OpenREM Documentation

Release 0.7.3

Ed McDonagh

August 30, 2016

Contents

OpenREM is an opensource framework created for the purpose of radiation exposure monitoring. The software is capable of importing and displaying data from a wide variety of x-ray dose related sources, and then 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:

First time installation

1.1 Pre-installation preparations

1.1.1 Install Python 2.7.x and pip

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

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\
```

Setuptools and pip

Install setuptools and pip – for details go to http://www.pip-installer.org/en/latest/installing.html. 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 getpip.py file:

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

Fig. 1.1: Python installation customisation dialogue

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.

Note: Before continuing, *consider virtualenv*

1.1.3 Install NumPy

Numpy is required for charts. OpenREM will work without NumPy, but charts will not be displayed.

For linux:

```
sudo apt-get install python-numpy
# If using a virtualenv, you might need to also do:
pip install numpy
```

For Windows:

Download NumPy from http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy

- Find the right version - look for **numpy-x.xx.x+mkl-cp27-cp27m-win32.whl** for 32-bit Windows or
- **numpy-x.xx.x+mkl-cp27-cp27m-win_amd64.whl** for 64-bit Windows.
- At the time of writing, `x.xx.x` was `1.11.0` - choose the latest version
- Install using pip:

```
pip install numpy1.11.0+mklcp27-cp27mwin32.whl
# or
pip install numpy1.11.0+mklcp27cp27mwin_amd64.whl
# changing the filename appropriately
```

1.1.4 Install pynetdicom (edited version)

Pynetdicom is used for the DICOM Store SCP and Query Retrieve SCU functions. See DICOM Store and QR for details.

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

1.1.5 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)

1.1.6 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: `lib/python2.7/site-packages/openrem/`
- Windows: `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 Install OpenREM 0.7

```
pip install openrem==0.7.3
```

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

1.2.2 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: `lib/python2.7/site-packages/openrem/`
- Windows: `C:\Python27\Lib\site-packages\openrem\`
- Windows virtualenv: `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 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.

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
logfilename = os.path.join(MEDIA_ROOT, "openrem.log")
qrfilename = os.path.join(MEDIA_ROOT, "openrem_qr.log")
storefilename = os.path.join(MEDIA_ROOT, "openrem_store.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
```

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
```

Finally you can set the logging level. Options are `DEBUG`, `INFO`, `WARNING`, `ERROR`, and `CRITICAL`, with progressively less logging.

```
LOGGING['loggers']['remapp']['level'] = 'INFO'              # General logs
LOGGING['loggers']['remapp.netdicom.qrscu']['level'] = 'INFO'      # Query Retrieve SCU logs
LOGGING['loggers']['remapp.netdicom.storescp']['level'] = 'INFO'  # Store SCP logs
```

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/`

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

- Virtualenv: `cd lib/python2.7/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 makemigrations --empty remapp
python manage.py migrate
```

The first command will create a skeleton `0001_initial.py` migration file. The second command runs the migration files, and will display the text `Applying remapp.0002_0_7_fresh_install_add_median... OK`, indicating that the median function has been added.

1.2.3 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!

1.2.4 Further instructions

Production webservers

Unlike the database, the production webserver can be left till later and can be changed again at any time.

For performance it is recommended that a production webserver is used instead of the inbuilt 'runserver'. Popular choices would be either Apache or you can do as the cool kids do and use Gunicorn with nginx.

The django website has instructions and links to get you set up with Apache.

An advanced guide using Apache, including auto-restarting the server when the code changes, has been contributed here: `apache_on_windows`

DICOM Store and query-retrieve

The best (and only practical way in a production environment) to get DICOM data into OpenREM is to have a DICOM store node (Store Service Class Provider/SCP) and possibly a query-retrieve service class user too.

To find out more about this, refer to the DICOM Store and QR docs.

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 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`

PostgreSQL Python connector from http://www.lfd.uci.edu/~gohlke/pythonlibs/#psycopg

- Find the right version - look for `psycopg2-x.x.x-cp27-cp27m-win32.whl` for 32-bit Windows or
- `psycopg2-x.x.x-cp27-cp27m-win_amd64.whl` for 64-it Windows.
- At the time of writing, `x.x.x` was `2.6.1` - choose the latest `cp27` version

NumPy from http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy

- Find the right version - look for `numpy-x.xx.x+mkl-cp27-cp27m-win32.whl` for 32-bit Windows or
- `numpy-x.xx.x+mkl-cp27-cp27m-win_amd64.whl` for 64-bit Windows.
- At the time of writing, `x.xx.x` was `1.11.0` - choose the latest `cp27` version

Pynetdicom from https://bitbucket.org/edmcdonagh/pynetdicom/get/default.tar.gz#egg=pynetdicom-0.8.2b2

- The downloaded file will be named something like `edmcdonagh-pynetdicom-2da8a57b53b3.tar.gz`
- Note: this version is modified in comparison to the version in PyPI

A webserver such as Apache, 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 a directory to collect all the packages in, then use pip to download them all:

```
mkdir openremfiles
pip install -d openremfiles openrem==0.7.3
```

Copy everything to the Windows machine

- Add the `pynetdicom` file, the `psycopg2` file and the `numpy` file to the directory with the other python packages
- 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

```
pip install openremfiles\numpy1.11.0+mklcp27-cp27mwin32.whl
# or if you have the 64 bit version
pip install openremfiles\numpy1.11.0+mklcp27-cp27mwin_amd64.whl
# adjusting the version number appropriately

pip install --no-index --find-links=openremfiles openrem==0.7.3

pip install openremfiles\edmcdonagh-pynetdicom-2da8a57b53b3.tar.gz
```

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 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.

1.4.1 PostgreSQL database (Linux)

Creating the database

Install PostgreSQL and the python connector

```
sudo apt-get install postgresql libpq-dev
```

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

```
pip install psycopg2
```

Change the security configuration

The default security settings are too restrictive to allow access to the database. Assumes version `9.4`, change as appropriate.

```
sudo nano /etc/postgresql/9.4/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`

Restart PostgreSQL so the new settings take effect:

```
sudo service postgresql restart
```

Optional: Specify the 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 `9.4` (change both as appropriate)

```
sudo service postgresql stop
mkdir /var/openrem/database
sudo cp -aRv /var/lib/postgresql/9.4/main /var/openrem/database/
sudo nano /etc/postgresql/9.4/main/postgresql.conf
```

Change the line

```
data_directory = '/var/lib/postgresql/9.4/main'
```

to

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

then restart PostgreSQL:

```
sudo service postgresql start
```

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: `lib/python2.7/site-packages/openrem/`

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

- Windows virtualenv: `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*)

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
```

Alternative instructions and further reference

Previous versions had instructions that used different backup options and the `pg_restore` command. To review these, please refer to the 0.6.2 documentation at docs.openrem.org/en/0.6.2/

Further details can be found on the PostgreSQL website

Useful PostgreSQL commands

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

-- List users
\du

-- List databases
\l

-- Exit the console
\q
```

1.4.2 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

- Download psycopg2 from http://www.lfd.uci.edu/~gohlke/pythonlibs/. Make sure it matches your python and Windows version.

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 III

- Click on servers to expand
- Double click on PostgreSQL 9.4
- 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.
- 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.

Install psycopg2

```
pip install psycopg2-2.6.1-cp27-cp27m-win32.whl
# or if you have the 64-bit version
pip install psycopg2-2.6.1-cp27-cp27m-win_amd64.whl
# adjusting the version number appropriately
```

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': 'openrem_db',`
- `'USER': 'openremuser',`
- `'PASSWORD': 'openrem_pw',`

1.4.3 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
```

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 openrem
```

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'(
```

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 > ba
```

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 openremda
```

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
```

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 >

"C:\Program Files\7-Zip\7z.exe" a F:\OpenREMdatabase\backup\%mydate%_%mytime%_openremdatabasemysql.zi

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

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

Upgrade to OpenREM 0.7.1

2.1 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

2.1.1 Changes since 0.7.0

Extremely minor change to the documenation links

2.2 Upgrading an OpenREM server with no internet access

2.2.1 Upgrade an offline OpenREM installation

Upgrading from OpenREM version 0.6 or later 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.

On a computer with internet access

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

```
mkdir openremfiles
pip install -d openremfiles openrem==0.7.1
```

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 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 now, then

```
pip install --no-index --find-links=openremfiles openrem==0.7.1
```

Now continue with *Upgrading from version 0.6.0* or *Upgrading from version 0.7.0 beta 7 or later* as appropriate, starting just after the `pip install` line.

2.3 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: `lib/python2.7/site-packages/openrem/`

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

- Windows virtualenv: `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.

2.3.1 Restart all the services!

Some of the commands and services have changed - follow the guide at Start all the services.

2.4 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.

2.4.1 Restart all the services!

Some of the commands and services have changed - follow the guide at Start all the services.

Upgrade to OpenREM 0.7.3

3.1 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

3.2 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.

3.3 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: `lib/python2.7/site-packages/openrem/`

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

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

3.3.1 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.

3.3.2 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
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
```

3.3.3 Restart all the services

Follow the guide at Start all the services.

3.3.4 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.

3.4 Upgrading from 0.6 series

Follow the instructions to Upgrade to OpenREM 0.7.1 first, then return to these instructions to upgrade to 0.7.3.

Start all the services

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

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

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

4.1.1 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*.

4.1.2 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 192.168.1.10:8000
```

Make sure you **change the IP address** to the address of the server! On Windows you can find the IP address information by typing

```
ipconfig
```

You are looking for a line that has `IPv4 Address` followed by four numbers with dots between, similar to the numbers before the colon in the command above.

With a linux server, type

```
ip add
```

Again you are looking for the same dotted number, this time it will be after `inet`. In both examples, you might have several to choose from depending on how many network cards (real or virtual) your computer has. Determining which one is which is probably beyond the scope of these instructions! If you get the IP address completely wrong, the command will fail with the error: `Error:  That IP address can't be assigned-to`.

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.

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

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

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

Linux - \ is the line continuation character:

```
celery multi start default -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 -A openremproject -c 4 -Q default ^
--pidfile=C:\path\to\media\celery\default.pid --logfile=C:\path\to\media\celery\default.log
```

For production use, see *Daemonising Celery* below

Set the number of workers (concurrency, `-c`) as you see fit. 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.

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.

4.3 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 DICOM Store and QR 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: `lib/python2.7/site-packages/openrem/` (remember to activate the virtualenv)
- Windows: `C:\Python27\Lib\site-packages\openrem\`
- Windows virtualenv: `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.
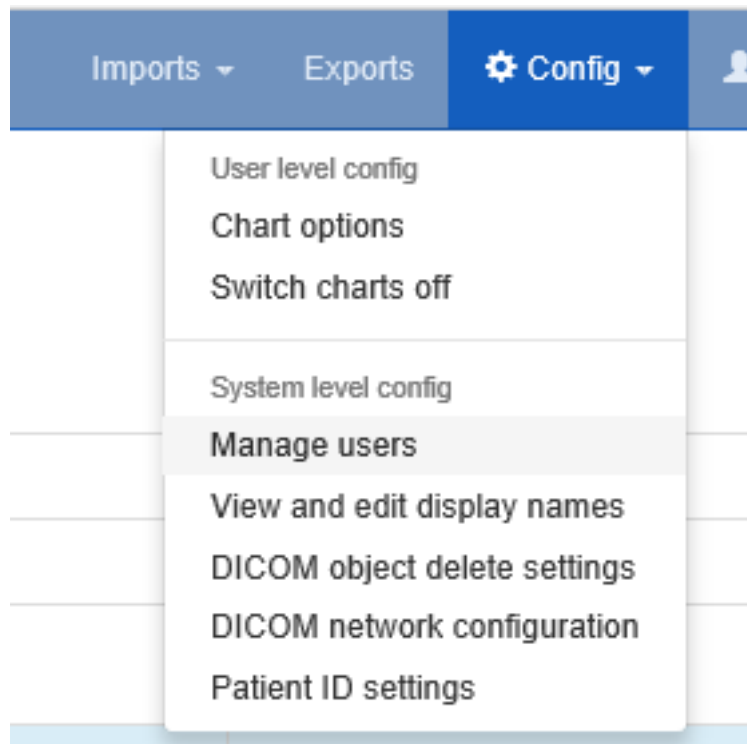
To stop Celery beat, just press `Ctrl+c`

4.4 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.

Imports ▾   Exports   ⚙ Config ▾   👤

User level config
Chart options
Switch charts off

System level config
Manage users
View and edit display names
DICOM object delete settings
DICOM network configuration
Patient ID settings

- After the first users are configured, this link will no longer be presented and instead you can go to `Config ->` `Manage users`.

- You will need the superuser username and password you created just after creating the database. The groups are

  - `viewgroup` can browse the data only

  - `importsizegroup` can use the csv import facility to add patient height and weight information

  - `importqrgroup` can use the DICOM query-retrieve facility to pull in studies, as long as they are pre-configured

  - `exportgroup` can view and export data to a spreadsheet

  - `pidgroup` can search using patient names and IDs depending on settings, and export with patient names and IDs if they are also a member of the `exportgroup`

  - `admingroup` can delete studies, configure DICOM Store/QR settings, configure DICOM keep or delete settings, configure patient ID settings, and abort and delete patient size import jobs. *Members of the admingroup no longer inherit the other groups permissions.*

- In addition to adding users to these groups, you may like to grant a second user `superuser` and `staff` status so that there are at least two people who can manage the users

- Return to the OpenREM interface (click on `View site` at the top right)

- 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 DICOM Store and QR (not yet up to date)

- 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 Viewing and editing individual x-ray system display names using the web interface for more information

4.5 Start using it!

Add some data!

```
openrem_rdsr.py rdsrfile.dcm
```

4.6 Further instructions

4.6.1 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. For now, please refer to the instructions and links at http://celery.readthedocs.org/en/latest/tutorials/daemonizing.html.

Configuration

See also: `local_settings.py` *Configuration*

5.1 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).

5.1.1 Configure what is deleted

Use the `Config` menu and select `DICOM object delete settings`:

This will open the configuration page:

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.

5.1.2 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:

More information about the DICOM network configuration can be found on the DICOM Store and QR page.

5.2 Viewing and editing individual x-ray system display names using the web interface
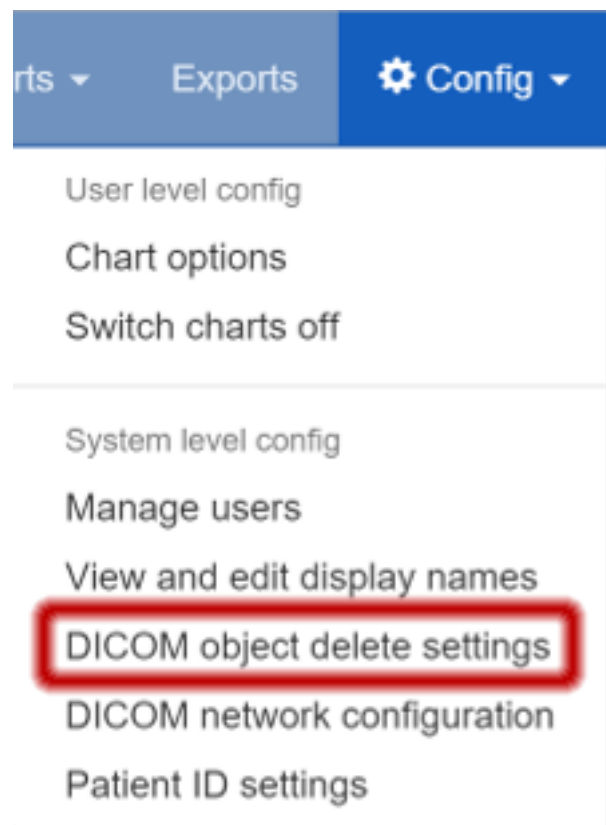
New in 0.7.0

Fig. 5.1: The `Config` menu

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?

☐ Delete radiography images after processing?

☐ Delete Philips CT dose info images after processing?

Submit

Fig. 5.2: Modify DICOM object deletion policy

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	Modify DICOM object delete settings
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. 5.3: Deletion policies can be reviewed on the DICOM network configuration page

Contents

5.2.1 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`.

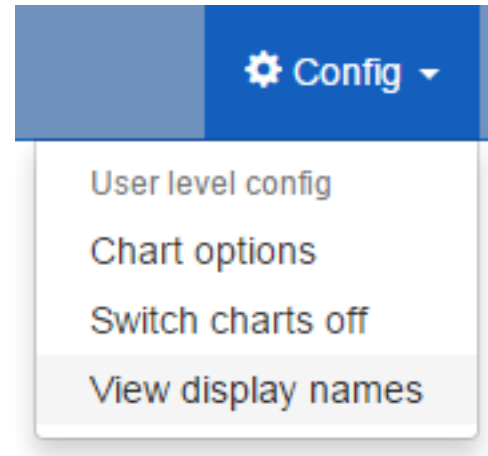### 5.2.2 Viewing x-ray system display names

Fig. 5.4: The `Config` menu (user)

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`, 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. The x-ray systems are grouped into modalities and displayed in five tables: CT; mammography; DX and CR; fluoroscopy; and other.

5.2.3 Changing x-ray system display names

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

This will take you to a page where you can view the list of x-ray systems with data in OpenREM. If you wish to change a display name then click on the corresponding row. The resulting page will allow you to edit the display name. Click on the `Update` button to confirm your changes:

You can set multiple rows to the same display name. 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.

| OpenREM | CT | Radiography | | | Imports ▾ | Exports | ⚙ Config ▾ | 👤 Welcome - logout |

Click on a row to edit the display name.

CT

There are 4 entries in this table.

Display name	Institution	Department	Manufacturer	Model	Station name	Serial no.	Software version	Gantry ID
CT scanner 1, Hospital A scanner 1	CT scanner 1, Hospital A	None	SIEMENS	SOMATOM Definition AS	scanner 1	64023	syngo CT 2012B	None
CT scanner 1, Hospital B scanner 1	CT scanner 1, Hospital B	None	TOSHIBA	Aquilion 64	scanner 1	64023	syngo CT 2012B	None
CT scanner 2, Hospital A scanner 2	CT scanner 2, Hospital A	None	SIEMENS	SOMATOM Definition 64	scanner 2	64023	syngo CT 2012B	None
CT scanner 2, Hospital B scanner 2	CT scanner 2, Hospital B	None	PHILIPS	Brilliance 64	scanner 2	64023	syngo CT 2012B	None

Mammography

There are 0 entries in this table.

Display name	Institution	Department	Manufacturer	Model	Station name	Serial no.	Software version	Gantry ID

DX and CR

There are 5 entries in this table.

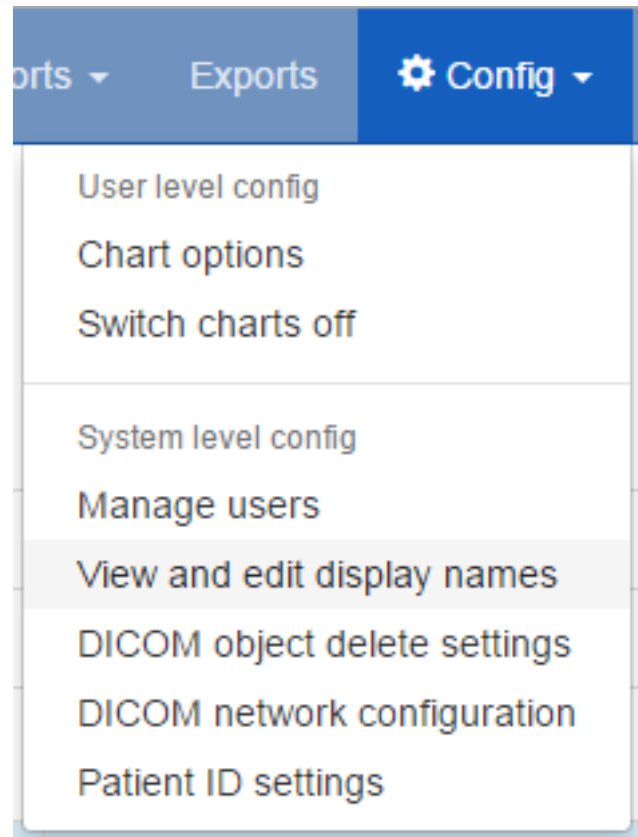| Display name	Institution	Department	Manufacturer	Model	Station name	Serial no.	Software version	Gantry ID
X-ray room A, Hospital B station 1	X-ray room A, Hospital B	None	Canon Inc.	CXDI	station 1	200170	V6.60.02	None
X-ray room B, Hospital B station 2	X-ray room B, Hospital B	None	Canon Inc.	CXDI	station 2	200170	V6.60.02	None
X-ray room C, Hospital A station 3	X-ray room C, Hospital A	None	Canon Inc.	CXDI	station 3	200170	V6.60.02	None
X-ray room C, Hospital B station 3	X-ray room C, Hospital B	None	Canon Inc.	CXDI	station 3	200170	V6.60.02	None

Fig. 5.5: Example list of display names

Fig. 5.6: The `Config` menu (admin)

Fig. 5.7: Example of the page for updating a display name

Importing data to OpenREM

6.1 Importing dose related data from DICOM files

If you are using linux, or for Windows if you have put `C:\Python27\;C:\Python27\Lib\site-packages;C:\Python27\` onto your system path, you should be able to import from the command line:

6.1.1 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
```

6.1.2 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

6.1.3 For radiographic DICOM images

```
openrem_dx.py filename.dcm
```

6.1.4 For CT dose summary files from Philips CT scanners

```
openrem_ctphilips.py filename.dcm
```

DICOM Store and QR

To make the most of OpenREM, you need a DICOM store node and maybe use DICOM query-retrieve too. The documents below walk you though how to set this up.

We are recommending that production installs make use of a third party provided DICOM store node as the in-built one has not yet proved reliable enough at this time - any help with testing and improvement would be most welcome! This is why we have provided documentation to work with conquest too.

7.1 DICOM Network Configuration

7.1.1 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 network configuration`, then click `Add new Store` and fill in the details (see figure 1):

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

7.1.2 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):

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.

7.1.3 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 and Running Conquest on Windows as a service for more information about using Conquest with OpenREM

7.1.4 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

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.

7.1.5 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 network configuration`, 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.

7.1.6 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, Modali
SOPClassUID Secondary Capture Image Storage, Study UID 1.2.840.113564.9.1.2843752344.47.2.5000947881
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
```

7.2 Conquest DICOM store node on Ubuntu

7.2.1 Installation

Ubuntu has reasonably up to date versions of the Conquest DICOM server in its repositories, so this makes installation very easy.

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
```

7.2.2 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).

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
```

7.2.3 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.

7.2.4 Configure Conquest to work with OpenREM

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

Bash scripts

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
```

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

```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
```

```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
```

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
# 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

# MG images
ImportModality2   = MG
ImportConverter2  = save to /var/lib/conquest-dicom-server/incoming/%o.dcm; system /etc/conquest-dico

# DX images
ImportModality3   = DX
ImportConverter3  = save to /var/lib/conquest-dicom-server/incoming/%o.dcm; system /etc/conquest-dico
# CR images
ImportModality4   = CR
ImportConverter4  = save to /var/lib/conquest-dicom-server/incoming/%o.dcm; system /etc/conquest-dico

# Philips CT
ImportConverter5  = ifequal "%V0008,0016","1.2.840.10008.5.1.4.1.1.7"; {save to /var/lib/conquest-dic

# Other objects
ImportConverter6  = destroy
```

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

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

7.3 DICOM Query Retrieve Service

To query retrieve dose related objects from a remote server, you need to review the DICOM Network Configuration first.

7.3.1 Query-retrieve using the web interface

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 fingure 2 for an example.

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).

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.

Advanced query options

- **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.

- **Ignore studies already in the database** *default ticked*: The RDSR import routine checks for the existance of the study UID in the database, and if it is found they it doesn't go any further. This might change in the future as there are instances where two RDSRs might legitimately have the same study UID, but different event UIDs. For image based imports, the individual events are checked, so if you think there is a reasonable chance that the database is missing individual images from a study, then you might like to deselect this setting. If the same dates are selected multiple times (to update during a day for example), activating this setting will result in the same exams all being transferred each time.

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 adminsitrator 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 `Retreive` 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.

7.3.2 Query-retrieve using the command line interface

In a command window/shell, `qrscu.py -h` should present you with the following output:

```
usage: qrscu.py [-h] [-ct] [-mg] [-fl] [-dx] [-f yyyy-mm-dd] [-t yyyy-mm-dd]
                [-e string] [-i string] [-sr] [-dup] qrid storeid

Query remote server and retrieve to OpenREM

positional arguments:
  qrid                  Database ID of the remote QR node
  storeid               Database ID of the local store node

optional arguments:
  -h, --help            show this help message and exit
  -ct                   Query for CT studies
```

```
-mg                    Query for mammography studies
-fl                    Query for fluoroscopy studies
-dx                    Query for planar X-ray studies
-f yyyy-mm-dd, --dfrom yyyy-mm-dd
                       Date from, format yyyy-mm-dd
-t yyyy-mm-dd, --duntil yyyy-mm-dd
                       Date until, format yyyy-mm-dd
-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
-sr                    Advanced: Query for structured report only studies
-dup                   Advanced: Retrieve studies that are already in database
```

As an example, if you wanted to query the PACS for DX images on the 5th 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:

```
qrscu.py 2 1 -dx -f 2010-04-05 -t 2010-04-05 -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:

```bash
#!/bin/bash

. /var/openrem/bin/activate  # activate virtualenv if you are using one, modify or delete this line

ONEHOURAGO=$(date -d "1 hour ago" "+%Y-%m-%d")

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 on Windows and run using the scheduler.

7.3.3 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):

```
qrscu.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 ['importe
[04/May/2016 11:30:03] INFO [remapp.netdicom.qrscu:267] Request association with Hospital PACS PACSAF
[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, starte
[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
[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
[04/May/2016 11:30:13] INFO [remapp.netdicom.qrscu:422] Now have 6 studies after deleting any contain
[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)
[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)
[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: open-rem_store.log*

The following instructions might also be useful with a Conquest setup, but they need review and updating:

7.4 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.

7.4.1 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.

7.4.2 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.

7.5 Configuring Conquest DICOM server to automatically forward data to OpenREM

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.

For example:

```
ImportModality1   = MG
ImportConverter1  = save to C:\conquest\dosedata\mammo\%o.dcm; system C:\conquest\openrem-mam-launch.
```

`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 has 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

# Import of CR images
ImportModality3   = CR
ImportConverter3  = save to C:\conquest\dosedata\dx\%o.dcm; system C:\conquest\openrem-dx-launch.bat

# Import of structured dose reports (this checks the DICOM tag 0008,0016 to see if it matches the val
ImportConverter4  = ifequal "%V0008,0016","1.2.840.10008.5.1.4.1.1.88.67"; {save to C:\conquest\dosed
```

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.

8.1 Configure what is stored

On the Config menu, select `Patient ID settings`. The initial settings are as follows:

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

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

8.2 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.
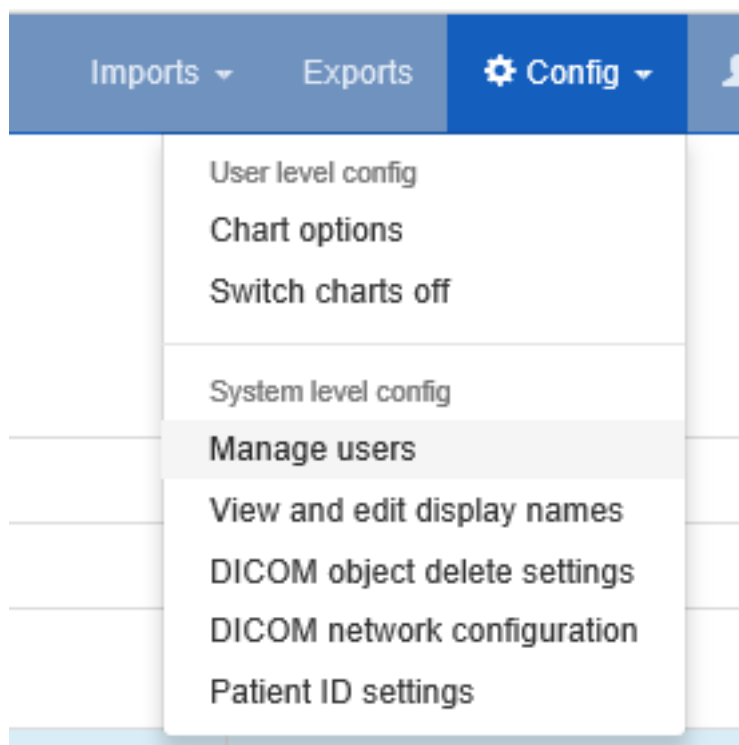
8.3 Using patient identifiable data

8.3.1 Querying for patient studies

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

Patient name:

Patient ID:

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

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

8.3.2 Study export with patient identifiers

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

Data export

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

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

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

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

Navigating, filtering and study details

9.1 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 or mammography studies. Alternatively, if you click on the station name link (in blue) you can filter to just that source modality.

New in 0.4.0: If you are not logged in, clicking any of the links will bring up the log in page.

9.2 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 model name includes the term 'soma':

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.

9.3 Viewing study details

By clicking on the study description link (in blue), you can see more details for an individual study:

Detail list of events

- Accession number: ab462362354
- Study date: 23 Jan 2013
- Study time: 1:17 p.m.
- Study description: Dual Energy^DE_TAP_IV (Adult)
- Requested procedure: CT Thorax abdomen and pelvis with contrast
- Patient age: 52.8
- Patient height and weight: 190 cm, 86 kg
- Hospital: Clinic B
- Scanner: SIEMENS | SOMATOM Definition Flash | CTAWP73491
- Study UID: 1.2.840.113564.9.1.27282345238.69.2.508347462734
- Comment:
- Test patient indicators? None

Acquisition protocol	Type	CTDIvol mGy	DLP mGy.cm	Scanning length (mm)	kVp	mA	Max mA	Exposure time per rotation (s)	Pitch	Exposure time (s)	Slice thickness (mm)	Collimation (mm)	X-ray modulation type
Topogram	Constant Angle Acquisition	0.14	11.26	803	120	35	35		None	8.190	0.600	3.60	OFF
Comment Internal technical scan parameters: Organ Characteristic = Thorax, Body Size = Adult, Body Region = Body, X-ray Modulation Type = OFF													
Topogram	Constant Angle Acquisition	0.14	11.54	824	120	35	35		None	8.400	0.600	3.60	OFF
Comment Internal technical scan parameters: Organ Characteristic = Thorax, Body Size = Adult, Body Region = Body, X-ray Modulation Type = OFF													
PreMonitoring	Stationary Acquisition	1.82	1.82	10	120	59	60	0.500	None	0.500	10.000	10.00	OFF
Comment Internal technical scan parameters: Organ Characteristic = Abdomen, Body Size = Adult, Body Region = Body, X-ray Modulation Type = OFF													
Monitoring	Stationary Acquisition	7.27	7.27	10	120	59	60	0.500	None	2.000	10.000	10.00	OFF
Comment Internal technical scan parameters: Organ Characteristic = Abdomen, Body Size = Adult, Body Region = Body, X-ray Modulation Type = OFF													
DE_TAP	Spiral Acquisition	8.11	630.63	797	100	165	430	0.500	0.8000	26.050	0.600	19.20	XYZ_EC
					140	131	307	0.500					
Comment Internal technical scan parameters: Organ Characteristic = Abdomen, Body Size = Adult, Body Region = Body, X-ray Modulation Type = XYZ_EC, Sn Filter (Tube B) = yes													

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.

Charts

10.1 Chart types

10.1.1 1. Bar chart of average values across a number of categories

An example of mean DAP per acquisition type is shown in figure 1.

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.

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 series. 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.

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 5).

The histogram data can be plotted as absolute values, or be normalised to a value of 1.0 (figure 6). 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.

Clicking on a legend entry toggles the visibility of the corresponding series.

10.1.2 2. Pie chart showing the frequency of each item in a category

Figure 7 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.

10.1.3 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 8.

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.

10.1.4 4. Pie chart showing the number of events 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 9). Clicking on one of the segments will take you to a pie chart that shows the number of events per on that day (figure 10).

10.1.5 5. Scatter plot showing one value vs. another

This plot type shows a data point per event (figure 11). 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 12). 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.

Clicking on a system's legend entry toggles the display of the corresponding series on the chart.

10.2 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.

10.3 New in 0.7.0

- Charts for fluoroscopy and mammography.
- New scatter plot chart type.
- Chart plotting options can be configured by choosing `Chart options` from the `User options` menu at the top of the OpenREM homepage.
- Chart average values can be set to either mean or median. Bar charts can be configured to plot both mean and median values as seperate series.

- Bar charts can be plotted with a series per x-ray system. This option can be switched on or off via the `Chart options`.

- The number of histogram data bins can be set to a value between 2 and 40 in `Chart options`. The default value is 20.

- Histogram calculation can be switched on or off in `Chart options`. The default is off. Performance is significantly better when set to off.

- Histogram plots can toggle between absolute or normalised values via the `Toggle Normalised histograms` button that is visible when viewing a histogram plot.

- The data in the bar charts can be sorted interactively by clicking on the sorting options below the individual chart. The default sorting type and direction can be set by choosing the `Chart options` item from the `User options` menu on the OpenREM homepage.

- Individual charts can be displayed full-screen by clicking on the `Toggle fullscreen` button that is positioned below each chart.

- The chart plotting status is displayed on the OpenREM homepage.

- The colours used for plotting have been updated.

10.4 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. 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.

10.5 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 $CTDI_{vol}$ 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

10.6 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 DAP for each requested procedure
- Pie chart of the frequency of 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

10.7 Chart types - fluoroscopy

- Bar chart of average DAP for each study description
- Pie chart of the frequency of each study description
- Pie chart showing the number of studies carried out per weekday

10.8 Chart types - mammography

- Scatter plot of average glandular dose vs. compressed thickness for each acquisition
- Pie chart showing the number of studies carried out per weekday

10.9 Performance notes

10.9.1 All chart types

For any study- or request-based charts, filtering using *Acquisition protocol* forces OpenREM to use a much slower method of querying the database for chart data. Where possible avoid filtering using this field, especially when viewing a large amount of data.

10.9.2 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.

10.9.3 Scatter plots

Switching off *Plot a series per system* in the *Chart options* will speed up data calculation.

Calculation and display of 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. It is hoped that they will be included in version 0.7.1.

- Users can continue to export OpenREM studies in a format that can be used with a stand-alone version of openSkin.

11.1 Functionality that will be available

- Skin dose map data calculated to the surface of a simple geometric phantom using the in-built openSkin routines (3D phantom)

- Phantom dimensions calculated from the height and mass of the patient

- Data can be calculated on import to OpenREM, or on demand when a study is viewed

- 3D skin dose map data shown graphically as a 2D image and a 3D model

- 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 and patient height and mass 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). A default height of 1.786 m and mass of 73.2 kg are used if patient-specific data are unavailable.

11.1.1 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. 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

- The current view can be saved as a png file

11.1.2 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

- 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

11.2 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.

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

11.3 Exporting data to openSkin

If skin dose maps are disabled the user you are presented with the option of exporting the study data as a csv file for use with a stand-alone installation of openSkin. Select the fluoroscopy study you wish to create the exposure incidence map for and go to the detail view. Then click on the link to create the OpenSkin export (figure 5).

11.4 Instructions for openSkin

Download the latest version as a zip file from openSkin downloads. At the time of release for OpenREM 0.7.0 the current openSkin beta was dated 14th September 2016. The application referred to here will only work on Windows.

- Extract the contents of the zip file into a folder on your computer and run the openSkin.exe executable

- Choose a phantom type: 3D or flat. See phantom design for details

- Select the source csv file - this should be the one exported from OpenREM

- Select the output folder - this should already exist as it can't be created in the dialogue

- Wait! Depending on the number of events in the export and the power of your machine, this 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`

11.4.1 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.

11.4.2 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.
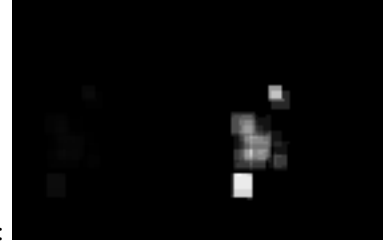
A native and 'colour equalised' version of the same export are shown below:

11.5 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.

Exporting study information

12.1 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. The following exports are currently available (version 0.5.0)

- CT basic, single sheet csv

- CT advanced, XLSX muliple-sheets

- Fluoroscopy basic, single sheet csv

- Mammography, single sheet csv

- Mammography NHSBSP, single sheet csv designed to satisfy NHSPSB reporting

- Radiographic, single sheet csv

- Radiographic, XLSX multiple sheets

For CT and radiographic, the XLSX export has multiple sheets. The first sheet contains a summary of all the study descriptions, requested procedures and series protocol names contained in the export:

This information is useful for seeing what data is in the spreadsheet, and can also be used to prioritise which studies or protocols to analyse based on frequency.

The second sheet of the exported file lists all the studies, with each study taking one line and each series in the study displayed in the columns to the right.

The remainder of the file has one sheet per series protocol name. Each series is listed one per line. If a single study has more than one series with the same protocol name, then the same study will appear on more than one line.

Clicking the link for an export redirects you to the Exports page, which you can also get to using the link at the top right of the navigation bar:

Whilst an export is being processed, it will be listed in the first table at the top. The current status is displayed to indicate export progress. If an export gets stuck for whatever reason, you may be able to abort the process by clicking the 'Abort' button. However this does not always cause an active export to terminate - you may find it completes anyway!

Completed exports are then listed in the second table, with a link to download the csv or xlsx 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

> **Warning:** Large exports have been killed by the operating system due to running out of memory - a 6500 CT exam xlsx export was killed after 3400 studies for example. This issue is being tracked as #116 and will hopefully be addressed in the next release. It is possible that if debug mode is turned off then memory will be managed better, but I also need to modify the xlsx export to make use of the memory optimisation mode in xlsxwriter.

OpenREM administration

Contents:

13.1 Deleting studies

New in 0.4.0

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^TAP120 kvIV (Adult) | F 1	4	1257.10	Delete
TOM CTAWP1234	2013-05-23 11:05	Thorax^TA_IV120 kV (Adult) | F 1	4	314.26	Delete
TOM	2013-05-23	Thorax^TAP120 kvIV (Adult) |	4	688.99	Delete

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

13.2 Adding patient size information from csv using the web interface

Contents

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

13.2.1 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:
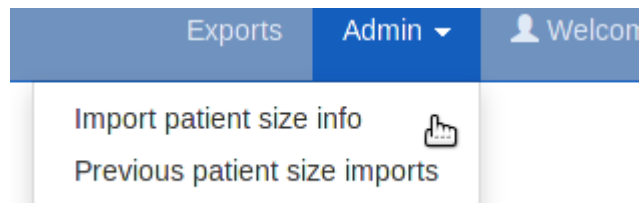
The first option takes you to a page where you can upload a csv file containing details of the patient height and weight, plus either the accession number or the Study Instance UID.

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.

13.2.2 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

BIRTH_DATE_D
ACCESSION_NUMBER
HEIGHT
WEIGHT
START_DATETIME_D
START_DATETIME_T
PACS_SPS_ID
DESCRIPTION
TOTAL_DOSE
CTDIVOL
DLP_SERIES
DLP_TOTAL
CT_PROTOCOL
PATIENT_ID

Process the data

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 of 59183	Abort

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.

13.2.3 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:

OpenREM   CT   Fluoroscopy   Mammography   Exports   Admin ▾   👤 Welcome Ed Ad

Import patient size info
Previous patient size imports

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
```

13.2.4 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.

13.3 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.

Troubleshooting

14.1 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.`
- Other linux: `/usr/lib/python2.7/site-packages/openrem/openremproject/local_settings.py`
- Linux virtualenv: `lib/python2.7/site-packages/openrem/openremproject/local_settings.py`
- Windows: `C:\Python27\Lib\site-packages\openrem\openremproject\local_settings.py`
- Windows virtualenv: `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.

14.2 Query-retrieve issues

Refer to the *Troubleshooting: openrem_qr.log* documentation

14.3 OpenREM DICOM storage nodes

Refer to the *Troubleshooting: openrem_store.log* documentation

14.4 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.

Documentation for the OpenREM code

Contents:

15.1 DICOM import modules

15.1.1 RDSR module

Ultimately this should be the only module required as it deals with all Radiation Dose Structured Reports. Currently this has only been tested on CT and fluoroscopy structured reports, but it also has the logic for mammography structured reports if they start to appear.

15.1.2 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.

15.1.3 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.

15.1.4 CT non-standard modules

Initially only Philips CT dose report images are catered for. These have all the information that could be derived from the images also held in the DICOM header information, making harvesting relatively easy.

15.2 Non-DICOM import modules

15.2.1 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.

15.3 Export from database

15.3.1 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.xlsx.**ctxlsx**

15.3.2 Single sheet CSV exports

Specialised csv exports - NHSBSP formatted mammography export

15.4 Tools and helper modules

15.4.1 OpenREM settings

Administrative module to define the name of the project and to add it to the Python path

15.4.2 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**
>>> • **keyword** (*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 val: database field :return: value or None

15.4.3 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

15.4.4 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*)
> Given one or two dates of the form yyyy-mm-dd, return a DICOM date range :param: date1, date2: One or two
> yyyy-mm-dd dates :type date1, date2: String :returns: DICOM date range as string

15.4.5 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.
>> **Parameters dataset** (*dataset*) – The DICOM dataset.
>> **Returns** str. – xml style string if any trigger values are found.

15.5 Models

15.6 Filtering code

15.7 Views

15.8 Export Views

15.9 Forms

15.10 DICOM networking modules

15.10.1 Query-retrieve module

Descriptive text

15.11 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.

15.11.1 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`

15.11.2 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:

```python
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`

15.11.3 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.

```python
# 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:

```python
# 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['plotRFStudyPerDayAnd
        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:

```python
return_structure =\
    rf_plot_calculations(f, request_results, median_available, user_profile.plotAverageChoice,
                         user_profile.plotSeriesPerSystem, user_profile.plotHistogramBins,
                         user_profile.plotHistograms)
```

After:

```python
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_s
                         plot_histogram_bins, plot_histograms):
```

After:

```
def rf_plot_calculations(f, request_results, median_available, plot_average_choice, plot_series_per_s
                         plot_histogram_bins, plot_study_per_day_and_hour, plot_study_freq, plot_hist
```

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_
                                              'study_description',
                                              'projectionxrayradiationdose__accumxraydose__accuminteg
                                              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:

```python
from remapp.forms import GeneralChartOptionsDisplayForm, DXChartOptionsDisplayForm, CTChartOptionsDis
```

After:

```python
from remapp.forms import GeneralChartOptionsDisplayForm, DXChartOptionsDisplayForm, CTChartOptionsDis
    RFChartOptionsDisplayForm
```

The RF form items need to be included

Before (abbreviated):

```python
if request.method == 'POST':
    general_form = GeneralChartOptionsDisplayForm(request.POST)
    ct_form = CTChartOptionsDisplayForm(request.POST)
    dx_form = DXChartOptionsDisplayForm(request.POST)
    if general_form.is_valid() and ct_form.is_valid() and dx_form.is_valid() and rf_form.is_valid():
        try:
            # See if the user has plot settings in userprofile
            user_profile = request.user.userprofile
        except:
            # Create a default userprofile for the user if one doesn't exist
            create_user_profile(sender=request.user, instance=request.user, created=True)
            user_profile = request.user.userprofile

        user_profile.plotCharts = general_form.cleaned_data['plotCharts']
        ...
        ...
        user_profile.plotHistogramBins = general_form.cleaned_data['plotHistogramBins']

        user_profile.plotCTAcquisitionMeanDLP = ct_form.cleaned_data['plotCTAcquisitionMeanDLP']
        ...
        ...
        user_profile.plotCTInitialSortingChoice = ct_form.cleaned_data['plotCTInitialSortingChoice']

        user_profile.plotDXAcquisitionMeanDAP = dx_form.cleaned_data['plotDXAcquisitionMeanDAP']
        ...
        ...
        user_profile.plotDXInitialSortingChoice = dx_form.cleaned_data['plotDXInitialSortingChoice']

        user_profile.save()

    messages.success(request, "Chart options have been updated")

...
...

general_form_data = {'plotCharts': user_profile.plotCharts,
                     'plotMeanMedianOrBoth': user_profile.plotAverageChoice,
                     'plotInitialSortingDirection': user_profile.plotInitialSortingDirection,
                     'plotSeriesPerSystem': user_profile.plotSeriesPerSystem,
                     'plotHistogramBins': user_profile.plotHistogramBins}

ct_form_data = {'plotCTAcquisitionMeanDLP': user_profile.plotCTAcquisitionMeanDLP,
                ...
                ...
                'plotCTInitialSortingChoice': user_profile.plotCTInitialSortingChoice}
```

```
dx_form_data = {'plotDXAcquisitionMeanDAP': user_profile.plotDXAcquisitionMeanDAP,
                ...
                ...
                'plotDXInitialSortingChoice': user_profile.plotDXInitialSortingChoice}


general_chart_options_form = GeneralChartOptionsDisplayForm(general_form_data)
ct_chart_options_form = CTChartOptionsDisplayForm(ct_form_data)
dx_chart_options_form = DXChartOptionsDisplayForm(dx_form_data)

return_structure = {'admin': admin,
                    'GeneralChartOptionsForm': general_chart_options_form,
                    'CTChartOptionsForm': ct_chart_options_form,
                    'DXChartOptionsForm': dx_chart_options_form
                    }
```

After (abbreviated):

```python
if request.method == 'POST':
    general_form = GeneralChartOptionsDisplayForm(request.POST)
    ct_form = CTChartOptionsDisplayForm(request.POST)
    dx_form = DXChartOptionsDisplayForm(request.POST)
    rf_form = RFChartOptionsDisplayForm(request.POST)
    if general_form.is_valid() and ct_form.is_valid() and dx_form.is_valid() and rf_form.is_valid():
        try:
            # See if the user has plot settings in userprofile
            user_profile = request.user.userprofile
        except:
            # Create a default userprofile for the user if one doesn't exist
            create_user_profile(sender=request.user, instance=request.user, created=True)
            user_profile = request.user.userprofile

        user_profile.plotCharts = general_form.cleaned_data['plotCharts']
        ...
        ...
        user_profile.plotHistogramBins = general_form.cleaned_data['plotHistogramBins']

        user_profile.plotCTAcquisitionMeanDLP = ct_form.cleaned_data['plotCTAcquisitionMeanDLP']
        ...
        ...
        user_profile.plotCTInitialSortingChoice = ct_form.cleaned_data['plotCTInitialSortingChoice']

        user_profile.plotDXAcquisitionMeanDAP = dx_form.cleaned_data['plotDXAcquisitionMeanDAP']
        ...
        ...
        user_profile.plotDXInitialSortingChoice = dx_form.cleaned_data['plotDXInitialSortingChoice']

        user_profile.plotRFStudyPerDayAndHour = rf_form.cleaned_data['plotRFStudyPerDayAndHour']
        user_profile.plotRFStudyFreq = rf_form.cleaned_data['plotRFStudyFreq']

        user_profile.save()

    messages.success(request, "Chart options have been updated")

...
...

general_form_data = {'plotCharts': user_profile.plotCharts,
```

```
                          ...
                          ...
                          'plotHistogramBins': user_profile.plotHistogramBins}

ct_form_data = {'plotCTAcquisitionMeanDLP': user_profile.plotCTAcquisitionMeanDLP,
                ...
                ...
                'plotCTInitialSortingChoice': user_profile.plotCTInitialSortingChoice}

dx_form_data = {'plotDXAcquisitionMeanDAP': user_profile.plotDXAcquisitionMeanDAP,
                ...
                ...
                'plotDXInitialSortingChoice': user_profile.plotDXInitialSortingChoice}

rf_form_data = {'plotDXStudyPerDayAndHour': user_profile.plotDXStudyPerDayAndHour,
                'plotRFStudyFreq': user_profile.plotRFStudyFreq}

general_chart_options_form = GeneralChartOptionsDisplayForm(general_form_data)
ct_chart_options_form = CTChartOptionsDisplayForm(ct_form_data)
dx_chart_options_form = DXChartOptionsDisplayForm(dx_form_data)
rf_chart_options_form = RFChartOptionsDisplayForm(rf_form_data)

return_structure = {'admin': admin,
                    'GeneralChartOptionsForm': general_chart_options_form,
                    'CTChartOptionsForm': ct_chart_options_form,
                    'DXChartOptionsForm': dx_chart_options_form,
                    'RFChartOptionsForm': rf_chart_options_form,
                    }
```

15.11.4 Additions to `displaychartoptions.html`

A new div needs to be added for the fluoroscopy chart options:

```html
<div class="panel-heading">
  <h3 class="panel-title">Fluoroscopy chart options</h3>
</div>
<div class="panel-body">
  <table>
    {% csrf_token %}
    {{ RFChartOptionsForm }}
  </table>
  <input class="btn btn-default" name="submit" type="submit" />
</div>
```

15.11.5 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_de
        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="#collapseStudyWorkloadP
                        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 f
                    <a onclick="enterFullScreen('collapseStudyWorkloadPieChart', 'piechartStudyWorklo
                </div>
            </div>
        </div>
    </div>
    <!-- End of HTML to include div container for studies per week day pie chart -->
{% endif %}

{% if request.user.userprofile.plotRFStudyFreq %}
    <!-- HTML to include div container for study name pie chart -->

    <script>
        $(window).resize(function() {
            chartSetExportSize('piechartStudyDIV');
            fitChartToDiv('piechartStudyDIV');
        });
    </script>

    <div class="panel-group" id="accordionPiechartStudy">
        <div class="panel panel-default">
            <div class="panel-heading">
                <h4 class="panel-title">
                    <a data-toggle="collapse" data-parent="#accordionPiechartStudy" href="#collapseSt
```

```
                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='
        </div>
    </div>
</div>
</div>
<!-- End of HTML to include div container for study name pie chart -->
{% endif %}
```

15.11.6 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,
}
```

That's it - you should now have two new charts visible in the fluoroscopy filtered page.

15.12 Indices and tables

- genindex
- modindex
- search

Previous Release Notes and Change Log

16.1 Version history change log

16.1.1 OpenREM version history

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 pynet-dicom

- #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 http://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

16.2 Release notes and upgrade instructions

Each release comes with specific upgrade instructions, so please follow the links below for the appropriate version.

16.2.1 Version specific information

Release Notes v0.6.0

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 trun-
  cated 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 -
      `numpy1.9.2+mklcp27nonewin32.whl` is likely to be the right ver-
      sion, unless you have 64bit Python installed, in which case use the
      `numpy1.9.2+mklcp27nonewin_amd64.whl` version instead.

    - Install using pip:

```
pip install numpy1.9.2+mklcp27nonewin32.whl
```

Install pynetdicom

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

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 `backupRestorePostgreSQL`

    - 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 re-
  quired:

```
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 DICOM Store and QR.

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 `backupRestorePostgreSQL`

  - 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
# 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
# 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,
? 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 migrati
? 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 `backupRestorePostgreSQL`

    - 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
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
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 `backupRestorePostgreSQL`

    - 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 --a
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 --a
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 remap
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 `backupRestorePostgreSQL`

    - 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 rem
                 # 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 rem
                 # 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\

# 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,
  ? 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 migrati
  ? 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

16.3 Contributing authors

Many people have contributed to OpenREM - either with code, documentation, bugs, examples or ideas, including:

* Elly Castellano
* Jonathan Cole
* Daniel Gordon
* Hamid Khosravi
* Laurence King
* Eivind Larsen
* John Loveland
* Ed McDonagh
* David Platten
* Erik-Jan Rijkhorst

Indices and tables

- genindex
- modindex
- search

c

d

g

n

r

C

D

G

M

N

R