

---

# **MicroTardis Documentation**

***Release 1***

**Joanna Huang**

May 23, 2012



# CONTENTS

<b>1</b>	<b>MicroTardis Overview</b>	<b>3</b>
1.1	System Framework . . . . .	3
<b>2</b>	<b>MicroTardis Installation Guide</b>	<b>5</b>
2.1	Installing Your MicroTardis Instance . . . . .	5
2.2	Setting up automatic ingesting at RMMF . . . . .	15
<b>3</b>	<b>MicroTardis Development Guide</b>	<b>19</b>
3.1	Communication . . . . .	19
3.2	Issue Trackers . . . . .	19
3.3	Software Repositories . . . . .	19
3.4	Local Development Environment Setup . . . . .	20
3.5	Testing . . . . .	23
3.6	Filters . . . . .	24
3.7	User Registration . . . . .	25
<b>4</b>	<b>MicroTardis Administration Guide</b>	<b>27</b>
4.1	Create New Users . . . . .	27
4.2	Create New Groups . . . . .	36
4.3	Assign Group Owners . . . . .	38
4.4	Manage Group Members . . . . .	38
4.5	Experiment Access Controls . . . . .	38
4.6	Publish Experiment . . . . .	38
<b>5</b>	<b>Indices and tables</b>	<b>39</b>



Contents:

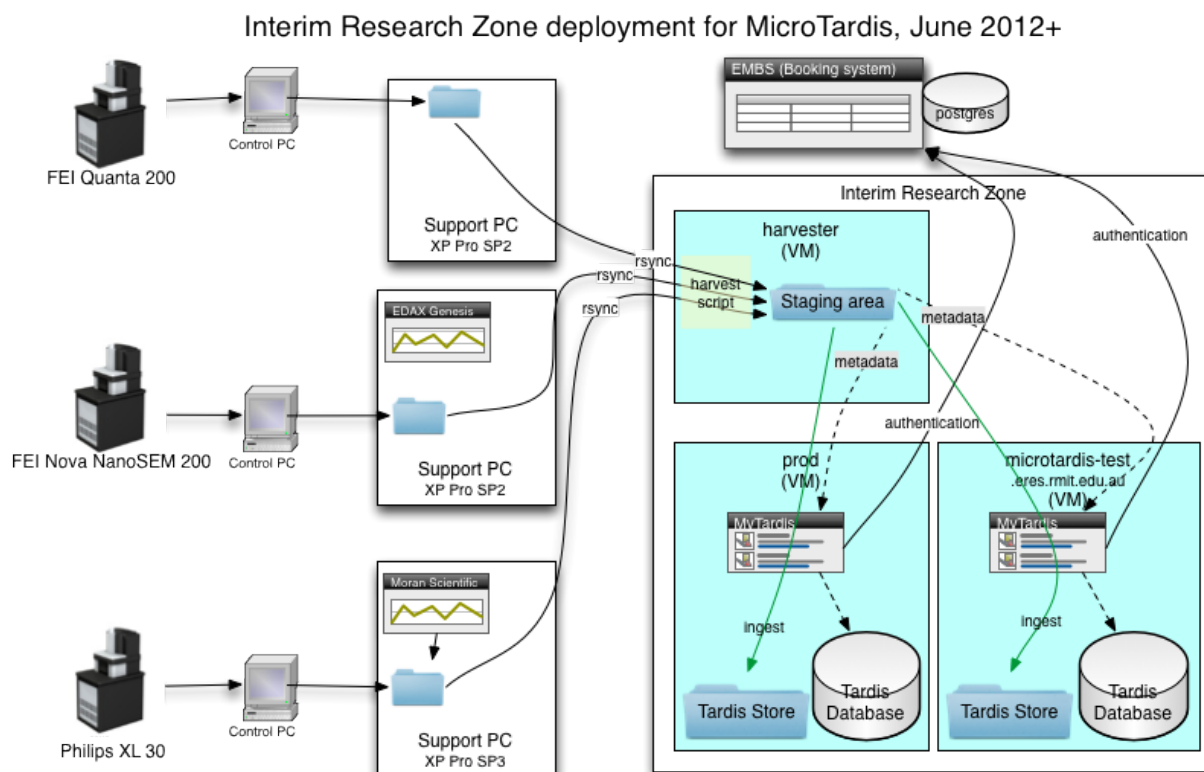


# MICROTARDIS OVERVIEW

MyTARDIS is a multi-institutional collaborative venture that facilitates the archiving and sharing of data and metadata collected at major facilities such as the Australian Synchrotron and ANSTO and within Institutions.

MicroTardis is an extension of the MyTARDIS system for the task of transferring data and metadata from remote microscopy facilities, automatic extraction of microscopy image or spectra metadata, and curation of experiments for researcher access.

## 1.1 System Framework







# MICROTARDIS INSTALLATION GUIDE

This document describes how to deploy MicroTardis/MyTardis system. The documentation includes two main parts: the first describes the installation of the MicroTardis system; and the second describes the installation of facility scripts.

## 2.1 Installing Your MicroTardis Instance

This document describes a step by step guide on how to install MicroTardis/MyTardis system. The documentation includes two main parts: the first part describes the installation of MyTardis system and MicroTardis extensions; and the second part is the instructions of deploying MicroTardis within RMIT ITS network.

### 2.1.1 Installing MicroTardis

**Please note that these installation instructions were written based on a MyTardis 2.5 installation on RHEL 6 with a root permission.**

#### Step 1: Internet Proxy Settings if Within RMIT Network

**Please skip this step if your machine isn't hosted within RMIT network.**

If you would like to install MicroTardis in a RMIT machine, it's required to have RMIT HTTP/HTTPS proxy settings to access the Internet.

1. Copy the following lines into `/etc/environment` with root permission to have system-wide proxy settings:

```
http_proxy=http://bproxy.rmit.edu.au:8080
https_proxy=http://bproxy.rmit.edu.au:8080
```

2. Save the file and re-login.
3. To make sure the setting is there by opening a terminal and issuing the command:

```
export | grep -i proxy
```

then you should see the proxy settings as what you have just configured.

#### Step 2: Prerequisites

MicroTardis is currently only supported on RHEL and Ubuntu with SELinux disabled. The following packages are essential to be system-wide installed.

### 1. Redhat:

```
yum install git gcc gcc-c++ httpd mod_wsgi mysql mysql-server MySQL-python
yum install python python-devel python-setuptools libjpeg-devel numpy python-matplotlib
yum install cyrus-sasl-ldap cyrus-sasl-devel openldap-devel libxslt libxslt-devel libxslt-python
easy_install PIL
```

### 2. Ubuntu:

```
apt-get install git gcc libapache2-mod-wsgi mysql mysql-server python-mysqldb
apt-get install python python-dev python-setuptools python-numpy python-matplotlib
apt-get install libpq-dev libssl-dev libsasl2-dev libldap2-dev libxslt1.1 libxslt1-dev python-l
easy_install PIL
```

## Step 3: Download MyTardis Source Code

### 1. Check out MyTardis source code into /opt/ directory:

```
cd /opt
git clone git://github.com/mytardis/mytardis.git
```

If you get an error of **Connection timed out** as shown below:

```
Initialized empty Git repository in /opt/mytardis/.git/
github.com[0: 207.97.227.239]: errno=Connection timed out
fatal: unable to connect a socket (Connection timed out)
```

It means that the git port (9418) might be blocked due to a firewall constraint. Please try the following commands to clone GitHub repository over HTTP/HTTPS instead:

```
cd /opt
git config --global http.proxy bproxy.rmit.edu.au:8080
git clone https://github.com/mytardis/mytardis.git
```

It might be slow and recommended to be used if the git port(9418) is blocked.

### 2. Get the MyTardis 2.5 release branch:

```
cd mytardis
git tag -l
git checkout 2.5.0-rc1
```

then you will see messages similar to the one below:

```
Note: checking out '2.5.0-rc1'.
```

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using `-b` with the checkout command again. Example:

```
git checkout -b new_branch_name
```

```
HEAD is now at 9615e03... Merge pull request #64 from shaunokeefe/sync-rebased
```

it means that you have checked out '2.5.0-rc1' branch successfully.

## Step 4: Download MicroTardis Extensions

1. To get the current master branch of MicroTardis and install it inside MyTardis folder:

```
cd /opt/mytardis/tardis
git clone https://github.com/mytardis/microtardis.git
```

Please note that it is essential to check out microtardis source codes into `/opt/mytardis/tardis` directory where the `tardis_portal` directory is. The `tardis_portal` directory contains main functions of MyTardis. It is necessary for MicroTardis to live in the same location of it to reuse or override its features.

## Step 5: Building

MicroTardis/MyTardis uses the Buildout Python-based build system to automatically create, assemble and deploy applications or modules required by MicroTardis/MyTardis project. It would automatically download and install the modules and their dependencies inside `/opt/mytardis` directory. Please note that this is not a system-wide installation. Buildout uses a Python tool called `setuptools` internally to install the packages.

1. Run the **bootstrap** script to bootstrap a buildout-based project:

```
cd /opt/mytardis
python bootstrap.py
```

2. Run the **buildout** script to download and install Python eggs and all dependencies:

```
cd /opt/mytardis
bin/buildout
```

*This can be run again at any time to check for and download any new dependencies.*

## 2.1.2 Deploying MicroTardis

### Step 6: MicroTardis settings.py File

Configuring MicroTardis/MyTardis is done through a standard Django `settings.py` file. MyTardis comes with a sample configuration file at `/opt/mytardis/tardis/settings_changeme.py`. In MicroTardis, there is also a settings file called `/opt/mytardis/tardis/microtardis/settings_microtardis.py` which is an extension of `/opt/mytardis/tardis/settings_changeme.py` that includes support to MicroTardis application.

1. To create a `settings.py` file for your deployment, just copy the file `/opt/mytardis/tardis/microtardis/settings_microtardis.py` into the directory where `settings_changeme.py` is in:

```
cp /opt/mytardis/tardis/microtardis/settings_microtardis.py /opt/mytardis/tardis/settings.py
```

2. To configure MicroTardis for interactive use to proceed following parts of configuration, please edit the file `/opt/mytardis/bin/django` and replace the following line:

```
djangorecipe.manage.main('tardis.test_settings')
```

with:

```
djangorecipe.manage.main('tardis.settings')
```

This means that the `/opt/mytardis/bin/django` command will run the interactive configuration rather than the test configuration. And we will use this command later on to manually create database tables or superuser, and so on.

### Step 7: MicroTardis Database

1. Ensure that the MySQL database has been started:

```
/etc/init.d/mysqld start
```

2. Configure MySQL to run every time the system starts:

```
chkconfig mysqld on
```

3. Create a database named **microtardis**:

```
mysql -e "CREATE DATABASE microtardis"
```

4. Run the following command to configure the database, and create user account and password; don't forget to replace **'microtardisuser'** and **'secret'** with a user name and a password of your choices:

```
mysql -e "GRANT ALL PRIVILEGES ON microtardis.* TO 'microtardisuser'@'localhost' IDENTIFIED BY 'secret'"
```

5. Edit the `/opt/mytardis/tardis/settings.py` file which you have just created in former step. Please ensure that the values of database parameters in `settings.py` match the values used to create your MicroTardis database:

```
DATABASES = {}
DATABASES['default'] = {}
DATABASES['default']['ENGINE'] = 'django.db.backends.mysql'
DATABASES['default']['HOST'] = 'localhost'
DATABASES['default']['PORT'] = '3306'
DATABASES['default']['NAME'] = 'microtardis'
DATABASES['default']['USER'] = 'microtardisuser'
DATABASES['default']['PASSWORD'] = 'secret'
```

This is the minimum set of changes required to successfully run the server. You can make any other site-specific changes in `/opt/mytardis/tardis/settings.py` as necessary.

6. (OPTIONAL) For the purpose of database maintenance, you might need to have root access to MySQL database. If you have root access, run the following command to ensure that the MySQL instance has a root password; don't forget to replace the word **'rootsecret'** with a password of yours:

```
mysqladmin password rootsecret
```

If you need to reset MySQL root password, then run the following command to reset the password of your choice:

```
mysqladmin -u root -pcurrentpassword password 'newpassword'
```

Please note that there is no space between **-p** and **currentpassword**. Or you can also change MySQL root password from MySQL prompt using UPDATE SQL command:

```
mysql> UPDATE user SET password=PASSWORD('newpassword') WHERE user='root';
mysql> FLUSH PRIVILEGES;
mysql> EXIT;
```

Once you've changed it, make sure you can login with your new password successfully. And now kill your running MySQL daemon, then restart it normally.

7. Rename `/opt/mytardis/tardis/tardis_portal/fixtures/initial_data.json` to ignore importing synchrotron-specific metadata schema:

```
cd /opt/mytardis/tardis/tardis_portal/fixtures/  
mv initial_data.json initial_data.json.ignored
```

The synchrotron-specific metadata schema is part of default schema in MyTardis 2.5 release branch. However MicroTardis doesn't use it for microscopy metadata data.

8. Run the following command to setup the database tables in the MySQL database:

```
cd /opt/mytardis  
bin/django syncdb --noinput --migrate
```

If you encountered an error looks like:

```
_mysql_exceptions.OperationalError: (1170, "BLOB/TEXT column 'string_value' used in key specification; the table must have a primary key")
```

Please ignore it for the moment. It's a bug in MyTardis, and hopefully they will fix it soon in next version of MyTardis. If you would like to know what the actual cause of this error is, please refer to [MERROR 1170 \(42000\)](#) for more details.

## Step 8: MicroTardis Administrator

1. Create an administrator account:

```
cd /opt/mytardis  
bin/django createsuperuser
```

Please keep your user name and password. You will need them to sign in MicroTardis administrator web interface.

## Step 9: Static Files

For performance reasons you should avoid static files being served via the application, and instead serve them directly through the webserver.

1. To collect all the static files to a single directory:

```
cd /opt/mytardis  
bin/django collectstatic
```

## Step 10: MicroTardis Staging Area and Store

In MyTardis/MicroTardis, **staging area** is an intermediate data storage area between the sources of raw data and the MyTardis/MicroTardis **data store**. It is used for gathering data from different sources that will be ready to ingest into MyTardis/MicroTardis data store at different times.

1. Setup MicroTardis staging area and data store

The default location of staging area or data store is in `mytardis/var`. If you have followed the installation instructions above, you should be able to see them:

```
ls -dl /opt/mytardis/var/staging  
ls -dl /opt/mytardis/var/store
```

You might have noticed that both of them are empty directories. In MicroTardis, data store is a file storage to keep ingested files with its specific file directory structure. In this part you are not expected to change or modify any data in MicroTardis data store including files and directories.

However, you are required to manually create a **staging structure** in MicroTardis staging area. Again, it needs a specific folder structure inside staging to enable data ingestion from staging area into data store and metadata extraction using predefined microscope-specific data filters. Please follow the short instructions below to create the staging area structure for your deployment.

- (a) The first thing to do is to create **user folders** inside your staging area:

```
cd /opt/mytardis/var/staging
mkdir your_username
```

You can use the administrator account that you've just created.

- (b) Then create **microscope folders** inside user folders with any name of microscope which is currently supported in MicroTardis: XL30, NovaNanoSEM, and Quanta200. For example:

```
cd /opt/mytardis/var/staging/your_username
mkdir NovaNanoSEM
```

MicroTardis currently only supports the following microscopes,

- Philips XL30 SEM (1999) with Oxford Si(Li) X-ray detector and HKL EDSD system
- FEI Nova NanoSEM (2007) with EDAX Si(Li) X-ray detector
- FEI Quanta 200 ESEM with EDAX Si(Li) X-ray detector and Gatan Alto Cyro stage

2. Copy example files into your microscope folders. Here are some example files for you to download for the purpose of testing,

- (a) XL30

- XL30.dat
- XL30.spt
- XL30.tif

- (a) NovaNanoSEM

- NovaNanoSEM.spc
- NovaNanoSEM.tif

- (a) Quanta200

- Quanta200.spc
- Quanta200.tif

Download them into microscope folders according to different microscopes. Then you will be able to see the folders/files you've just created/downloaded on *MicroTardis Create Experiment* web interface later after you successfully start your deployment server.

3. (OPTIONAL) Specify directory paths of your own staging area and data store if you would like to change the locations of them instead of using the default ones.

- (a) Edit your settings.py file, for example:

```
vi /opt/mytardis/tardis/settings.py
```

- (b) Find the following lines in the settings.py file:

```
#STAGING_PATH = '/directory/path/of/your/own/staging'
#FILE_STORE_PATH = '/directory/path/of/your/own/store'
```

(c) Uncomment the line and specify the real location of your own staging area or data store.

#### 4. (OPTIONAL) Set up remote staging area and data store.

If you need to use remote or mounted staging/store area, please create symbolic links in /opt/mytardis/var to replace default staging and store directories.

(a) Create a symbolic link for staging area from MicroTardis to the remote storage:

```
cd /opt/mytardis/var
rmdir staging
ln -s /mnt/your_remote_staging staging
```

(b) Create a symbolic link for data store from MicroTardis to the remote storage:

```
cd /opt/mytardis/var
rmdir store
ln -s /mnt/your_remote_store store
```

#### 5. (OPTIONAL) With respect to automatic data collection on staging area which automatically harvests data from data sources into staging area, please see an example of [RMIT MicroTardis Data Harvest](#) for more details.

### Step 11: Apache and mod\_wsgi

#### 1. Create a symbolic link from MyTardis to standard /var/www/html structure (makes a fixed path for later changes):

```
cd /var/www/html
chmod o+w /var/www/html
sudo -u apache ln -s /opt/mytardis mytardis
chmod o-w /var/www/html
```

#### 2. Set up a virtual host for MicroTardis web portal by editing /etc/httpd/conf/httpd.conf file:

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html/mytardis
    <Directory />
        Options +FollowSymLinks
        AllowOverride None
    </Directory>
    <Directory /var/www/html/mytardis>
        Options Indexes +FollowSymLinks MultiViews
        AllowOverride All
        Order allow,deny
        allow from all
    </Directory>
</VirtualHost>
```

#### 3. Edit /etc/httpd/conf.d/wsgi.conf file:

```
LoadModule wsgi_module modules/mod_wsgi.so
<IfModule mod_wsgi.c>
    AddHandler wsgi-script .wsgi
    Include /var/www/html/mytardis/apache/apache_django_wsgi.conf
</IfModule>
```

4. Create `apache_django_wsgi.conf` file:

```
cd /var/www/html/mytardis/apache/
cp apache_django_wsgi.conf_changeme apache_django_wsgi.conf
```

5. Edit the `apache_django_wsgi.conf` file as shown below:

```
Alias /static/ /var/www/html/mytardis/static/
<Directory /var/www/html/mytardis/static/>
Order deny,allow
Allow from all
</Directory>

WSGIScriptAlias / "/var/www/html/mytardis/apache/django.wsgi"

<Directory "/var/www/html/mytardis/apache">
Allow from all
</Directory>
```

Remember to delete or comment out all the original configuration in `apache_django_wsgi.conf`:

```
WSGIScriptAlias / "/Users/steve/django-jython-svn/myTARDIS_checkout/tardis/apache/django.wsgi"

<Directory "/Users/steve/django-jython-svn/myTARDIS_checkout/tardis/apache">
Allow from all
</Directory>
```

6. Create `django.wsgi` file:

```
cd /var/www/html/mytardis/apache/
cp django.wsgi_changeme django.wsgi
```

7. Edit the `django.wsgi` file with instructions shown below followed by an example of `django.wsgi`.

- (a) Please copy the value of **sys.path** variable from `/opt/mytardis/bin/django.wsgi` file which is a list of full directory paths of modules required by MicroTardis.

- (b) Remember to delete or comment out the following line in your `django.wsgi` file:

```
sys.path.append('/Users/steve/django-jython-svn/myTARDIS_checkout')
```

- (c) Example:

```
#!/usr/bin/python

import os
import sys
sys.path[0:0] = [
    '/opt/mytardis',
    '/opt/mytardis/eggs/nose-1.1.2-py2.6.egg',
    '/opt/mytardis/eggs/coverage-3.4-py2.6-linux-x86_64.egg',
    '/opt/mytardis/eggs/django_nose-1.0-py2.6.egg',
    '/opt/mytardis/eggs/nosexcover-1.0.7-py2.6.egg',
    '/opt/mytardis/eggs/python_ldap-2.4.9-py2.6-linux-x86_64.egg',
    '/opt/mytardis/eggs/python_magic-0.4.0dev-py2.6.egg',
    '/opt/mytardis/eggs/python_memcached-1.48-py2.6.egg',
    '/opt/mytardis/eggs/pysolr-2.1.0_beta-py2.6.egg',
    '/opt/mytardis/eggs/docutils-0.8.1-py2.6.egg',
    '/opt/mytardis/eggs/flexmock-0.9.3-py2.6.egg',
    '/opt/mytardis/eggs/compare-0.2b-py2.6.egg',
    '/opt/mytardis/eggs/django_jasmine-0.3.2-py2.6.egg',
```



```

' /opt/mytardis/eggs/celery-2.5.1-py2.6.egg',
' /opt/mytardis/eggs/django_celery-2.5.1-py2.6.egg',
' /opt/mytardis/eggs/django_kombu-0.9.4-py2.6.egg',
' /opt/mytardis/eggs/iso8601-0.1.4-py2.6.egg',
' /opt/mytardis/eggs/html2text-3.200.3-py2.6.egg',
' /opt/mytardis/eggs/pyoai-2.4.4-py2.6.egg',
' /opt/mytardis/eggs/Wand-0.1.9-py2.6.egg',
' /opt/mytardis/eggs/djangorecipe-1.1.2-py2.6.egg',
' /opt/mytardis/eggs/Django-1.3-py2.6.egg',
' /opt/mytardis/eggs/zc.recipe.egg-1.3.2-py2.6.egg',
' /opt/mytardis/eggs/zc.buildout-1.5.2-py2.6.egg',
' /opt/mytardis/eggs/lxml-2.2.7-py2.6-linux-x86_64.egg',
' /opt/mytardis/eggs/django_picklefield-0.2.0-py2.6.egg',
' /opt/mytardis/eggs/ordereddict-1.1-py2.6.egg',
' /opt/mytardis/eggs/python_dateutil-1.5-py2.6.egg',
' /opt/mytardis/eggs/kombu-2.1.3-py2.6.egg',
' /opt/mytardis/eggs/anyjson-0.3.1-py2.6.egg',
' /opt/mytardis/eggs/importlib-1.0.2-py2.6.egg',
' /opt/mytardis/eggs/setuptools-0.6c12dev_r88846-py2.6.egg',
' /opt/mytardis/eggs/httpplib2-0.7.4-py2.6.egg',
' /opt/mytardis/eggs/pytz-2012b-py2.6.egg',
' /opt/mytardis/eggs/South-0.7.4-py2.6.egg',
' /opt/mytardis/eggs/BeautifulSoup-3.2.1-py2.6.egg',
' /opt/mytardis/eggs/django_haystack-1.2.6-py2.6.egg',
' /opt/mytardis/eggs/django_form_utils-0.2.0-py2.6.egg',
' /opt/mytardis/eggs/django_extensions-0.8-py2.6.egg',
' /opt/mytardis/eggs/django_registration-0.8-py2.6.egg',
' /opt/mytardis/eggs/elementtree-1.2.7_20070827_preview-py2.6.egg',
' /opt/mytardis/eggs/feedparser-5.1.1-py2.6.egg',
' /opt/mytardis/eggs/amqplib-1.0.2-py2.6.egg',
' /opt/mytardis/parts/django',
]

os.environ['DJANGO_SETTINGS_MODULE'] = 'tardis.settings'
import django.core.handlers.wsgi
application = django.core.handlers.wsgi.WSGIHandler()

```

## Step 12: Permission Settings

MicroTardis is a Python web application. The web server (i.e. Apache) needs to have permissions to access the WSGI script in the directory of MicroTardis, or write log files or data into it. The following commands will give apache user to do this.

1. As root, make all file/directories in mytardis as group *apache* with *rx* access permission:

```
chgrp apache -R /opt/mytardis
chmod g+rx -R /opt/mytardis
```

2. Enable apache to write log files into the default directory:

```
chmod g+w /opt/mytardis
```

3. Set proper file access permission to `/opt/mytardis/var` directory to make Apache able to write data in data store:

```
chmod g+rw -R /opt/mytardis/var
```

### Step 13: SELinux

1. Disable SELinux protection in RHEL.

- (a) To turn SELinux off immediately, without rebooting use (turning off SELinux temporarily):

```
setenforce 0
```

- (b) Completely turning off SELinux,

Edit `/etc/selinux/config` (e.g. `$sudo vi /etc/selinux/config`).

Find the line:

```
SELINUX=enforcing
```

If you simply want to set selinux to *permissive* mode which will still warn you when something would have been denied, change it to:

```
SELINUX=permissive
```

If you want to completely disable SELinux, change it to:

```
SELINUX=disabled
```

Save the file, then you will need to reboot your system to create the desired effect.

### Step 14: Firewall Settings

1. Open file `/etc/sysconfig/iptables`:

```
vi /etc/sysconfig/iptables
```

2. Append rules as follows:

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
-A INPUT -m state --state NEW -p tcp --dport 443 -j ACCEPT
```

3. Save and close the file.

4. Restart iptables:

```
/etc/init.d/iptables restart
```

### Step 15: MicroTardis Web Portal

1. Configure Apache to run every time the system starts:

```
chkconfig httpd on
```

2. Test if Apache service is running:

```
service httpd status
```

3. Start Apache service,

- (a) Simply start Apache service if it's not running:

```
service httpd start
```

- (b) Restart Apache service if it's already running:

```
service httpd restart
```

4. Check if MicroTardis Web Portal is working fine via browser with URL:

```
http://your_hostname.domain_name/
```

For example:

```
http://microtardis-test.eres.rmit.edu.au/
```

If everything works fine, then you will be able to see MicroTardis's Welcome web page.

## 2.2 Setting up automatic ingesting at RMMF

In order for MicroTardis to automatically harvest user data from support PCs at RMMF, four things are required:

1. An Rsync client (DeltaCopy) installed on each of the support PCs. (As of writing - May 2012 - this is done).
2. Harvest scripts and cron jobs installed on harvest machine. (As of writing, this is `datapuller.isis.rmit.edu.au`, and has been done).
3. Atom dataset provider installed on harvest machine. (As of writing, this is `datapuller.isis.rmit.edu.au` and has been done).
4. Atom ingest installed and configured on MicroTardis machine. (As of writing, this has only been done on `microtardis-test.eres.rmit.edu.au`)

### 2.2.1 1. Installing DeltaCopy

Straightforward. Download, install. It's free. Make it run as a service all the time. Make a note of the IP address and name of shared folder.

### 2.2.2 2. Installing harvest scripts

The scripts are simply bash scripts that use rsync to copy files over from the three support PCs. They should be triggered by a cron job:

```
cd /usr/local
git clone https://github.com/stevage/MicroTardis-Harvest
```

To configure which individual machines to harvest from and where to store data to, modify the scripts - they are commented.

### 2.2.3 3. Install atom dataset provider

The Atom dataset provider is a server which, when requested, scans the staging area directories and provides a list of the most recent files as an Atom feed, so they can be ingested into Tardis.

Get the source code here: <https://github.com/stevage/atom-dataset-provider>

It is installed on the Harvest machine in `/usr/local/microtardis/atom-dataset-provider`.

Use this script to start it:

```
#!/bin/bash
# /usr/local/microtardis/atom-dataset-provider/provider.sh
bash -x ./kill-provider.sh
set -x
LOG='pwd'../logs/atom-dataset-provider.log
PATTERN='([EeSs][0-9]+)'
EXCLUDEPATTERN='[Tt]humbs.db|e80940/.*Old\ Data/'
GROUPPATTERN='/^('"{STAGING}"${PATTERN}"'/[^/]+/[^/]+/[^/]+/).+$/
PATH=/usr/local/microtardis/local/bin:$PATH
STAGING=/mnt/np_staging/
nohup 'pwd'bin/atom-dataset-provider -d "$STAGING" --group-pattern "$GROUPPATTERN" --exclude-patter
```

The variables *GROUPPATTERN*, *PATTERN* and *EXCLUDEPATTERN* are regular expressions which define which files and folders are included on the atom feed, and how folders are grouped as datasets.

By default, it runs on port 4000. On the datapuller machine, Apache must be configured to forward “<http://datapuller.isis.rmit.edu.au/atom>” to local port 4000.

## 2.2.4 4. Install atom ingest

Two scripts described here (*auto\_ingest.sh* and *killcelery.sh*) require sudo privileges. It is probably possible to find an alternative way to install that doesn’t require sudo.

Atom ingest is a Django “app” that is installed inside MyTardis. It has no user interface, but runs in the background to periodically poll the Atom dataset provider for data. When new data is found, it creates records in MyTardis, then copies the files over to the Tardis store:

```
cd /opt/mytardis/tardis/apps
sudo mkdir mytardis-app-atom
sudo chown <youruser> mytardis-app-atom
```

Get the source code. Currently this is in my (Steve Bennett’s) GitHub repository, but this may change:

```
git clone https://github.com/stevage/mytardis-app-atom
```

The code needs to be visible to the Apache user:

```
sudo chmod g+r -R mytardis-app-atom
```

Module names with hyphens cause problems for Python, so we rename it:

```
mv mytardis-app-atom atom
```

The Atom ingest app can be configured with many different policies for user creation, experiment creation etc. Edit the file *atom/options.py*. These are the settings we currently use:

**class IngestOptions:**

```
# Names of parameters, must match fixture entries.
# Some are also used for <category> processing in the feed itself.
PARAM_ENTRY_ID = 'EntryID'
PARAM_EXPERIMENT_ID = 'ExperimentID'
PARAM_UPDATED = 'Updated'
PARAM_EXPERIMENT_TITLE = 'ExperimentTitle'

ALLOW_EXPERIMENT_CREATION = True           # Should we create new experiments
ALLOW_EXPERIMENT_TITLE_MATCHING = True     # If there's no id, is the title enough to match on
ALLOW_UNIDENTIFIED_EXPERIMENT = False     # If there's no title/id, should we process it as "uncat
DEFAULT_UNIDENTIFIED_EXPERIMENT_TITLE="Uncategorized Data"
```

```

ALLOW_UNNAMED_DATASETS = True           # If a dataset has no title, should we ingest it with a
DEFAULT_UNNAMED_DATASET_TITLE = '(assorted files)'
ALLOW_USER_CREATION = False             # If experiments belong to unknown users, create them?
# Can existing datasets be updated? If not, we ignore updates. To cause a new dataset to be created,
# feed must have a unique EntryID for the dataset (eg, hash of its contents).
ALLOW_UPDATING_DATASETS = True
# If a datafile is modified, do we re-harvest it (creating two copies)? Else, we ignore the update.
ALLOW_UPDATING_DATAFILES = True
HIDE_REPLACED_DATAFILES = True
# If files are served as /user/instrument/experiment/dataset/datafile.tif
# then 'datafile.tif' is at depth 5. This is so we can maintain directory structure that
# is significant within a dataset. Set to -1 to assume the deepest directory.

DATAFILE_DIRECTORY_DEPTH = 7 # /mnt/rmmf_staging/el23/NovaNanoSEM/exp1/dsl/test3.tif

# Yes, we want to extract metadata from ingested files.
USE_MIDDLEWARE_FILTERS = True

# If we can transfer files "locally" (ie, via SMB mount), then replace URL_BASE_TO_REPLACE with
# to construct a file path that can be copied from.
USE_LOCAL_TRANSFERS = True
URL_BASE_TO_REPLACE = "http://datapuller.isis.rmit.edu.au/"
LOCAL_SOURCE_PATH = "/mnt/rmmf_staging/"

# Should we always examine every dataset entry in the feed, even after encountering "old" entries?
ALWAYS_PROCESS_FULL_FEED = False

HTTP_PROXY = "http://bproxy.rmit.edu.au:8080"

```

It is likely these will need to be changed as requirements change. In particular, `ALLOW_EXPERIMENT_CREATION` may need to be turned off - it is useful for importing large amounts of data initially.

Next, configure the CeleryD tasks that fire the auto ingest. CeleryD is a scheduling mechanism used by MyTardis.

If the file `atom/settings_atom.py` doesn't exist, create it. Make its contents as follows:

```

# Settings to ensure atom ingest is triggered by celery.
import djcelery
from datetime import timedelta

CELERYBEAT_SCHEDULE = {
    # Every minute, check for new datasets.
    "update-feeds": {
        "task": "atom_ingest.walk_feed",
        "schedule": timedelta(seconds=60),
        "args": ('http://datapuller.isis.rmit.edu.au/atom',),
    },
    # Less frequently, do a full harvest to see if we have missed anything.
    "update-feeds-full": {
        "task": "atom_ingest.walk_feed",
        "schedule": timedelta(seconds=900),
        "args": ('http://datapuller.isis.rmit.edu.au/atom', True)
    },
}

# Multiple concurrent tasks makes logs complicated and doesn't improve performance.
CELERYD_CONCURRENCY = 1
djcelery.setup_loader()

```

Now, install the app into MyTardis. In `/opt/mytardis/tardis/settings.py`, find the line `"INSTALLED_APPS = ("tardis.microtardis",) + INSTALLED_APPS"`. Add two lines as follows:

```
INSTALLED_APPS = ("tardis.microtardis",) + INSTALLED_APPS
INSTALLED_APPS = ("tardis.apps.atom",) + INSTALLED_APPS
from tardis.apps.atom.settings_atom import *
```

Note the `"tardis.apps.atom"` name matches the directory structure: `tardis/apps/atom`.

The app is now installed, but CeleryD is not running. Create this script in `/opt/mytardis/tardis/autoingest.sh`:

```
#!/bin/bash -x
if [ `whoami` != root ]; then
    echo This script needs to be run as sudo.
    exit
fi
LOG=/var/www/html/mytardis/autoingest.log
sudo -u apache bash -c "nohup `pwd`/bin/django celeryd --beat --purge --loglevel=INFO >> $LOG &"
```

To be able to stop the app, create this script in `/opt/mytardis/tardis/killcelery.sh`:

```
#!/bin/bash -x
ps ax | grep "[c]eleryd" | awk {'print $1'} | xargs kill -9
```

And of course:

```
chmod a+x autoingest.sh killcelery.sh
```

To start the autoingest:

```
./autoingest.sh ; tail -f autoingest.log
```

## 2.2.5 Troubleshooting

If you get errors of this type:

```
...
File "/opt/mytardis/tardis/apps/atom/atom_ingest.py", line 16, in <module>
    from tardis.tardis_portal.util import get_local_time, get_utc_time, get_local_time_naive
ImportError: cannot import name get_local_time_naive
```

You need to install some bug fixes to the MyTardis code:

```
$ cd /opt/mytardis/tardis
$ git remote add steve https://stevage@github.com/stevage/mytardis-ruggedisation.git
$ git fetch steve
```

(If a password is requested, press enter.):

```
$ git merge steve/atom-ingest-fixes
```

# MICROTARDIS DEVELOPMENT GUIDE

This document describes information for MicroTardis developers.

## 3.1 Communication

1. The [microtardis](#) email list is used for communication between developers. To join this list, fill in the form on the [Contact owner to join](#) page.
2. Alternatively, [tardis-devel](#) email list is the one you can use to contact MyTardis/MicroTardis community. New contributors are welcome, however all developers should review the [pull-request checklist](#) before making pull requests.

## 3.2 Issue Trackers

1. The main entry point for users and system administrator is [microtardis](#) email list.
2. Defects and issues found in the MicroTardis software are tracked using the [MyTardis Lighthouse](#) tracker.

## 3.3 Software Repositories

1. The GitHub service is used for MicroTardis software repository and can be browsed using the GitHub source-code browser,
  - (a) [MyTardis at GitHub](#)
  - (b) [MicroTardis at GitHub](#)
2. **For write access (git push permission) to the MicroTardis repository, email the [microtardis](#) list.**

If you are a contributor to MicroTardis software, please create your personal account in [GitHub](#), and send your account name to [microtardis](#) asking for pull and push permissions to MicroTardis GitHub repository.
3. Check out code from GitHub
  - (a) MyTardis software repository

The software can be checked out with the following command:

```
git clone https://github.com/mytardis/mytardis.git
```

(b) MicroTardis software repository

If you have write access to MicroTardis github repository, use the command below with your own username instead to check out MicroTardis source codes:

```
git clone https://username@github.com/mytardis/microtardis.git
```

For anonymous checkouts, the following command can be used:

```
git clone https://github.com/mytardis/microtardis.git
```

## 3.4 Local Development Environment Setup

### 1. Internet Proxy Settings if Within RMIT Network

**Please skip this step if your development machine isn't hosted within RMIT network.**

If you would like to install MicroTardis in a RMIT machine, it's required to have RMIT HTTP/HTTPS proxy settings to access the Internet.

- Copy the following lines into `/etc/environment` with root permission to have system-wide proxy settings:

```
http_proxy=http://bproxy.rmit.edu.au:8080
https_proxy=http://bproxy.rmit.edu.au:8080
```

- Save the file and re-login.
- To make sure the setting is there by opening a terminal and issuing the command:

```
export | grep -i proxy
```

then you should see the proxy settings as what you have just configured.

### 2. Prerequisites

MicroTardis is currently only supported on RHEL and Ubuntu. The following packages are essential to be system-wide installed.

- Redhat:

```
yum install git gcc gcc-c++ httpd mod_wsgi mysql mysql-server MySQL-python
yum install python python-devel python-setuptools libjpeg-devel numpy python-matplotlib
yum install cyrus-sasl-ldap cyrus-sasl-devel openldap-devel libxslt libxslt-devel libxslt-python
easy_install PIL
```

- Ubuntu:

```
apt-get install git gcc libapache2-mod-wsgi mysql mysql-server python-mysqldb
apt-get install python python-dev python-setuptools python-numpy python-matplotlib
apt-get install libpq-dev libssl-dev libsasl2-dev libldap2-dev libxslt1.1 libxslt1-dev python
easy_install PIL
```

### 3. Download MyTardis source code

- Choose a folder to install MyTardis. For example, your home directory.
- Check out latest version of MyTardis Source Code:

```
cd ~
git clone https://github.com/mytardis/mytardis.git
```



#### 4. Download MicroTardis extensions

- Check out latest version of MicroTardis Extensions:

```
cd ~/mytardis/tardis
git clone https://github.com/mytardis/microtardis.git
```

Please note that it is essential to check out microtardis source codes into /opt/mytardis/tardis directory where the tardis\_portal directory is. The tardis\_portal directory contains main functions of MyTardis. It is necessary for MicroTardis to live in the same location of it to reuse or override its features.

#### 5. Use **Buildout** to set up a development environment

MicroTardis/MyTardis uses the Buildout Python-based build system to automatically create, assemble and deploy applications or modules required by MicroTardis/MyTardis project to build a local development environment. It would automatically download and install the modules and their dependencies inside /opt/mytardis directory. Please note that this is not a system-wide installation. Buildout uses a Python tool called setuptools internally to install the packages.

- Run the **bootstrap** script to bootstrap a buildout-based project:

```
cd ~/mytardis
python bootstrap.py
```

- Run the **buildout** script to download and install Python eggs and all dependencies:

```
cd ~/mytardis
bin/buildout
```

*This can be run again at any time to check for and download any new dependencies.*

#### 6. Create settings.py file

Configuring MicroTardis/MyTardis is done through a standard Django *settings.py* file. MyTardis comes with a sample configuration file at ~/mytardis/tardis/settings\_changeme.py. In MicroTardis, there is also a settings file called ~/mytardis/tardis/microtardis/settings\_microtardis.py which is an extension of ~/mytardis/tardis/settings\_changeme.py that includes support to MicroTardis application.

To create a settings.py file in your developmnet server, just copy the file ~/mytardis/tardis/microtardis/settings\_microtardis.py into the directory where settings\_changeme.py is in:

```
cp ~/mytardis/tardis/microtardis/settings_microtardis.py ~/mytardis/tardis/settings.py
```

#### 7. To configure MicroTardis for interactive use to proceed following parts of configuration, please edit the file ~/mytardis/bin/django and replace the following line:

```
djangorecipe.manage.main('tardis.test_settings')
```

with:

```
djangorecipe.manage.main('tardis.settings')
```

This means that the ~/mytardis/bin/django command will run the interactive configuration rather than the test configuration. And we will use this command later on to manually create database tables or superuser, and so on.

#### 8. To configure database for development purpose, edit the database settings in ~/mytardis/tardis/settings.py file which you have just created as shown below:

```
DATABASES = {}
DATABASES['default'] = {}
DATABASES['default']['ENGINE'] = 'django.db.backends.sqlite3'
DATABASES['default']['HOST'] = ''
DATABASES['default']['PORT'] = ''
DATABASES['default']['NAME'] = path.join(path.dirname(__file__), 'microtardis.db').replace('\\', '/')
DATABASES['default']['USER'] = ''
DATABASES['default']['PASSWORD'] = ''
```

9. Rename `~/mytardis/tardis/tardis_portal/fixtures/initial_data.json` to ignore importing synchrotron-specific metadata schema:

```
cd ~/mytardis/tardis/tardis_portal/fixtures/
mv initial_data.json initial_data.json.ignored
```

The synchrotron-specific metadata schema is part of default schema in MyTardis 2.5 release branch. However MicroTardis doesn't use it for microscopy metadata data.

10. Setup database tables in the SQLite database:

```
cd ~/mytardis
bin/django syncdb --noinput --migrate
```

11. Create an administrator account:

```
cd ~/mytardis
bin/django createsuperuser
```

Please keep your user name and password. You will need them to sign in MicroTardis administrator web interface.

12. Setup MicroTardis staging area and data store

In MyTardis/MicroTardis, **staging area** is an intermediate data storage area between the sources of raw data and the MyTardis/MicroTardis **data store**. It is used for gathering data from different sources that will be ready to ingest into MyTardis/MicroTardis data store at different times.

The default location of staging area or data store is in `mytardis/var`. If you have followed the installation instructions above, you should be able to see them:

```
ls -dl ~/mytardis/var/staging
ls -dl ~/mytardis/var/store
```

You might have noticed that both of them are empty directories. In MicroTardis, data store is a file storage to keep ingested files with it specific file directory structure. In this part you are not expected to change or modify any data in MicroTardis data store including files and directories.

However, you are required to manually create a **staging structure** in MicroTardis staging area. Again, it needs a specific folder structure inside staging to enable data ingestion from staging area into data store and metadata extraction using predefined microscope-specific data filters. Please follow the short instructions below to create the staging area structure for your deployment.

- (a) The first thing to do is to create **user folders** inside your staging area:

```
cd ~/mytardis/var/staging
mkdir your_username
```

You can use the administrator account that you've just created.

- (b) Then create **microscope folders** inside user folders with any name of microscope which is currently supported in MicroTardis: XL30, NovaNanoSEM, and Quanta200. For example:

```
cd ~/mytardis/var/staging/your_username
mkdir NovaNanoSEM
```

MicroTardis currently only supports the following microscopes,

- Philips XL30 SEM (1999) with Oxford Si(Li) X-ray detector and HKL EDSD system
- FEI Nova NanoSEM (2007) with EDAX Si(Li) X-ray detector
- FEI Quanta 200 ESEM with EDAX Si(Li) X-ray detector and Gatan Alto Cyro stage

13. Copy example files into your microscope folders. Here are some example files for you to download for the purpose of testing,

(a) XL30

- XL30.dat
- XL30.spt
- XL30.tif

(a) NovaNanoSEM

- NovaNanoSEM.spc
- NovaNanoSEM.tif

(a) Quanta200

- Quanta200.spc
- Quanta200.tif

Download them into microscope folders according to different microscopes.

Then you will be able to see the folders/files you've just created/downloaded on [MicroTardis Create Experiment](#) web interface later after you successfully start your development server.

3. Start the development server:

```
cd ~/mytardis
bin/django runserver
```

4. MicroTardis web portal should now be running at:

<http://127.0.0.1:8000>

5. You can now log into [Django Administration Tool](#) with the administrator account you just created to do routine database maintenance:

<http://127.0.0.1:8000/admin>

## 3.5 Testing

The file `~/mytardis/tardis/microtardis/test_settings_microtardis.py` is an alternative `~/mytardis/tardis/test_settings.py` for MyTardis that includes support for MicroTardis extensions for testing purpose.

1. Copy `~/mytardis/tardis/microtardis/test_settings_microtardis.py` into the directory where the `tardis/test_settings.py` is in:

```
cd ~/mytardis
cp tardis/microtardis/test_settings_microtardis.py tardis/test_settings_microtardis.py
```

2. Run the testcases to verify success:

```
cd ~/mytardis
bin/django test --settings=tardis.test_settings_microtardis
```

## 3.6 Filters

The **POST\_SAVE\_FILTERS** variable in `~/mytardis/tardis/microtardis/settings_microtardis.py` file contains a list of post-save filters that are executed when a new `DataFile` object is created and saved to the database. The MicroTardis Filters are built upon the Django signal infrastructure.

1. The **POST\_SAVE\_FILTERS** variable in settings file is specified like:

```
POST_SAVE_FILTERS = [
    ("tardis.microtardis.filters.exiftags.make_filter", ["MICROSCOPY_EXIF", "http://exif.schema"]),
    ("tardis.microtardis.filters.spctags.make_filter", ["EDAXGenesis_SPC", "http://spc.schema"]),
    ("tardis.microtardis.filters.dattags.make_filter", ["HKLESDS_DAT", "http://dat.schema"]),
]
```

2. The format they are specified in is:

```
(<filter class path>, [args], {kwargs})
```

Where *args* and *kwargs* are both optional.

3. In MicroTardis, filters are in charge of creating microscope metadata schemas in database on the fly and extracting metadata from raw data files and saving metadata into database.

In terms of spectra values extraction, MicroTardis doesn't store those values in database but keep them in spectrum files instead. It has a function called `get_spectra_csv` in `microtardis/views.py` in charge of extracting spectra values from spectrum files (`.spt` or `.spc`) on the fly as users request to download them in CSV file format from web portal interface.

Currently we have the following filters implemented,

Microscope	Detector	Analysis System	File Extension	Filter or Function	Description
Philips XL30 SEM	Si(Li) X-ray detector	EDAX Genesis	.tif	exiftags.py	extract image metadata
.spt		get_spectra_csv in views.py		extract spectra values (in CSV format)	
.dat		dattags.py		extract spectrum metadata	
FEI Nova NanoSEM	Si(Li) X-ray detector	EDAX Genesis	.tif	exiftags.py	extract image metadata
.spc		spctags.py		extract spectrum metadata	
get_spectra_csv in views.py		extract spectra values (in CSV format)			
FEI Quanta 2000 SEM	Si(Li) X-ray detector	EDAX Genesis	.tif	exiftags.py	extract image metadata
.spc		spctags.py		extract spectrum metadata	
get_spectra_csv in views.py		extract spectra values (in CSV format)			

### 3.7 User Registration

MicroTardis has two authentication methods. One is **local** user authentication which is the default solution using a local authentication database. In this case, site administrator is required to perform user account maintenance and validation tasks. For each valid user, site administrator has to manually create a user account with a username and a password and grant this user proper privileges to register he/she in MicroTardis. For a quick guide to create user account, please see [MicroTardis Administration Guide](#).

The other method is **integrated** user authentication with [RMMF Booking System \(EMBS\)](#) which uses a remote RMMF authentication database. MicroTardis site administrator is not required to maintain user accounts in MicroTardis database in this case. It automatically creates a new user account in MicroTardis local authentication database when a valid RMMF user (who has registered as an user in EMBS) first log in MicroTardis with a valid username and a valid password. In this case, the first login is considered as user registration in MicroTardis. After successful registration, MicroTardis would only use local authentication database to authenticate users. There won't be communication between MicroTardis and EMBS for user authentication if the user account already exists in MicroTardis database.



# MICROTARDIS ADMINISTRATION GUIDE

This step-by-step guide will show you how to perform the common administrative tasks on your MicroTardis server with administrative privileges that are required for an administrator to perform these tasks.

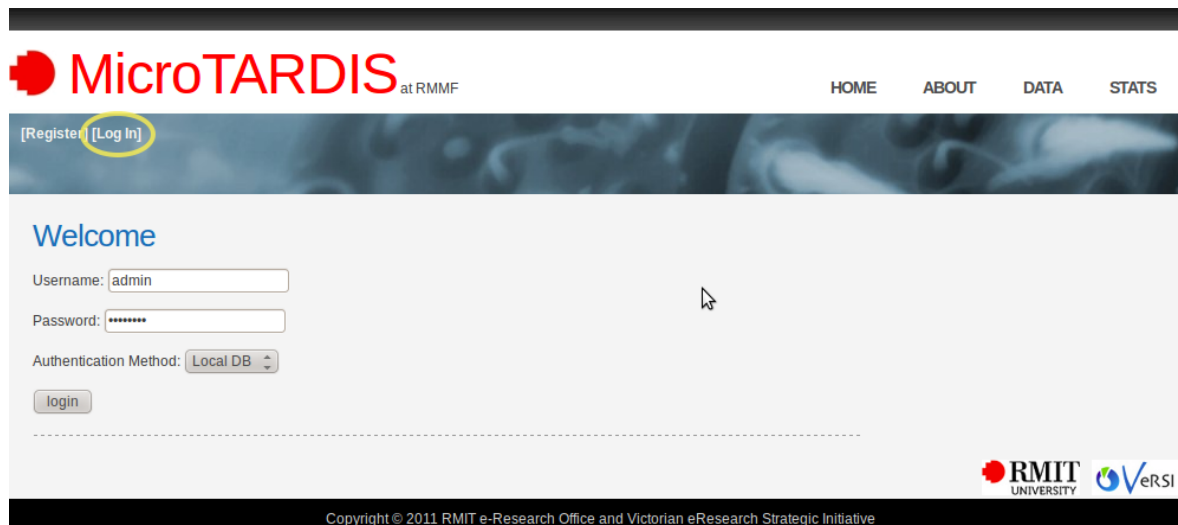
## 4.1 Create New Users

MicroTardis has two authentication methods. One is **local** user authentication which is the default solution using a local authentication database. In this case, site administrator is required to perform user account maintenance and validation tasks. For each valid user, site administrator has to manually create a user account with a username and a password and grant this user proper privileges to register he/she in MicroTardis.

The other method is **integrated** user authentication with [RMMF Booking System \(EMBS\)](#) which uses a remote RMMF authentication database. MicroTardis site administrator is not required to maintain user accounts in MicroTardis database in this case. It automatically creates a new user account in MicroTardis local authentication database when a valid RMMF user (who has registered as an user in EMBS) first log in MicroTardis with a valid username and a valid password. In this case, the first login is considered as user registration in MicroTardis. After successful registration, MicroTardis would only use local authentication database to authenticate users. There won't be communication between MicroTardis and EMBS for user authentication if the user account already exists in MicroTardis database.

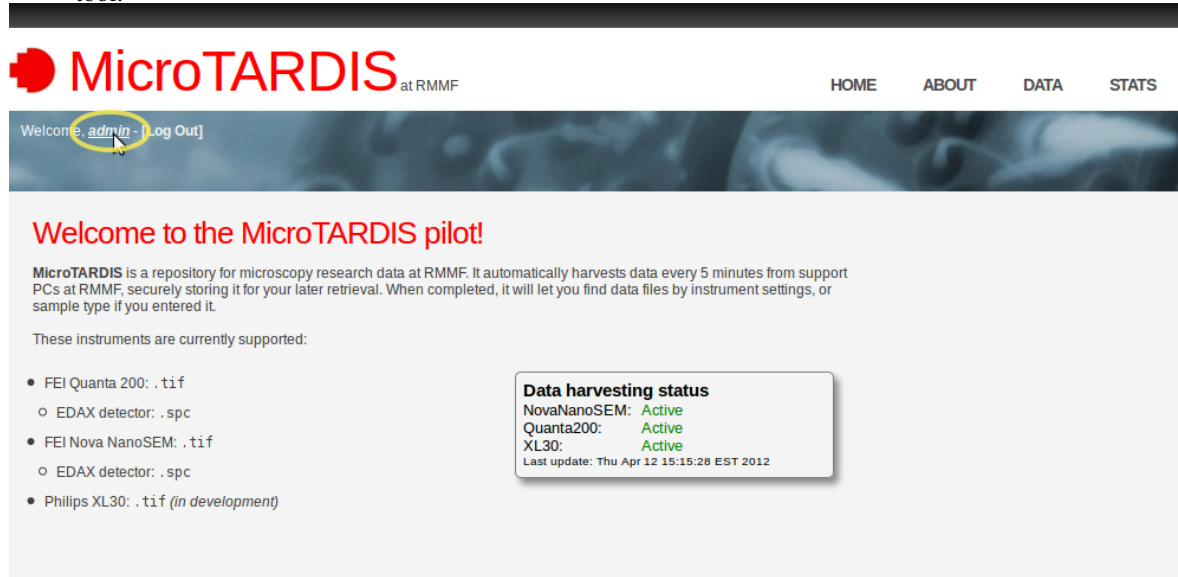
### 4.1.1 Sign in MicroTardis Administration Interface

1. First, please sign in MicroTardis via its login page with your administrator account.



The screenshot shows the MicroTARDIS login page. At the top left is the MicroTARDIS logo with 'at RMMF' text. To the right are navigation links: HOME, ABOUT, DATA, and STATS. Below the logo, there are links for [Register] and [Log In], with the latter circled in yellow. The main content area has a 'Welcome' heading followed by login fields: 'Username: admin', 'Password: \*\*\*\*\*', and 'Authentication Method: Local DB' with a dropdown arrow. A 'login' button is below these fields. At the bottom right are the RMIT UNIVERSITY and V eRSI logos. A footer line reads 'Copyright © 2011 RMIT e-Research Office and Victorian eResearch Strategic Initiative'.

2. Click on your username shown in the top left-hand corner of the screen to access to MicroTardis administration tool.



The screenshot shows the MicroTARDIS administration home page after a successful login. The top navigation bar is identical to the login page. The main header area says 'Welcome, admin - [Log Out]', with 'admin' circled in yellow. Below this is a red heading 'Welcome to the MicroTARDIS pilot!'. A paragraph explains that MicroTARDIS is a repository for microscopy research data at RMMF, harvesting data every 5 minutes. Below this, it states 'These instruments are currently supported:' followed by a bulleted list: 'FEI Quanta 200: .tif' (with sub-item 'EDAX detector: .spc'), 'FEI Nova NanoSEM: .tif' (with sub-item 'EDAX detector: .spc'), and 'Philips XL30: .tif (in development)'. To the right of the list is a box titled 'Data harvesting status' containing: 'NovaNanoSEM: Active', 'Quanta200: Active', 'XL30: Active', and 'Last update: Thu Apr 12 15:15:28 EST 2012'.

3. Then you will see the following page after successful login, which is the home page of MicroTardis administration tool.



Django administration
Welcome, **admin**. Documentation / Change password / Log out

### Site administration

Auth	
Groups	<a href="#">Add</a> <a href="#">Change</a>
<b>Users</b>	<a href="#">Add</a> <a href="#">Change</a>
Djcelery	
Crontabs	<a href="#">Add</a> <a href="#">Change</a>
Intervals	<a href="#">Add</a> <a href="#">Change</a>
Periodic tasks	<a href="#">Add</a> <a href="#">Change</a>
Tasks	<a href="#">Change</a>
Workers	<a href="#">Add</a> <a href="#">Change</a>
Microtardis	
Datafile_harvests	<a href="#">Add</a> <a href="#">Change</a>
Datafile_hiddens	<a href="#">Add</a> <a href="#">Change</a>
Dataset_harvests	<a href="#">Add</a> <a href="#">Change</a>
Dataset_hiddens	<a href="#">Add</a> <a href="#">Change</a>
Experiment_hiddens	<a href="#">Add</a> <a href="#">Change</a>
Registration	
Registration profiles	<a href="#">Add</a> <a href="#">Change</a>
Sites	
Sites	<a href="#">Add</a> <a href="#">Change</a>

#### Recent Actions

##### My Actions

None available

## 4.1.2 Create Data in Three Related Tables

You will have to add objects into the following three different database tables to complete the process of creating a user account.

- **Users:** This is a Django built-in data model. Django comes with a user authentication system that handles user accounts, groups, permissions and cookie-based user sessions. MicroTardis/MyTardis uses it to benefit from Django's authentication system.
- **User profiles:** It is an extension to the Django standard user model to describe more information about users in MicroTardis/MyTardis.
- **User authentications:** This is used to specify user's authentication method. MyTardis supports multiple authentication mechanisms: VBL, LDAP, local DB, and Shibboleth. MicroTardis uses local DB authentication by default.

The following three subsections will show you how to create these data objects step-by-step.

### Add a User

1. First of all, find **Users** in **Auth** box on your administration home page; then click on it.

Django administration Welcome, **admin**. [Documentation](#) / [Change password](#) / [Log out](#)

### Site administration

Auth	
Groups	<a href="#">Add</a> <a href="#">Change</a>
<b>Users</b>	<a href="#">Add</a> <a href="#">Change</a>

Recent Actions

My Actions  
None available

Djcelery	
Crontabs	<a href="#">Add</a> <a href="#">Change</a>
Intervals	<a href="#">Add</a> <a href="#">Change</a>
Periodic tasks	<a href="#">Add</a> <a href="#">Change</a>
Tasks	<a href="#">Change</a>
Workers	<a href="#">Add</a> <a href="#">Change</a>

Microtardis	
Datafile_ harvests	<a href="#">Add</a> <a href="#">Change</a>
Datafile_ hiddens	<a href="#">Add</a> <a href="#">Change</a>
Dataset_ harvests	<a href="#">Add</a> <a href="#">Change</a>
Dataset_ hiddens	<a href="#">Add</a> <a href="#">Change</a>
Experiment_ hiddens	<a href="#">Add</a> <a href="#">Change</a>

Registration	
Registration profiles	<a href="#">Add</a> <a href="#">Change</a>

Sites	
Sites	<a href="#">Add</a> <a href="#">Change</a>

2. Then you will see the following page showing **Users** data table. Click on **Add user** button in the top right-hand corner of the screen.

Django administration Welcome, **admin**. [Documentation](#) / [Change password](#) / [Log out](#)

Home > Auth > Users

### Select user to change

Search:

Action:   0 of 1 selected

<input type="checkbox"/>	Username	E-mail address	First name	Last name	Staff status
<input type="checkbox"/>	admin				<input checked="" type="checkbox"/>

1 user

**Add user**

**Filter**

**By staff status**

☐ All  
☐ Yes  
☐ No

**By superuser status**

☐ All  
☐ Yes  
☐ No

**By active**

☐ All  
☐ Yes  
☐ No

3. Input **username** and assign a **password** to the user. Click on **Save** button in the bottom right-hand corner of screen to create an user object.

**Django administration** Welcome, **admin**. Documentation / Change password / Log out

Home > Auth > Users > Add user

### Add user

First, enter a username and password. Then, you'll be able to edit more user options.

<b>Username:</b>	<input type="text" value="E12345"/>
	<small>Required. 30 characters or fewer. Letters, digits and @/./+/-/_ only.</small>
<b>Password:</b>	<input type="password" value="*****"/>
<b>Password confirmation:</b>	<input type="password" value="*****"/>
	<small>Enter the same password as above, for verification.</small>

Save and add another Save and continue editing **Save**

#### 4. Continue to edit **user's** personal information,

**Django administration** Welcome, **admin**. Documentation / Change password / Log out

Home > Auth > Users > E12345

### Change user

History View on site →

<b>Username:</b>	<input type="text" value="E12345"/>
	<small>Required. 30 characters or fewer. Letters, digits and @/./+/-/_ only.</small>
<b>Password:</b>	<input type="password" value="*****"/>
	<small>Use "[a-z0-9]{30}" or use the <a href="#">change password form</a>.</small>

#### Personal info

First name:	<input type="text" value="James"/>
Last name:	<input type="text" value="Bond"/>
E-mail address:	<input type="text" value="James.Bond@007.fake.address"/>

#### Permissions

☒ **Active**  
Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

☐ **Staff status**  
Designates whether the user can log into this admin site.

☐ **Superuser status**  
Designates that this user has all permissions without explicitly assigning them.

5. (OPTIONAL) Tick the checkbox of **Staff status** to mark the user as a *staff* member if you would like to create a MicroTardis site administrator account. Once the user is marked as a *staff* member, and thus is allowed access to the administration interface.

**Permissions**

☒ **Active**  
Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

☒ **Staff status**  
Designates whether the user can log into this admin site.

☐ **Superuser status**  
Designates that this user has all permissions without explicitly assigning them.

Hold down "Control", or "Command" on a Mac, to select more than one.

User permissions:

**Available user permissions**  
  
 admin | log entry | Can add log entry  
 admin | log entry | Can change log entry  
 admin | log entry | Can delete log entry  
 auth | group | Can add group  
 auth | group | Can change group  
 auth | group | Can delete group  
 auth | message | Can add message  
 auth | message | Can change message  
 auth | message | Can delete message  
 auth | permission | Can add permission  
 auth | permission | Can change permission  
 auth | permission | Can delete permission  
 auth | user | Can add user  
 auth | user | Can change user

Choose all

**Chosen user permissions**  
 Select your choice(s) and click +

Clear all

6. (OPTIONAL) Tick the checkbox of **Superuser status** to mark the user as a *superuser* member if you would like to create a super account with full access and data management control. Once the user is marked as a *superuser* member, and thus is assigned all permissions.

**Permissions**

☒ **Active**  
Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

☐ **Staff status**  
Designates whether the user can log into this admin site.

☒ **Superuser status**  
Designates that this user has all permissions without explicitly assigning them.

Hold down "Control", or "Command" on a Mac, to select more than one.

User permissions:

**Available user permissions**  
  
 admin | log entry | Can add log entry  
 admin | log entry | Can change log entry  
 admin | log entry | Can delete log entry  
 auth | group | Can add group  
 auth | group | Can change group  
 auth | group | Can delete group  
 auth | message | Can add message  
 auth | message | Can change message  
 auth | message | Can delete message  
 auth | permission | Can add permission  
 auth | permission | Can change permission  
 auth | permission | Can delete permission  
 auth | user | Can add user  
 auth | user | Can change user

Choose all

**Chosen user permissions**  
 Select your choice(s) and click +

Clear all

7. Assign **basic access permissions** for a regular user. Four basic user permissions are recommended,

- **Can add experiment:** allow users to create their own experiments.
- **Can change experiment:** allow users to edit their own experiments.
- **Can change experiment acl:** allow users to manage experiment access.
- **Can change user authentication:** allow users to manage their passwords.

The following three steps will show you how to assign access permissions to users.

- (a) Select these three basic access permissions from **Available user permissions** box (press *Ctrl* for multiple selection),

lons

**Permissions**

☒ **Active**  
Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

☐ **Staff status**  
Designates whether the user can log into this admin site.

☐ **Superuser status**  
Designates that this user has all permissions without explicitly assigning them.

**User permissions:**

Hold down "Control", or "Command" on a Mac, to select more than one.

**Available user permissions**

Search:

- admin | log entry | Can add log entry
- admin | log entry | Can change log entry
- admin | log entry | Can delete log entry
- auth | group | Can add group
- auth | group | Can change group
- auth | group | Can delete group
- auth | message | Can add message
- auth | message | Can change message
- auth | message | Can delete message
- auth | permission | Can add permission
- auth | permission | Can change permission
- auth | permission | Can delete permission
- auth | user | Can add user
- auth | user | Can change user

**Chosen user permissions**

Select your choice(s) and click +

- tardis\_portal | experiment | Can add experiment
- tardis\_portal | experiment | Can change experiment
- tardis\_portal | experiment act | Can change experim

**Choose all** **Clear all**

8. Leave everything else as default values, then click on **Save** button on the same page to finish editing user's info and basic permissions.

**Important dates**

**Last login:** Date: 2012-04-12 Today | Time: 17:03:22 Now |

**Date joined:** Date: 2012-04-12 Today | Time: 17:03:22 Now |

**Groups**

Groups: +

In addition to the permissions manually assigned, this user will also get all permissions granted to each group he/she is in. Hold down "Control", or "Command" on a Mac, to select more than one.

**Delete** **Save and add another** **Save and continue editing** **Save**

## Add a User Profile

1. Go back to MicroTardis administration home page. Find **User profiles** in **Tardis\_Portal** box; then click on it.

<b>Sites</b>	
Sites	<a href="#">Add</a> <a href="#">Change</a>
<b>Tardis_Portal</b>	
Author_experiments	<a href="#">Add</a> <a href="#">Change</a>
Datafile parameter sets	<a href="#">Add</a> <a href="#">Change</a>
Datafile parameters	<a href="#">Add</a> <a href="#">Change</a>
Dataset parameter sets	<a href="#">Add</a> <a href="#">Change</a>
Dataset parameters	<a href="#">Add</a> <a href="#">Change</a>
Dataset_files	<a href="#">Add</a> <a href="#">Change</a>
Datasets	<a href="#">Add</a> <a href="#">Change</a>
Experiment acis	<a href="#">Add</a> <a href="#">Change</a>
Experiment parameter sets	<a href="#">Add</a> <a href="#">Change</a>
Experiment parameters	<a href="#">Add</a> <a href="#">Change</a>
Experiments	<a href="#">Add</a> <a href="#">Change</a>
Free text search fields	<a href="#">Add</a> <a href="#">Change</a>
Group admins	<a href="#">Add</a> <a href="#">Change</a>
Parameter names	<a href="#">Add</a> <a href="#">Change</a>
Schemas	<a href="#">Add</a> <a href="#">Change</a>
Tokens	<a href="#">Add</a> <a href="#">Change</a>
User authentications	<a href="#">Add</a> <a href="#">Change</a>
<b>User profiles</b>	<a href="#">Add</a> <a href="#">Change</a>

- Then you will see the following page showing **User profiles** data table. Click on **Add user profile** button in the top right-hand corner of the screen.

**Django administration** Welcome, **admin**. Documentation / Change password / Log out

Home > Tardis\_portal > User profiles

**Select user profile to change**

Action:   0 of 1 selected

<input type="checkbox"/>	User profile
<input type="checkbox"/>	admin

1 user profile

[Add user profile](#) **+**

- Choose the **User object** that you just created, then click on **Save** button in the bottom right-hand corner of screen to create an user profile object.

**Django administration** Welcome, **admin**. Documentation / Change password / Log out

Home > Tardis\_portal > User profiles > Add user profile

**Add user profile**

User:  [+](#)

☒ IsDjangoAccount

## Add a User Authentication

- Go back to MicroTardis administration home page. Find **User authentications** in **Tardis\_Portal** box; then click on it.

Sites	
Sites	<a href="#">+ Add</a> <a href="#">Change</a>
Tardis_Portal	
Author_ experiments	<a href="#">+ Add</a> <a href="#">Change</a>
Datafile parameter sets	<a href="#">+ Add</a> <a href="#">Change</a>
Datafile parameters	<a href="#">+ Add</a> <a href="#">Change</a>
Dataset parameter sets	<a href="#">+ Add</a> <a href="#">Change</a>
Dataset parameters	<a href="#">+ Add</a> <a href="#">Change</a>
Dataset_files	<a href="#">+ Add</a> <a href="#">Change</a>
Datasets	<a href="#">+ Add</a> <a href="#">Change</a>
Experiment acfs	<a href="#">+ Add</a> <a href="#">Change</a>
Experiment parameter sets	<a href="#">+ Add</a> <a href="#">Change</a>
Experiment parameters	<a href="#">+ Add</a> <a href="#">Change</a>
Experiments	<a href="#">+ Add</a> <a href="#">Change</a>
Free text search fields	<a href="#">+ Add</a> <a href="#">Change</a>
Group admins	<a href="#">+ Add</a> <a href="#">Change</a>
Parameter names	<a href="#">+ Add</a> <a href="#">Change</a>
Schemas	<a href="#">+ Add</a> <a href="#">Change</a>
Tokens	<a href="#">+ Add</a> <a href="#">Change</a>
<b>User authentications</b>	<a href="#">+ Add</a> <a href="#">Change</a>
User profiles	<a href="#">+ Add</a> <a href="#">Change</a>

- Then you will see the following page showing **User authentications** data table. Click on **Add user authentication** button in the top right-hand corner of the screen.

Django administration Welcome, **admin**. [Documentation](#) / [Change password](#) / [Log out](#)

Home > Tardis\_portal > User authentications

**Select user authentication to change** [Add user authentication](#) +

Action:   0 of 1 selected

<input type="checkbox"/>	User authentication
<input type="checkbox"/>	admin - Local DB

1 user authentication

- Choose the **UserProfile** object that you just created, give a **Username**, and specify **AuthenticationMethod** as **localdb**. Then click on **Save** button in the bottom right-hand corner of screen to create an user authentication object.

Django administration Welcome, **admin**. [Documentation](#) / [Change password](#) / [Log out](#)

Home > Tardis\_portal > User authentications > Add user authentication

**Add user authentication**

**User Profile:**  [+](#)

**Username:**

**Authentication:**

## 4.2 Create New Groups

In Django, groups are a generic way of categorising users so site administrator can apply permissions to those users. A user can belong to any number of groups. A user in a group automatically has the permissions granted to that group. For example, if the group *Students* has the permission *can\_add\_experiment*, any user in that group will have that permission.

In MicroTardis/MyTardis, groups are also a convenient way to share experiments and the associated datasets and



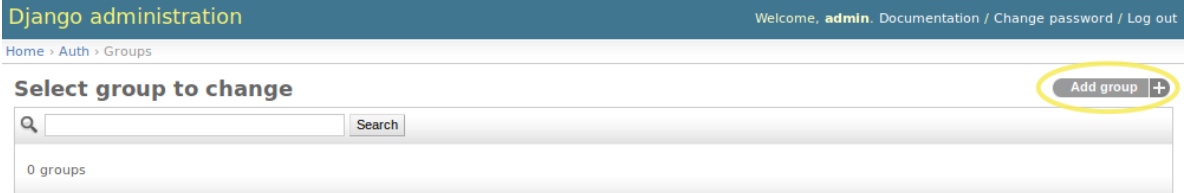
datafiles with users in the same group.

Please note that regular users don't have permissions to create groups. By default, only site administrators and superusers can do this.

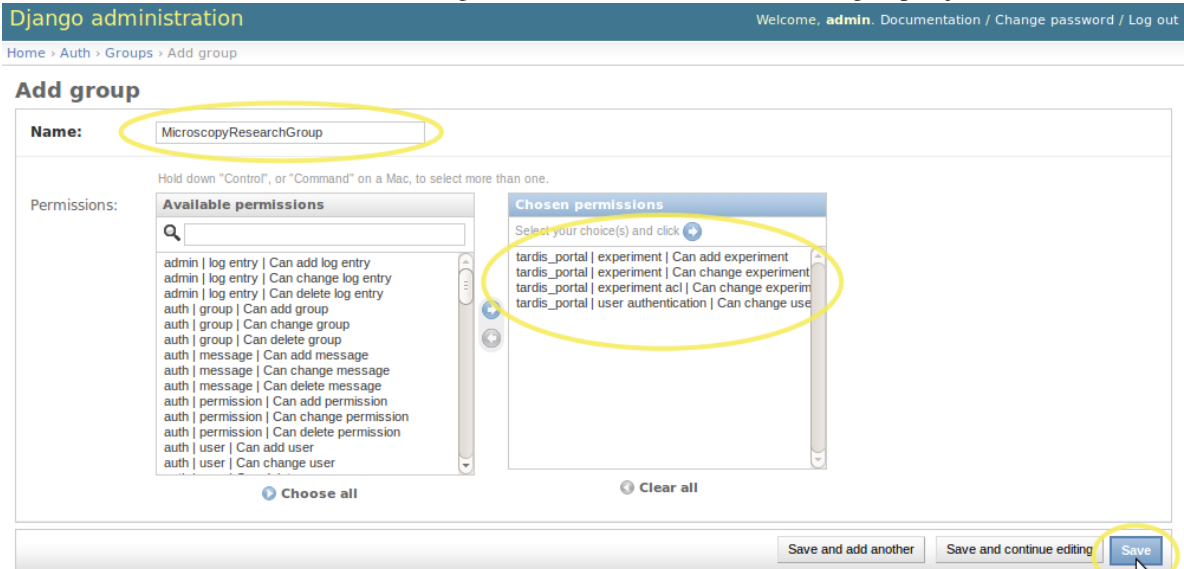
1. Go back to MicroTardis administration home page. Find **Groups** in **Auth** box; then click on it.



2. Then you will see the following page showing **Groups** data table. Click on **Add group** button in the top right-hand corner of the screen.



3. Choose a group **Name** and assign **Permissions** to this group. Here the four basic access permissions are assigned to the group. You can give more or some other permissions to your groups depending on your real needs. Then click on **Save** button in the bottom right-hand corner of screen to create a group object.



## 4.3 Assign Group Owners

All users which are owners of groups have the permission to edit the membership. You have two ways to assign an administrator/owner to a group,

1. via MicroTardis web portal
2. via MicroTardis administration interface

## 4.4 Manage Group Members

## 4.5 Experiment Access Controls

## 4.6 Publish Experiment

# INDICES AND TABLES

- *genindex*
- *modindex*
- *search*