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:
Installation instructions

1.1 Quick setup

Note: Most path names are represented using the linux convention of a / separator. If you are installing in a Windows environment you will need to use the Windows \ separator.

1. Install python (might need to be 2.7)
2. Install setuptools and pip
3. Install OpenREM
   • pip install OpenREM-ver.tar.gz
4. Configure OpenREM
   • Locate install location, typically something/lib/python2.7/site-packages/openrem
   • There are two files that need renaming:
     - openrem/openrem/settings.py.example to openrem/openrem/settings.py
     - openrem/openrem/wsgi.py.example to openrem/openrem/wsgi.py
   • in the settings.py file, set the database details.
     • For testing ONLY, use
       - ‘ENGINE’: ‘django.db.backends.sqlite3’,
       - ‘NAME’: ‘/ENTER/PATH/WHERE/DB/FILE/CAN/GO’
         * Windows example: ‘NAME’: ‘C:\Documents\User\OpenREM\openrem.db’
         * Linux example: ‘NAME’: ‘/var/openrem/openrem.db’
5. Create the database
   • python path/to/openrem/manage.py syncdb
   • (optional for a test database) python path/to/openrem/manage.py convert_to_south remapp
6. Start test web server
   • python path/to/openrem/manage.py runserver
• If you are using a headless server and need to be able to see the web interface from another machine, use `python path/to/openrem/manage.py runserver x.x.x.x:8000` replacing the ‘x’ with the IP address of the server and ‘8000’ with the port you wish to use.

7. Open the web address given, appending `/openrem` (http://localhost:8000/openrem)

8. Add some data!
   • `openrem_rdsr.py` `rdsrfile.dcm`

### 1.2 More in depth process

1. **Install virtualenv or maybe virtualenvwrapper** Recommended if the server is ever going to be used for more than one python application – virtualenv sets up an isolated python environment

2. **Install OpenREM** As per the Quick setup instructions above. Don’t configure OpenREM yet

3. **Install a production database** SQLite is great for getting things running quickly and testing if the setup works, but is really not recommended for production use on any scale. Therefore it is recommended to use a different database such as PostgreSQL or MySQL.

   Here are the authors instructions for installing PostgreSQL on linux:

#### 1.2.1 Installing PostgreSQL for OpenREM on Ubuntu linux

**Install PostgreSQL and the python connector**

```bash
• sudo apt-get install postgresql
• sudo apt-get build-dep python-psycopg2
```

The second command installed a lot of things, at least some of which are necessary for this to work!

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

```bash
• pip install psycopg2
```

**Create a user for the database**

```bash
• sudo passwd postgres
• Enter password, twice
• sudo -u postgres createuser -P openrem_user
• Enter password, twice
• Superuser, No
• Create databases, No
• Create new roles, No
```
Optional: Specify the location for the database

You might like to do this if you want to put the database on an encrypted location

For this example, I’m going to assume all the OpenREM programs and data are in the folder /var/openrem/:

- sudo /etc/init.d/postgresql stop
- mkdir /var/openrem/database
- sudo cp -aRv /var/lib/postgresql/9.1/main /var/openrem/database/
- sudo nano /etc/postgresql/9.1/main/postgresql.conf

Change the line

- data_directory = '/var/lib/postgresql/9.1/main' to
- data_directory = '/var/openrem/database/main'
- sudo /etc/init.d/postgresql start

Create the database

- su postgres
- psql template1
- CREATE DATABASE openrem_db OWNER openrem_user ENCODING 'UTF8';
- \q
- exit

Change the security configuration

The default security settings are too restrictive to allow access to the database.

- sudo nano /etc/postgresql/9.1/main/pg_hba.conf
- Add the following line:
  - local openrem_db openrem_user md5
- sudo /etc/init.d/postgresql restart

Configure OpenREM to use the database

Find and edit the settings file, eg

- nano local/lib/python2.7/site-packages/openrem/openrem/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’,
**Fire it up with OpenREM**

- python path/to/openrem/manage.py syncdb
- python path/to/openrem/manage.py convert_to_south remapp

4. **Install and configure a production webserver**  
   Unlike the database, the production webserver can be left till later and can be changed again at any time.

   However, for performance it is recommended that a production webserver is used. Popular choices would be either Apache or you can do as the cool kids do and use Gunicorn with nginx.

5. **Configure OpenREM**  
   Follow the ‘Configure OpenREM’ instructions in the Quick setup section above, but this time with the production database details.

   Configure the production webserver too.

6. **Create the database**

- python path/to/openrem/manage.py syncdb

7. **Convert the database to use South**  
   South is a django application to manage database migrations. Using South means that future changes to the database model can be calculated and executed automatically with simple commands when OpenREM is upgraded.

- python path/to/openrem/manage.py convert_to_south remapp

### 1.3 Related guides

#### 1.3.1 Running Conquest on Windows as a service

*Note:*

This guide assumes Conquest has already been installed and runs ok. These instructions are based on Windows XP. **Run as a service**

1. Make sure conquest isn’t running.
2. Open a file browser and navigate to your conquest folder.
3. Right-hand click on the “ConquestDICOMServer.exe” file and choose “Run as...”
4. Enter the username and password of a Windows user with administrator rights.
5. Once conquest is running, click on the “Install server as NT service” on the “Configuration” tab.
6. Close the conquest Window.
7. Log in to Windows as a user with administrator rights.
8. Go to “Control panel” -> “Administrative Tools” -> “Services”.
9. There will be a service with the same name as conquest’s AE title. Right-hand click the mouse on this service and select “Properties”.
10. On the “Log On” tab check the box that says “Allow service to interact with the desktop”.
11. Click “Apply” then “OK”.

---

*Chapter 1. Installation instructions*
12. Right-hand click on the service again and click “Restart”.

The “Allow service to interact with the desktop” seems to be necessary for the batch to run that puts the dose report into OpenREM.

**Firewall settings**

Windows is able to change its firewall settings after you think everything is working ok! Assuming you have control of the firewall, add three port exceptions to the Windows firewall on the server computer: ports 80 and 443 for Apache, and whichever port that was chosen for conquest (104 is the port allocated to DICOM, but you may have used a higher port above 1024 for permissions reasons).

The firewall instructions at [portforward.com](http://portforward.com) were found to be a useful guide for this.

### 1.3.2 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:

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

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

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

Grant the `backup` user privileges to create databases:

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

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

```sql
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):

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

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

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

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

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

An example DOS batch file to back up the contents of the `openremdatabasemysql` database using a time stamp of the form `yyyy-mm-dd_hhmm`, zip up the resulting file, delete the uncompressed version and then copy it to a network location (the network copy will only work if the user that runs the batch file has permission on the network):
@echo off
For /f "tokens=1-4 delims=/ " %%a in ('date /t') do (set mydate=%c-%b-%a)
For /f "tokens=1-2 delims=/:" %%a in ('time /t') do (set mytime=%a%%b)

"C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqldump.exe" -ubackup -pbackup openremdatabasemysql > F:\OpenREMdatabase\backup\%mydate%_%mytime%_openremdatabasemysql.sql
"C:\Program Files\7-Zip\7z.exe" a F:\OpenREMdatabase\backup\%mydate%_%mytime%_openremdatabasemysql.zip F:\OpenREMdatabase\backup\%mydate%_%mytime%_openremdatabasemysql.sql

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

copy F:\OpenREMdatabase\backup\%mydate%_%mytime%_openremdatabasemysql.zip "\Srv-mps-001\xls_protect\PATDOSE\OpenREMbackup\"
2.1 Importing dose related data from DICOM files

If you are using linux or have put "C:\Python27\Lib\site-packages\openrem\scripts" and "C:\Python27\Lib\site-packages\openrem" onto your path, you should be able to import from the command line:

- For Radiation Dose Structured Reports
  - openrem_rdsr.py filename.dcm
- For mammography DICOM images (see restrictions on testing)
  - openrem_mg.py filename.dcm
- For CT dose summary files from Philips CT scanners
  - openrem_ctphilips.py filename.dcm
CHAPTER 3

Using the OpenREM interface and export function

3.1 Navigating the OpenREM web interface

placeholder

3.2 Filtering for specific studies

placeholder

3.3 Exporting to csv and xlsx sheets

placeholder
Upgrade instructions

4.1 Upgrading from tar.gz package

4.1.1 Code upgrade

1. `pip install OpenREM-version.tar.gz`

4.1.2 Database migration

Always do a database migration using South after an upgrade in case any of the database models have changed. This will normally not be the case.

1. `python path/to/openrem/manage.py schemamigration --auto remapp`

   *Example:* `python lib/python2.7/site-packages/openrem/manage.py schemamigration --auto remapp`

2. If response to the last command is ‘Nothing seems to have changed’, no migration is required. Else, follow the instructions to migrate.

3. Restart the web server, in case changes have been made to the web interface.
CHAPTER 5

Documentation for the OpenREM code

Contents:

5.1 DICOM import modules

5.1.1 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.

`remapp.extractors.rdsr.rdsr(rdsr_file)`

Extract radiation dose related data from DICOM Radiation SR objects.

**Parameters** filename *(str)* – relative or absolute path to Radiation Dose Structured Report.

**Tested with:**
- CT: Siemens, Philips and GE RDSR, GE Enhanced SR.
- Fluoro: Siemens Artis Zee RDSR

5.1.2 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.

`remapp.extractors.mam.mam(mg_file)`

Extract radiation dose structured report related data from mammography images

**Parameters** filename *(str)* – relative or absolute path to mammography DICOM image file.

**Tested with:**
- GE Senographe DS software versions ADS_43.10.1 and ADS_53.10.10 only.
5.1.3 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.

```
remapp.extractors.ct_philips.ct_philips(philips_file)

Extract radiation dose structured report related data from Philips CT dose report images
```

Parameters:
- `filename (str)`: relative or absolute path to Philips CT dose report DICOM image file.

Tested with:
- Philips Gemini TF PET-CT v2.3.0
- Brilliance BigBore v3.5.4.17001.

5.2 Non-DICOM import modules

5.2.1 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.

```
remapp.extractors.ptsizecsv2db.csv2db(*args, **kwargs)

Import patient height and weight data from csv RIS exports. Can be called from openrem_ptsizecsv.py script
```

Parameters:
- `--si-uid (bool)`: Use Study Instance UID instead of Accession Number. Short form `-s`.
- `csvfile (str)`: relative or absolute path to csv file
- `id (str)`: Accession number column header or header if `-u` or `--si-uid` is set. Quote if necessary.
- `height (str)`: Patient height column header. Create if necessary, quote if necessary.
- `weight (str)`: Patient weight column header. Create if necessary, quote if necessary.

Example:
```
openrem_ptsizecsv.py -s MyRISExport.csv StudyInstanceUID HEIGHT weight
```

5.3 Export from database

5.3.1 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.
5.3.2 Single sheet CSV exports

remapp.exports.exportcsv.exportFL2excel(request)
Export filtered fluoro database data to a single-sheet CSV file.

Parameters request (HTTP get) – Query parameters from the fluoro filtered page URL.

remapp.exports.exportcsv.exportCT2excel(request)
Export filtered CT database data to a single-sheet CSV file.

Parameters request (HTTP get) – Query parameters from the CT filtered page URL.

remapp.exports.exportcsv.exportMG2excel(request)
Export filtered mammography database data to a single-sheet CSV file.

Parameters request (HTTP get) – Query parameters from the mammo filtered page URL.

5.4 Tools and helper modules

5.4.1 OpenREM settings

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

remapp.extractors.openrem_settings.name_of_project()
Returns the name of the project. Default OpenREM

remapp.extractors.openrem_settings.add_project_to_path()
Add project to path, assuming this file is within project

5.4.2 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 Content_item_descriptions 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  Content_item_descriptions entry for code value passed

5.4.3 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

5.4.4 DICOM time and date values

Module to convert between 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

```python
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

```python
remapp.tools.dcmdatetime.make_date(dicomdate)
```

Given a DICOM date, return a Python date.

Parameters

- **dicomdate** (*str.*) – DICOM style date.

**Returns**

Python date value

```python
remapp.tools.dcmdatetime.make_time(dicomtime)
```

Given a DICOM time, return a Python time.

Parameters

- **dicomtime** (*str.*) – DICOM style time.

**Returns**

Python time value

```python
remapp.tools.dcmdatetime.make_date_time(dicomdatetime)
```

Given a DICOM date time, return a Python date time.

Parameters

- **dicomdatetime** (*str.*) – DICOM style date time.

**Returns**

Python date time value

### 5.4.5 Test for QA or other non-patient related studies

```python
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.

### 5.5 Models

### 5.6 Filtering code

### 5.7 Indices and tables

- `genindex`
Note: OpenREM does not currently include a DICOM Store SCP, ie you will need to install a DICOM store server in order to send RSDRs or other DICOM files from modalities.

If you have no preference, Conquest is recommended as a free, open source, scriptable DICOM server. Please note though that you will need to include the RSRD SOP in the dgatesop.lst file.
CHAPTER 6

Indices and tables

• genindex
• modindex
• search
Python Module Index

c
- check_uid
- ctphilips

d
- dcmdatetime

e
- exportcsv

f

get_values

n
- not_patient_indicators

o
- openrem_settings

p
- psizecsv2db

r
- rdsr
- remapp.exports.exportcsv
- remapp.extractors.ct_philips
- remapp.extractors.mam
- remapp.extractors.openrem_settings
- remapp.extractors.psizecsv2db
- remapp.extractors.rdsr
- remapp.tools.check_uid
- remapp.tools.dcmdatetime
- remapp.tools.get_values
- remapp.tools.not_patient_indicators