_from_0_6.py.inactive
```

to:

```
remapp/migrations/0002_upgrade_0_7_from_0_6.py
```

and then run

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

Note: With a large database, this may take some time!

- Review the new `openremproject/local_settings.py.example` file and copy accross the logging section. Then see *Log file* settings in the install docs.

  If you are using PuTTY on Windows to interact with a linux server, you can select the logging configuration section of the example file with your mouse, and it will be automatically copied to the clipboard. Then open the existing `local_settings.py` file with nano, move the curser down to the bottom and click the right mouse button to paste.

Restart all the services!

Some of the commands and services have changed - follow the guide at *Start all the services*.

Upgrading from version 0.7.0 beta 7 or later

- Stop any Celery workers
- You will need to do a database migration.

```
pip install openrem==0.7.1
```

From the openrem folder (see above):

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

- Review the new `local_settings.py.example` file and copy accross the logging section. Then see *Log file* settings in the install docs.

Restart all the services!

Some of the commands and services have changed - follow the guide at *Start all the services*.

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.

Preparing for the upgrade

Convert to South

Make sure you have converted your database to South before attempting an upgrade

Quick reminder of how, if you haven't done it already

```
# Linux: Debian/Ubuntu and derivatives
python /usr/local/lib/python2.7/dist-packages/openrem/manage.py convert_to_south␣
↪remapp
# Linux: other distros. In a virtualenv replace all up to lib/ as appropriate
python /usr/local/lib/python2.7/site-packages/openrem/manage.py convert_to_south␣
↪remapp

# Windows, assuming no virtualenv
python C:\Python27\Lib\site-packages\openrem\manage.py convert_to_south remapp
```

Additional installs

OpenREM requires two additional programs to be installed to enable the new features: *Numpy* for charts, and *pynetdicom* for the DICOM Store Service Class Provider. Note that the version of pynetdicom must be later than the current pypi release!

Install NumPy

For linux:

```
sudo apt-get install python-numpy
# If using a virtualenv, you might need to also do:
pip install numpy
```

For Windows, there are various options:

1. Download executable install file from SourceForge:

    - Download a pre-compiled Win32 .exe NumPy file from http://sourceforge.net/projects/numpy/files/NumPy/. You need to download the file that matches the Python version, which should be 2.7. At the time of writing the latest version was 1.9.2, and the filename to download was `numpy-1.9.2-win32-superpack-python2.7.exe`. The filename is truncated on SourceForge, so you may need to click on the *i* icon to see which is which. It's usually the third *superpack*.
    - Run the downloaded binary file to install NumPy.

2. Or download a `pip` installable wheel file:

- Download NumPy from http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy - `numpy-1.9.2+mkl-cp27-none-win32.whl` is likely to be the right version, unless you have 64bit Python installed, in which case use the `numpy-1.9.2+mkl-cp27-none-win_amd64.whl` version instead.
- Install using pip:

```
pip install numpy-1.9.2+mkl-cp27-none-win32.whl
```

Install pynetdicom

```
pip install https://bitbucket.org/edmcdonagh/pynetdicom/get/default.tar.gz
↪#egg=pynetdicom-0.8.2b2
```

Upgrading from versions prior to 0.5.1

You must upgrade to 0.5.1 first. Instructions for doing this can be found in the *OpenREM Release Notes version 0.5.1*.

Upgrading from version 0.5.1

- Back up your database
  - For PostgreSQL you can refer to *Backing up a PostgreSQL database (Windows)*
  - For a non-production SQLite3 database, simply make a copy of the database file
- The 0.6.0 upgrade must be made from a 0.5.1 (or later) database, and a schema migration is required:

```
pip install openrem==0.6.0

# Linux: Debian/Ubuntu and derivatives
python /usr/local/lib/python2.7/dist-packages/openrem/manage.py schemamigration --
↪auto remapp
python /usr/local/lib/python2.7/dist-packages/openrem/manage.py migrate remapp
# Linux: other distros. In a virtualenv replace all up to lib/ as appropriate
python /usr/local/lib/python2.7/site-packages/openrem/manage.py schemamigration --
↪auto remapp
python /usr/local/lib/python2.7/site-packages/openrem/manage.py migrate remapp
# Windows:
python C:\Python27\Lib\site-packages\openrem\manage.py schemamigration --auto remapp
python C:\Python27\Lib\site-packages\openrem\manage.py migrate remapp
```

- Restart the services
  - Restart the webserver
  - Restart Celery

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

Upgrades: Convert to South

Always make sure you have converted your database to South before attempting an upgrade

Quick reminder of how, if you haven't done it already

```
# Linux: Debian/Ubuntu and derivatives
python /usr/local/lib/python2.7/dist-packages/openrem/manage.py convert_to_south␣
↪remapp
# Linux: other distros. In a virtualenv replace all up to lib/ as appropriate
python /usr/local/lib/python2.7/site-packages/openrem/manage.py convert_to_south␣
↪remapp

# Windows, assuming no virtualenv
python C:\Python27\Lib\site-packages\openrem\manage.py convert_to_south remapp
```

Upgrading from before 0.5.0

Upgrading from version 0.3.9 or earlier

- Back up your database
  - For PostgreSQL you can refer to *Backing up a PostgreSQL database (Windows)*
  - For a non-production SQLite3 database, simply make a copy of the database file
- `pip install openrem==0.4.2`
- Migrate the schema

  ```
  # Linux: Debian/Ubuntu and derivatives
  python /usr/local/lib/python2.7/dist-packages/openrem/manage.py␣
  ↪schemamigration --auto remapp
  # Linux: other distros. In a virtualenv replace all up to lib/ as␣
  ↪appropriate
  python /usr/local/lib/python2.7/site-packages/openrem/manage.py␣
  ↪schemamigration --auto remapp
  # Windows:
  python C:\Python27\Lib\site-packages\openrem\manage.py schemamigration --
  ↪auto remapp
  ```

When South has considered the changes to the schema, you will see the following message:

```
? The field 'Observer_context.device_observer_name' does not have a␣
↪default specified, yet is NOT NULL.
? Since you are making this field nullable, you MUST specify a default
? value to use for existing rows. Would you like to:
?  1. Quit now.
?  2. Specify a one-off value to use for existing columns now
?  3. Disable the backwards migration by raising an exception; you can␣
↪edit the migration to fix it later
? Please select a choice: 3
```

– As per the final line above, please select option 3, and then execute the migration:

```
# Linux: Debian/Ubuntu and derivatives
python /usr/local/lib/python2.7/dist-packages/openrem/manage.py migrate␣
↪remapp
# Linux: other distros. In a virtualenv replace all up to lib/ as␣
↪appropriate
python /usr/local/lib/python2.7/site-packages/openrem/manage.py migrate␣
↪remapp

# Windows, assuming no virtualenv
python C:\Python27\Lib\site-packages\openrem\manage.py migrate remapp
```

- Create and populate the database settings in the new `local_settings.py` file

  The `openrem/openrem` folder can be found at:

```
# Linux: Debian/Ubuntu and derivatives
/usr/lib/python2.7/dist-packages/openrem/openrem
# Linux: other distros. In a virtualenv replace all up to lib/ as␣
↪appropriate
/usr/lib/python2.7/site-packages/openrem/openrem
# Windows:
C:\Python27\Lib\site-packages\openrem\openrem
```

  In the `openrem/openrem` folder, create a new file called `local_settings.py` and copy the contents of this link into a the file and save it. Alternatively, rename `local_settings.py.example` to `local_settings.py` - this is an older version of the file.

  Copy the database details from `settings.py` into `local_settings.py`

- Change the secret key - you can use http://www.miniwebtool.com/django-secret-key-generator/ to generate a new one
- Move the existing `settings.py` out of the python directories (delete or move somewhere as a backup)
- Rename the `settings.py.new` to `settings.py`
- Restart your webserver to check everything looks ok
- Add some users
  - Go to the admin interface (eg http://localhost:8000/admin) and log in with the user created when you originally created the database (the `manage.py syncdb` command - *Do you want to create a superuser*)
  - Create some users and add them to the appropriate groups (if there are no groups, go to the OpenREM homepage and they should be there when you go back to admin).
    * `viewgroup` can browse the data only

    * `exportgroup` can do as view group plus export data to a spreadsheet, and will be able to import height and weight data in due course (See Issue #21)
    * `admingroup` can delete studies in addition to anything the export group can do

Upgrading from versions 0.4.0 - 0.4.2

- Back up your database
  - For PostgreSQL you can refer to *Backing up a PostgreSQL database (Windows)*
  - For a non-production SQLite3 database, simply make a copy of the database file
- Install version 0.5.0
  - `pip install openrem==0.5.0`
- Install RabbitMQ
  - Linux - Follow the guide at http://www.rabbitmq.com/install-debian.html
  - Windows - Follow the guide at http://www.rabbitmq.com/install-windows.html
- Move `local_settings.py` details from `openrem` to `openremproject`

  The inner `openrem` Django project folder has now been renamed `openremproject` The customised `local_settings.py` file and the `wsgi.py` file have remain in the old `openrem` folder. The `openrem/openrem` folder can be found as detailed in the upgrade from '0.3.9 or earlier' instructions above, and the new `openrem/openremproject` folder is in the same place.

  - Move `local_settings.py` to `openremproject`. If you have kept the older local_settings file, you may like to instead rename the `local_settings.py.example` file instead, then transfer the database settings and change the secret key.
  - Set the path for `MEDIA_ROOT`. The webserver needs to be able to write to this location - it is where OpenREM will store export files etc so that they can be downloaded. For suggestions, see the main _install instructions.
  - Set `ALLOWED_HOSTS`. For details see the Django docs A `'*'` allows any host - see the Django docs for the risk of this.
- Move `wsgi.py` from `openrem` to `openremproject` or rename `wsgi.py.example` in `openremproject`

  If you haven't edited it, simply rename the new version in `openremproject`. Otherwise, move the old version and edit the following line as follows:

```
# Old:
os.environ.setdefault("DJANGO_SETTINGS_MODULE", "openrem.settings")
# New:
os.environ.setdefault("DJANGO_SETTINGS_MODULE", "openremproject.settings")
```

- Tidying up - you should delete the old `openrem` folder - you might like to take a backup first!
- Update web server configuration

  The configuration of the webserver will need to be updated to reflect the new location for the `settings.py` file and the `wsgi.py` file.

  If you are using the built-in test webserver, static files will no-longer be served unless you use the `insecure` option:

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

- Migrate the schema

```
# Linux: Debian/Ubuntu and derivatives
python /usr/local/lib/python2.7/dist-packages/openrem/manage.py␣
↪schemamigration --auto remapp
python /usr/local/lib/python2.7/dist-packages/openrem/manage.py migrate␣
↪remapp
# Linux: other distros. In a virtualenv replace all up to lib/ as␣
↪appropriate
python /usr/local/lib/python2.7/site-packages/openrem/manage.py␣
↪schemamigration --auto remapp
python /usr/local/lib/python2.7/site-packages/openrem/manage.py migrate␣
↪remapp
# Windows:
python C:\Python27\Lib\site-packages\openrem\manage.py schemamigration --
↪auto remapp
python C:\Python27\Lib\site-packages\openrem\manage.py migrate remapp
```

After restarting the webserver, you should now have OpenREM 0.5.0 up and running. If you wish to test export functionality at this stage, start the Celery task queue - instructions in the *Installing OpenREM* docs or at the end of this guide.

Now move to *Upgrading from version 0.5.0*.

Upgrading from version 0.4.3

- Back up your database
  - For PostgreSQL you can refer to *Backing up a PostgreSQL database (Windows)*
  - For a non-production SQLite3 database, simply make a copy of the database file
- The 0.5.1 upgrade *must* be made from a 0.5.0 database, so a schema migration is required:

```
pip install openrem==0.5.0

    # Linux: Debian/Ubuntu and derivatives
    python /usr/local/lib/python2.7/dist-packages/openrem/manage.py␣
↪schemamigration --auto remapp
    python /usr/local/lib/python2.7/dist-packages/openrem/manage.py␣
↪migrate remapp
    # Linux: other distros. In a virtualenv replace all up to lib/ as␣
↪appropriate
    python /usr/local/lib/python2.7/site-packages/openrem/manage.py␣
↪schemamigration --auto remapp
    python /usr/local/lib/python2.7/site-packages/openrem/manage.py␣
↪migrate remapp
    # Windows:
    python C:\Python27\Lib\site-packages\openrem\manage.py␣
↪schemamigration --auto remapp
    python C:\Python27\Lib\site-packages\openrem\manage.py migrate remapp
```

Upgrading from version 0.5.0

- Back up your database
  - For PostgreSQL you can refer to *Backing up a PostgreSQL database (Windows)*
  - For a non-production SQLite3 database, simply make a copy of the database file
- Install 0.5.1:

```
pip install openrem==0.5.1
```

- Find out how many migration files you have

  Method 1:

  Use a file browser or terminal to list the contents of the `migrations` folder, eg:

```
# Linux: Debian/Ubuntu and derivatives
ls /usr/local/lib/python2.7/dist-packages/openrem/remapp/
↪migrations/
# Linux: other distros. In a virtualenv replace all up to lib/ as␣
↪appropriate
ls /usr/local/lib/python2.7/site-packages/openrem/remapp/
↪migrations/
# Windows (alternatively use the file browser):
dir C:\Python27\Lib\site-packages\openrem\remapp\migrations\
```

  Method 2:

  Use the Django `manage.py` program to list the existing migrations:

```
# Linux: Debian/Ubuntu and derivatives
python /usr/local/lib/python2.7/dist-packages/openrem/manage.py␣
↪migrate --list remapp
# Linux: other distros. In a virtualenv replace all up to lib/ as␣
↪appropriate
python /usr/local/lib/python2.7/site-packages/openrem/manage.py␣
↪migrate --list remapp
# Windows
python C:\Python27\Lib\site-packages\openrem\manage.py migrate --
↪list remapp
```

  The output should look something like this - there can be any number of existing migrations (though 0001_initial will always be present):

```
remapp
(*) 0001_initial
(*) 0002_auto__chg_field_ct_accumulated_dose_data_ct_dose_length_product_
↪total_
(*) 0003_auto__chg_field_general_equipment_module_attributes_station_name
(*) 0004_auto__chg_field_ct_radiation_dose_comment__chg_field_accumulated_
↪proje
(*) 0005_auto__add_exports__add_size_upload
(*) 0006_auto__chg_field_exports_filename
(*) 0007_auto__add_field_irradiation_event_xray_detector_data_relative_
↪xray_exp
( ) 000x_051datamigration
( ) 000x_051schemamigration
```

- Rename the two 051 migration files to follow on from the existing migrations, for example `0008_051schemamigration.py` and `0009_051datamigration.py` for the existing migrations above, or `0002_051schemamigration.py` and `0003_051datamigration.py` if the only migration is the initial migration. The `051schemamigration` **must** come before the `051datamigration`.

  If you are using linux, you might like to do it like this (from within the `openrem` folder):

```
mv remapp/migrations/000{x,8}_051schemamigration.py
mv remapp/migrations/000{x,9}_051datamigration.py
```

- If you now re-run `migrate --list remapp` you should get a listing with the `051schemamigration` and the `051datamigration` listed at the end:

```
remapp
 (*) 0001_initial
 (*) 0002_auto__chg_field_ct_accumulated_dose_data_ct_dose_length_product_total_
 (*) 0003_auto__chg_field_general_equipment_module_attributes_station_name
 (*) 0004_auto__chg_field_ct_radiation_dose_comment__chg_field_accumulated_proje
 (*) 0005_auto__add_exports__add_size_upload
 (*) 0006_auto__chg_field_exports_filename
 (*) 0007_auto__add_field_irradiation_event_xray_detector_data_relative_xray_exp
 ( ) 0008_051schemamigration
 ( ) 0009_051datamigration
```

  The star indicates that a migration has already been completed. If you have any that are not completed apart from the `051schemamigration` and the `051datamigration` then please resolve these first.

- Now execute the migrations:

```
# Linux: Debian/Ubuntu and derivatives
python /usr/local/lib/python2.7/dist-packages/openrem/manage.py migrate remapp
# Linux: other distros. In a virtualenv replace all up to lib/ as appropriate
python /usr/local/lib/python2.7/site-packages/openrem/manage.py migrate remapp
# Windows
python C:\Python27\Lib\site-packages\openrem\manage.py migrate remapp
```

Restart the web server

If you are using the built-in test web server (*not for production use*):

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

Otherwise restart using the command for your web server

Restart the Celery task queue

For testing, in a new shell:

```
# Linux: Debian/Ubuntu and derivatives
cd /usr/local/lib/python2.7/dist-packages/openrem/
# Linux: other distros. In a virtualenv replace all up to lib/ as appropriate
cd /usr/local/lib/python2.7/site-packages/openrem/
# Windows
cd C:\Python27\Lib\site-packages\openrem\
```

(continues on next page)

(continued from previous page)

```
# All
celery -A openremproject worker -l info
```

For production use, see http://celery.readthedocs.org/en/latest/tutorials/daemonizing.html

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

Always make sure you have converted your database to South before attempting an upgrade

Quick reminder of how, if you haven't done it already:

Linux:

```
python /usr/local/lib/python2.7/dist-packages/openrem/manage.py convert_to_
↪south remapp
```

Windows:

```
python C:\Python27\Lib\site-packages\openrem\manage.py convert_to_south␣
↪remapp
```

Upgrading from versions before 0.4.3

If you are upgrading from 0.3.9 or earlier, you will need to upgrade to version 0.4.2 first. See the *OpenREM Release Notes version 0.4.3*.

If you are upgrading from 0.4.0 or later, the instructions in *OpenREM Release Notes version 0.4.3* still need to be followed to install/setup RabbitMQ and Celery and to update the configuration files, but you can go straight to 0.5.0 rather than installing 0.4.3.

Upgrading from version 0.4.3

```
pip install openrem==0.5.0
```

(Will need `sudo` or equivalent if using linux without a virtualenv)

Database migration

Assuming no virtualenv

Linux:

```
python /usr/local/lib/python2.7/dist-packages/openrem/manage.py schemamigration --
↪auto remapp
python /usr/local/lib/python2.7/dist-packages/openrem/manage.py migrate remapp
```

Windows:

```
C:\Python27\Lib\site-packages\openrem\manage.py schemamigration --auto remapp
C:\Python27\Lib\site-packages\openrem\manage.py migrate remapp
```

Restart the web server

If you are using the built-in test web server (*not for production use*):

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

Otherwise restart using the command for your web server

Restart the Celery task queue

For testing, in a new shell: *(assuming no virtualenv)*

Linux:

```
cd /usr/local/lib/python2.7/dist-packages/openrem/
celery -A openremproject worker -l info
```

Windows:

```
cd C:\Python27\Lib\site-packages\openrem\
celery -A openremproject worker -l info
```

For production use, see http://celery.readthedocs.org/en/latest/tutorials/daemonizing.html

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

Always make sure you have converted your database to South before attempting an upgrade

Quick reminder of how, if you haven't done it already:

Linux:

```
python /usr/local/lib/python2.7/dist-packages/openrem/manage.py convert_to_
↪south remapp
```

Windows:

```
python C:\Python27\Lib\site-packages\openrem\manage.py convert_to_south␣
↪remapp
```

Upgrading from 0.3.9 or earlier

It is essential that you upgrade to at least 0.4.0 first, then upgrade to 0.4.3. Otherwise the settings file will be overwritten and you will lose your database settings. There is also a trickier than usual database migration and instructions for setting up users. *Fresh installs should start with the latest version.*

Upgrade to version 0.4.2

```
pip install openrem==0.4.2
```

(Will need `sudo` or equivalent if using linux without a virtualenv)

Then follow the instructions in *OpenREM Release Notes version 0.4.0* from migrating the database onwards, before coming back to these instructions.

Upgrading from 0.4.0 or above

Install OpenREM version 0.4.3

```
pip install openrem==0.4.3
```

(Will need `sudo` or equivalent if using linux without a virtualenv)

RabbitMQ

The message broker RabbitMQ needs to be installed to enable the export and upload features

- Linux - Follow the guide at http://www.rabbitmq.com/install-debian.html
- Windows - Follow the guide at http://www.rabbitmq.com/install-windows.html

Move and edit local_settings.py file and wsgi.py files

The inner `openrem` Django project folder has now been renamed `openremproject` to avoid import confusion that prevented Celery working on Windows.

When you upgrade, the `local_settings.py` file and the `wsgi.py` file will remain in the old `openrem` folder. Both need to be moved across to the `openremproject` folder, and edited as below.

The new and old folders will be found in:

- Linux: `/usr/local/lib/python2.7/dist-packages/openrem/`
- Linux with virtualenv: `/home/myname/openrem/lib/python2.7/site-packages/openrem/`
- Windows: `C:\Python27\Lib\site-packages\openrem\`

Edit the local_settings.py file

The `MEDIA_ROOT` path needs to be defined. This is the place where the study exports will be stored for download and where the patient size information csv files will be stored temporarily whilst they are bing processed.

The path set for `MEDIA_ROOT` is up to you, but the user that runs the webserver must have read/write access to the location specified because it is the webserver than reads and writes the files. In a debian linux, this is likely to be `www-data` for a production install. Remember to use forward slashes in the config file, even for Windows.

Linux example:

```
MEDIA_ROOT = "/var/openrem/media/"
```

Windows example:

```
MEDIA_ROOT = "C:/Users/myusername/Documents/OpenREM/media/"
```

The `ALLOWED_HOSTS` needs to be defined, as the `DEBUG` mode is now set to `False`. This needs to contain the server name or IP address that will be used in the URL in the web browser. For example:

```
ALLOWED_HOSTS = [
    '192.168.56.102',
    '.doseserver.',
    'localhost',
]
```

A dot before a hostname allows for subdomains (eg www.doseserver), a dot after a hostname allows for FQDNs (eg doseserver.ad.trust.nhs.uk). Alternatively, a single `'*'` allows any host, but removes the security the feature gives you.

Edit the wsgi.py file with the new project folder name

If you aren't using the wsgi.py file as part of your webserver setup, you might like to simply rename the `wsgi.py.example` file in the `openremproject` folder.

If you are using it, edit the line:

```
os.environ.setdefault("DJANGO_SETTINGS_MODULE", "openrem.settings")
```

to read:

```
os.environ.setdefault("DJANGO_SETTINGS_MODULE", "openremproject.settings")
```

Tidying up

Finally, you should delete the old `openrem` folder - you might like to take a backup first!

Database migration

Assuming no virtualenv

Linux:

```
python /usr/local/lib/python2.7/dist-packages/openrem/manage.py schemamigration --
↪auto remapp
python /usr/local/lib/python2.7/dist-packages/openrem/manage.py migrate remapp
```

Windows:

```
C:\Python27\Lib\site-packages\openrem\manage.py schemamigration --auto remapp
C:\Python27\Lib\site-packages\openrem\manage.py migrate remapp
```

Web server

If you are using a production webserver, you will probably need to edit some of the configuration to reflect the change in location of `settings.py`, for example `DJANGO_SETTINGS_MODULE = openremproject.settings`, and then restart the web server. You may need to do something similar for the location of `wsgi.py`.

If you are using the built-in test web server (*not for production use*), then the static files will not be served unless you add `--insecure` to the command. This is one of the consequences of setting `DEBUG` to `False`:

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

Start the Celery task queue

Note: The webserver and Celery both need to be able to read and write to the `MEDIA_ROOT` location. Therefore you might wish to consider starting Celery using the same user or group as the webserver, and setting the file permissions accordingly.

For testing, in a new shell: *(assuming no virtualenv)*

Linux:

```
cd /usr/local/lib/python2.7/dist-packages/openrem/
celery -A openremproject worker -l info
```

Windows:

```
cd C:\Python27\Lib\site-packages\openrem\
celery -A openremproject worker -l info
```

For production use, see http://celery.readthedocs.org/en/latest/tutorials/daemonizing.html

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

- `pip install openrem==0.4.2` *Go straight to 0.4.2*
- Migrate the database

  Warning: A database migration is required that will need a choice to be made

  - **Linux:** `python /usr/lib/python2.7/dist-packages/openrem/manage.py schemamigration --auto remapp`
  - **Windows:** `C:\Python27\Lib\site-packages\openrem\manage.py schemamigration --auto remapp`

  When South has considered the changes to the schema, you will see the following message:

  ```
  ? The field 'Observer_context.device_observer_name' does not have a␣
  ↪default specified, yet is NOT NULL.
  ? Since you are making this field nullable, you MUST specify a default
  ? value to use for existing rows. Would you like to:
  ?  1. Quit now.
  ?  2. Specify a one-off value to use for existing columns now
  ?  3. Disable the backwards migration by raising an exception; you can␣
  ↪edit the migration to fix it later
  ? Please select a choice: 3
  ```

  As per the final line above, the correct choice is `3`. The fields that are now nullable previously weren't. Existing data in those fields will have a value, or those tables don't exist in the current database. The problem scenario is if after the migration these tables are used and one of the new nullable fields is left as null, you will not be able to migrate back to the old database schema without error. This is not something that you will want to do, so this is ok.

  Do the migration:

  - **Linux:** `python /usr/lib/python2.7/dist-packages/openrem/manage.py migrate remapp`
  - **Windows:** `C:\Python27\Lib\site-packages\openrem\manage.py migrate remapp`

- Update the settings files

  Warning: The settings file has changed and will need to be manually edited.

  Changes need to be made to the `settings.py` file where the database details are stored. Normally upgrades don't touch this file and the copy in the upgrade has a `.example` suffix. **This upgrade and potentially future ones will need to change this file**, so the format has been changed. The `settings.py` file will now be replaced each time the code is upgraded. In addition, there is a new `local_settings.py` file that contains things that are specific to your installation, such as the database settings.

  This upgrade will include a file called `settings.py.new` and the `local_settings.py.example` file. You will need to do the following:

  - Copy the database settings from your current `settings.py` file to the `local_settings.py.example` file and rename it to remove the `.example`. Both of these files are in the `openrem/openrem` directory, which will be somewhere like

        - Linux: `/usr/lib/python2.7/dist-packages/openrem/openrem/`
        - Windows: `C:\Python27\Lib\site-packages\openrem\openrem\`
    - Move the existing `settings.py` out of the python directories
    - Rename the `settings.py.new` to `settings.py`

- Create a new secret key

  All versions of openrem ship with the same secret key. This key is used for web security checks, and should be unique (and secret) for each installation.

    - Generate a new secret key - http://www.miniwebtool.com/django-secret-key-generator/ is a suitable method of creating a new key.
    - Copy the new key and use it to replace the default key in the `local_settings.py` file

- Restart your webserver
- Add some users
    - Go to the admin interface (eg http://localhost:8000/admin) and log in with the user created when you originally created the database (`manage.py syncdb`)
    - Create some users and add them to the appropriate groups (if there are no groups, go to the OpenREM homepage and they should be created).
        - `viewgroup` can browse the data only
        - `exportgroup` can do as view group plus export data to a spreadsheet, and will be able to import height and weight data in due course (See Issue #21)
        - `admingroup` can delete studies in addition to anything the export group can do

11.3 Contributing authors

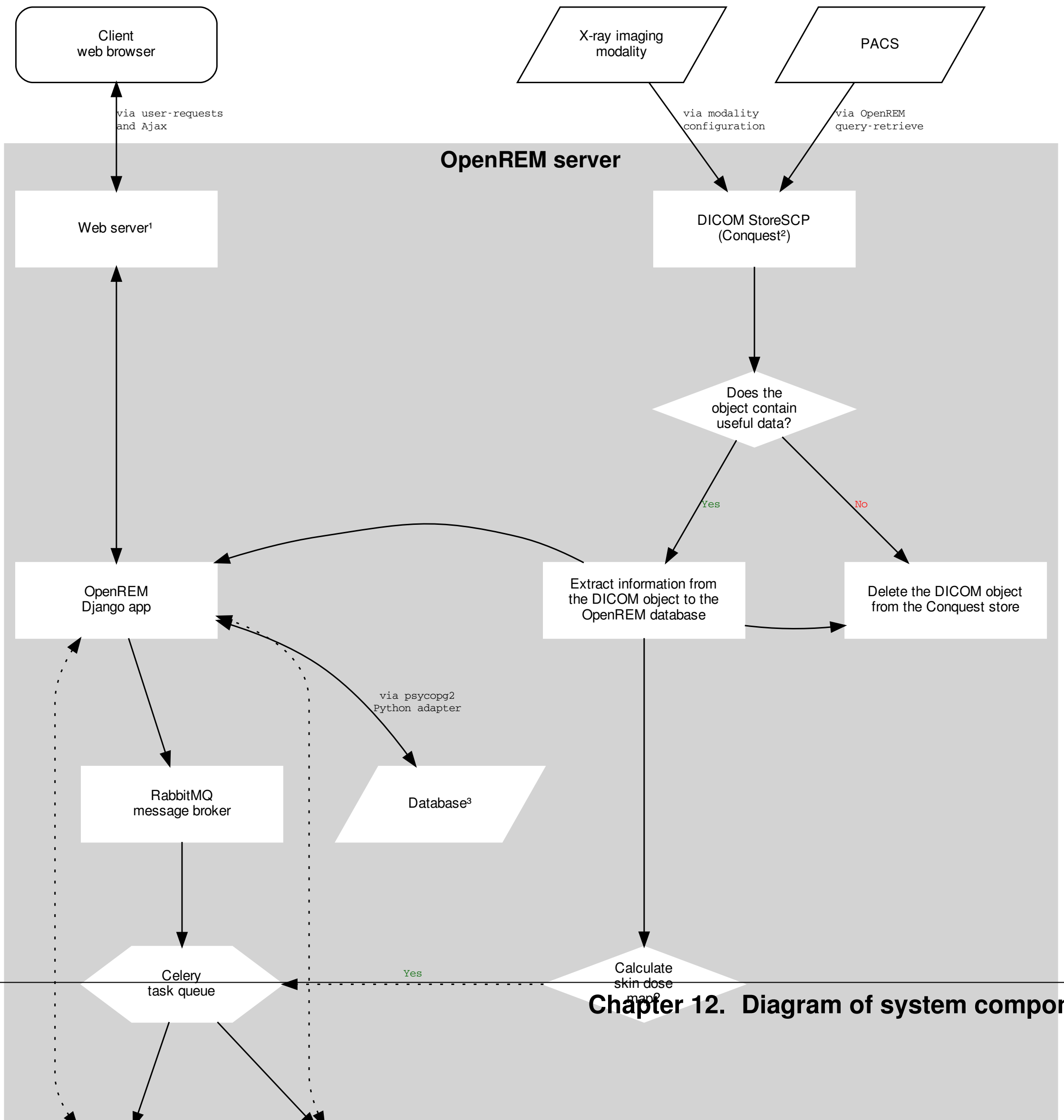
Many people have contributed to OpenREM - either with code, documentation, bugs, examples or ideas, including:

- BjörnAlbers
- Erlend Andersen
- Njål Brekke
- Elly Castellano
- Jonathan Cole
- Jamie Dormand
- Ben Earner
- Daniel Gordon
- Hamid Khosravi
- Laurence King
- Eivind Larsen
- John Loveland
- Ed McDonagh
- Richard Miles

- Luuk Oostveen
- David Platten
- Richard Raynor
- Erik-Jan Rijkhorst
- Arnold Schilham
- Tim de Wit

CHAPTER 12

Diagram of system components



12.1 Alternatives

12.1.1 1: Web servers

The recommended web server for Windows is Microsoft IIS - see *Running OpenREM on Windows with IIS* for details. This has replaced the recommendation to use Apache due to difficulties in obtaining the required binary files, as described in the *Web servers* section of the installation document.

The recommended web server for Linux is Gunicorn with NGINX - see *Running OpenREM on Linux with Gunicorn and NGINX* for details.

Alternatively, a built-in web server is included that will suffice for testing purposes and getting started.

12.1.2 2: DICOM Store node

Any DICOM Store can be used, as long as it can be used to call the OpenREM import script. A built-in store is available, but not recommended for production use. See *DICOM Network Configuration* for more details. Conquest is the recommended DICOM Store service to use.

12.1.3 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.

CHAPTER 13

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index