HANDS ON SESSION

M. Eguiraun
14th March 2019

MAX IV

Outline

- Hardware Repository
- MXCuBE
- Exercises

MXCuBE

- Macromolecular Xtallography Customized Beamline Environment
  - Started in 2005 at ESRF
  - Beamline control and data acquisition platform for running MX experiments

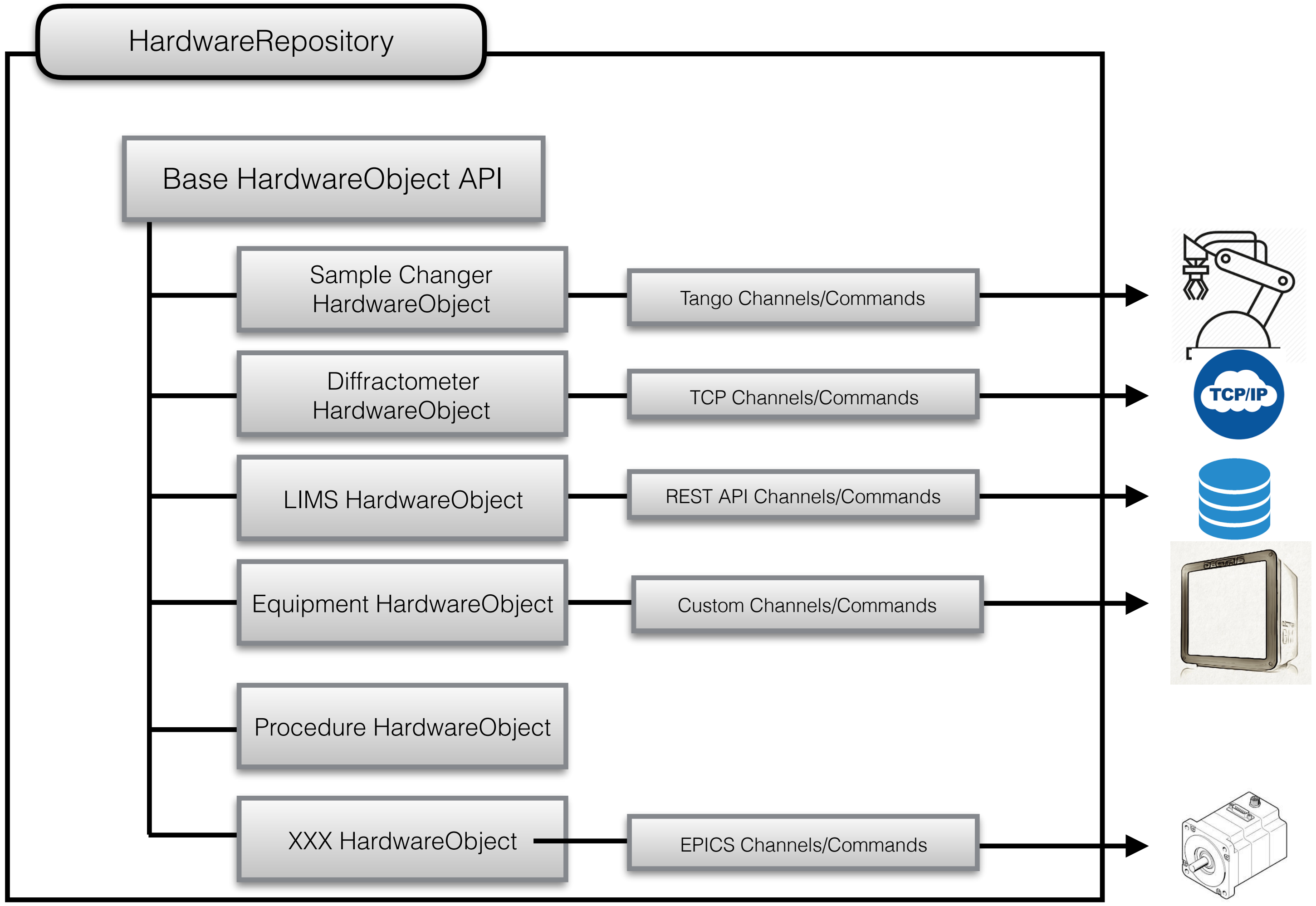
- Supported by the following partners: ESRF, Soleil, MAX IV, HZB, EMBL, Global Phasing Ltd, DESY, ALBA (LNLS)
  - Common solution for scientist
  - Already tested software & builtin experience
  - *Quick* setup

MXCuBE - Main Features

- Customizable interface for each beamline/facility (PyQt bricks, web)
- Hide the complexity of the Hardware to the user (and to the developers…) thanks to the usage of the **HardwareObjects**
- Reuse of existing code for different beamlines
  - same or similar hardware
  - same or similar experimental procedures
- A huge builtin experience (many years + many people + many beamlines)
- Currently QT and Web versions

MXCuBE - HardwareRepository

- Hardware Abstraction Layer
- It acts as a container/Pool of single python objects (called Hardware Objects)
  - The information necessary for a hardware object to operate a physical device. Supported protocols: Tango, Spec, Exporter, Sardana, EPICS



MXCuBE - HardwareObjects

- A HO is not only hardware! Procedures/sequences etc
- Link between devices and the graphical interface
- Configured through xml files
- emitting signals to others HOs, graphical elements
- Hardware mockups available

```
<device class="MicrodiffMotor">
   <username>Omega</username>
   <exporter_address>130.235.94.124:9001</exporter_address>
   <motor_name>Omega</motor_name>
   <unit>1e-3</unit>
</device>
```

udiff_omega.xml

```
class MicrodiffMotor(Device):

    def init(self):
        self.position_attr = self.addChannel({"type":"exporter", "name":"position"  }, self.motor_name)

    def getPosition(self):
        return self.position_attr.getValue()

    def move(self, absolutePosition)
        self.position_attr.setValue(absolutePosition)
```

MicrodiffMotor.py

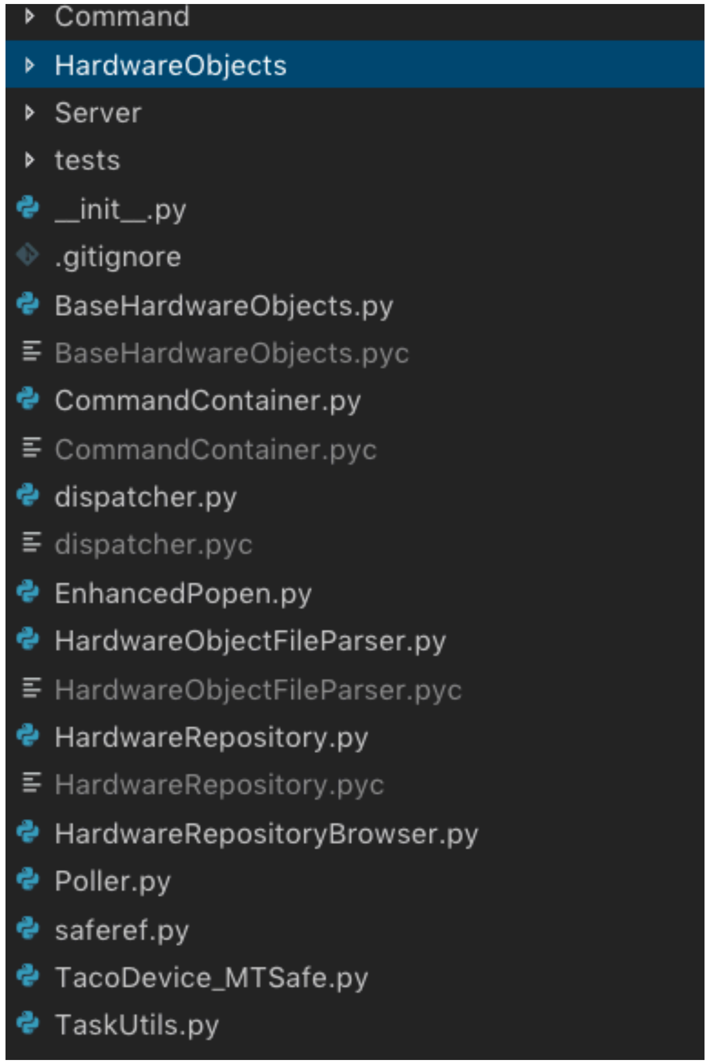
MXCuBE - HardwareObjects

```
<!-- Example beamline setup file -->

<object class="BeamlineSetup" role="BeamlineSetup">
  <!-- Objects directly associatd with hardware -->
  <object href="/transmission-mockup" role="transmission"/>
  <object href="/minidiff" role="diffractometer"/>
  <object href="/cats" role="sample_changer"/>
  <object href="/spec_mxCuBE/res" role="resolution"/>

  <!-- Software (abstract) concepts -->
  <object href="/shape-history" role="shape_history"/>
  <object href="/session" role="session"/>
  <object href="/lims" role="lims_client"/>
  <object href="/edna_config" role="data_analysis"/>
  <!--<object href="/workflow-mockup" role="workflow"/> -->

  <!-- Procedures and routines -->
  <object href="/energyscan" role="energy"/>
  <object href="/mxcollect" role="collect"/>

  <!-- Is it possible to change the beam wavelentgh.
       Should perhaps be associated with the diffractometer -->
  <tunable_wavelength>True</tunable_wavelength>

  <!-- Disables or enables the number of passes input box, used
  for acquisitions.-->
  <disable_num_passes>False</disable_num_passes>

  <!-- Should be moved to a detector object in the future -->
  <detector>
    <manufacturer>MAR</manufacturer>
    <type>marccd</type>
    <model>marmosaic</model>
    <px>0.07324</px>
    <py>0.07324</py>
    <cansum>no</cansum>
    <has_shutterless>False</has_shutterless>
  </detector>
```

```
  <!-- Default values for an acquisition -->
  <default_acquisition_values>
    <exposure_time>10</exposure_time>
    <start_angle>0.0</start_angle>
    <range>1</range>
    <number_of_passes>1</number_of_passes>
    <start_image_number>1</start_image_number>
    <run_number>1</run_number>
    <overlap>0</overlap>
    <number_of_images>1</number_of_images>
    <detector_mode>1</detector_mode>
  </default_acquisition


  <!-- Default values for a characterization -->
  <default_characterisation_values>
    <exposure_time>5</exposure_time>
    <start_angle>0.0</start_angle>
    <range>1</range>
    <number_of_passes>1</number_of_passes>
    <start_image_number>1</start_image_number>
    <run_number>1</run_number>
    <overlap>0</overlap>
    <number_of_images>1</number_of_images>
    <detector_mode>1</detector_mode>
  </default_characterisation_values>
</object>
```

MXCuBE - HardwareObjects

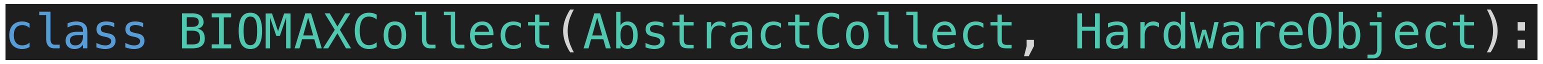
- Important HO
  - Collect, Diffractometer,
  - This is the main entry point: **BeamlineSetup**
- Folder structure…
- Specific folder for each facility
- Try to inherit as much as possible



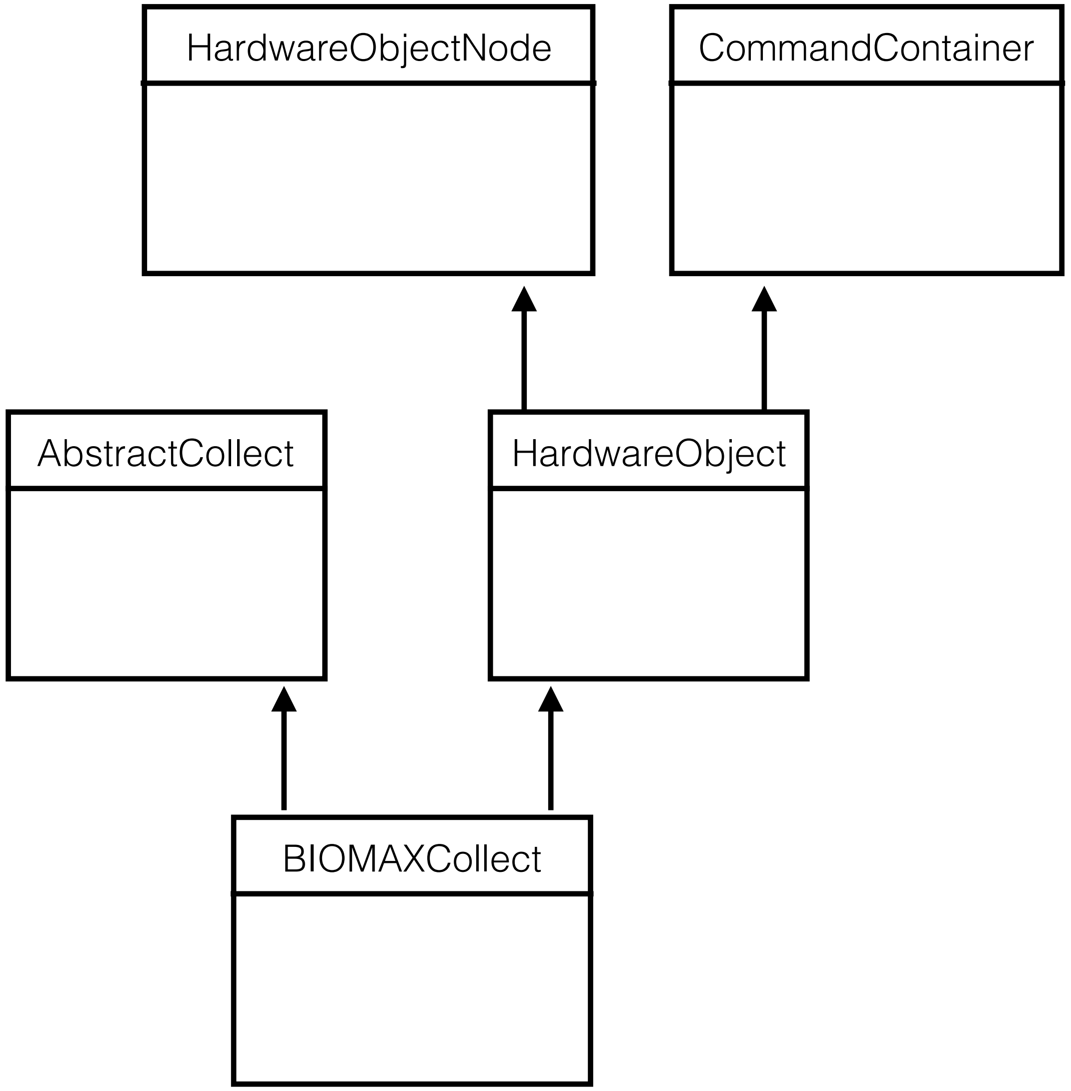
```
Command
HardwareObjects
Server
tests
__init__.py
.gitignore
BaseHardwareObjects.py
BaseHardwareObjects.pyc
CommandContainer.py
CommandContainer.pyc
dispatcher.py
dispatcher.pyc
EnhancedPopen.py
HardwareObjectFileParser.py
HardwareObjectFileParser.pyc
HardwareRepository.py
HardwareRepository.pyc
HardwareRepositoryBrowser.py
Poller.py
saferef.py
TacoDevice_MTSafe.py
TaskUtils.py
```

MXCuBE - HardwareObjects

- Inheritance example



```
class BIOMAXCollect(AbstractCollect, HardwareObject):
```



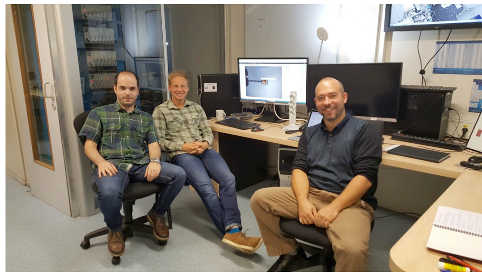
MXCuBE - HardwareObjects

- Let’s play a bit

```
from HardwareRepository import HardwareRepository as hwr
hwr_dir = '../test/HardwareObjectsMockup.xml/'
hwr = hwr.HardwareRepository(hwr_dir)
hwr.connect()
dtox = hwr.getHardwareObject(‘dtox’)
dtox.getPosition()
dtox.move(100)
dtox.getPosition()
```

MXCuBE 3

- Beamline control and data acquisition as web application
- Modern technologies
- Future easier integration and maintenance
- Remote access in a more *natural* way
- Reuse existing HardwareObjects
- Challenges:
  - Refactor existing code, remove dependencies
  - New design for the user interface
  - Decoupling logic and interface: any client possible



https://github.com/mxcube/mxcube3

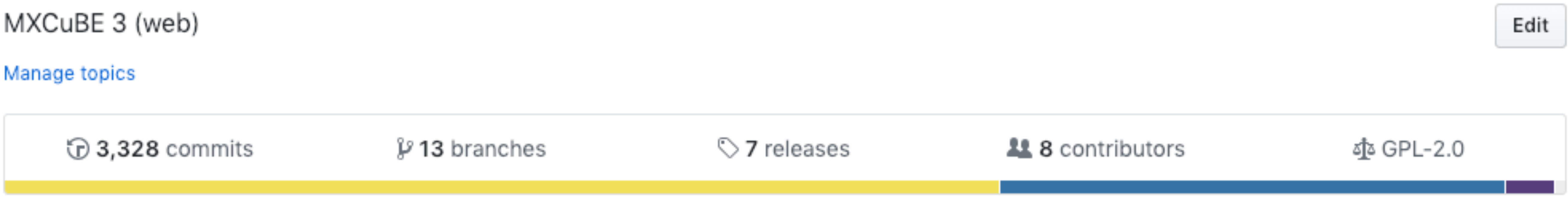
MXCuBE 3

- Under development
  - kickoff meeting in September 2015
  - v 3.0.2
- In production in MAX IV, ESRF, Elettra (Roberto?)
- Tests in Soleil?
- Still issues few to be solved

Latest release

v3.0.2

a0e49fb



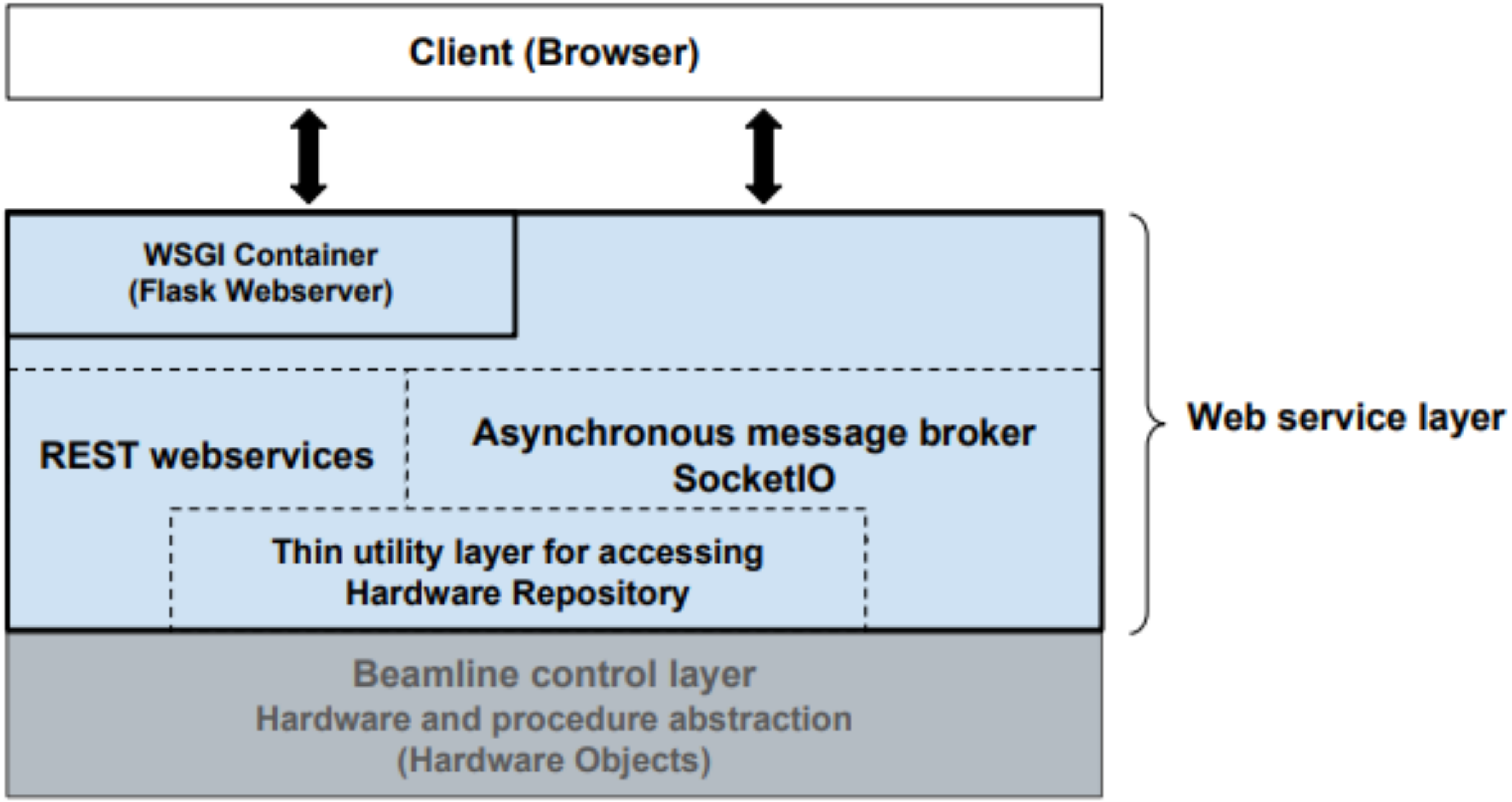
MXCuBE 3 (web)

3,328 commits · 13 branches · 7 releases · 8 contributors · GPL-2.0

https://github.com/mxcube/mxcube3

MXCuBE 3

- REST API backend
- Websockets for forwarding events (SocketIO)
- Thin layer for adapting HO and mxcube3 communication

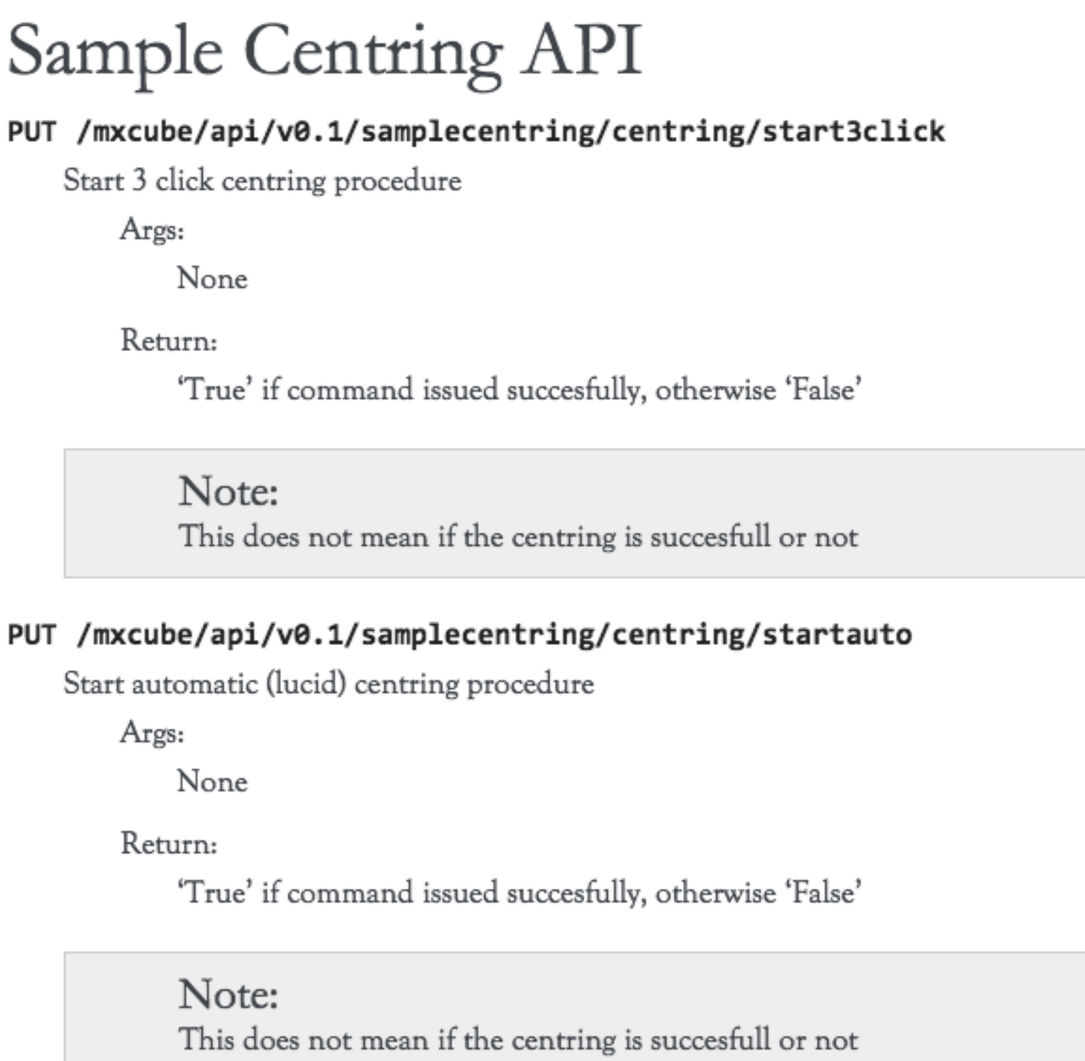


Backend

- Python **Flask** microwebframework:
  - web server made simple
  - extensions (database, login, …)
  - easily adaptable to your needs while scalable
  - big community
- http request **API**: rest like (but probably not fully rest)
  - an url for each function
  - Simple to add new features without changing existing ones
- Flask **socketio** for sending HO messages
  - server-client bi-directional communication, websocket based
- Reuse the existing Hardware Repository

Http requests

- API for the calls from client to server (*GET*, *PUT*, *POST, DELETE*)
- Decoupling the server and the client
- Should be easy to understand by the client
  - ➡ (http://example.com/queue/4/12/execute)



Sample Centring API

`PUT /mxcube/api/v0.1/samplecentring/centring/start3click`

Start 3 click centring procedure

Args:

None

Return:

'True' if command issued succesfully, otherwise 'False'

Note:
This does not mean if the centring is succesfull or not

`PUT /mxcube/api/v0.1/samplecentring/centring/startauto`

Start automatic (lucid) centring procedure

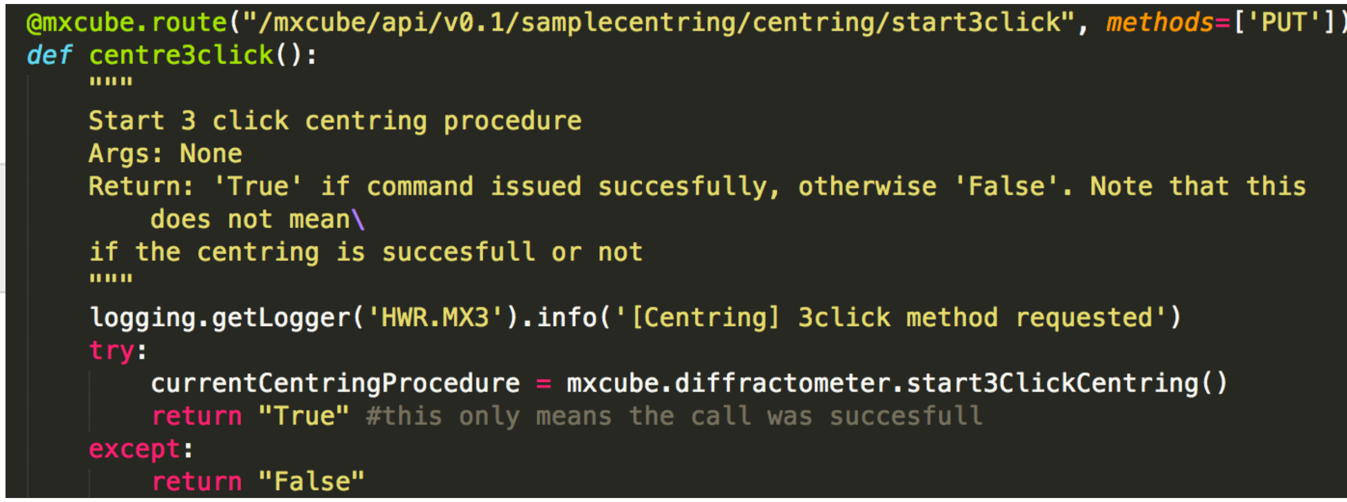
Args:

None

Return:

'True' if command issued succesfully, otherwise 'False'

Note:
This does not mean if the centring is succesfull or not



```
@mxcube.route("/mxcube/api/v0.1/samplecentring/centring/start3click", methods=['PUT'])
def centre3click():
    """
    Start 3 click centring procedure
    Args: None
    Return: 'True' if command issued succesfully, otherwise 'False'. Note that this
        does not mean\
    if the centring is succesfull or not
    """
    logging.getLogger('HWR.MX3').info('[Centring] 3click method requested')
    try:
        currentCentringProcedure = mxcube.diffractometer.start3ClickCentring()
        return "True" #this only means the call was succesfull
    except:
        return "False"
```

Adding new devices

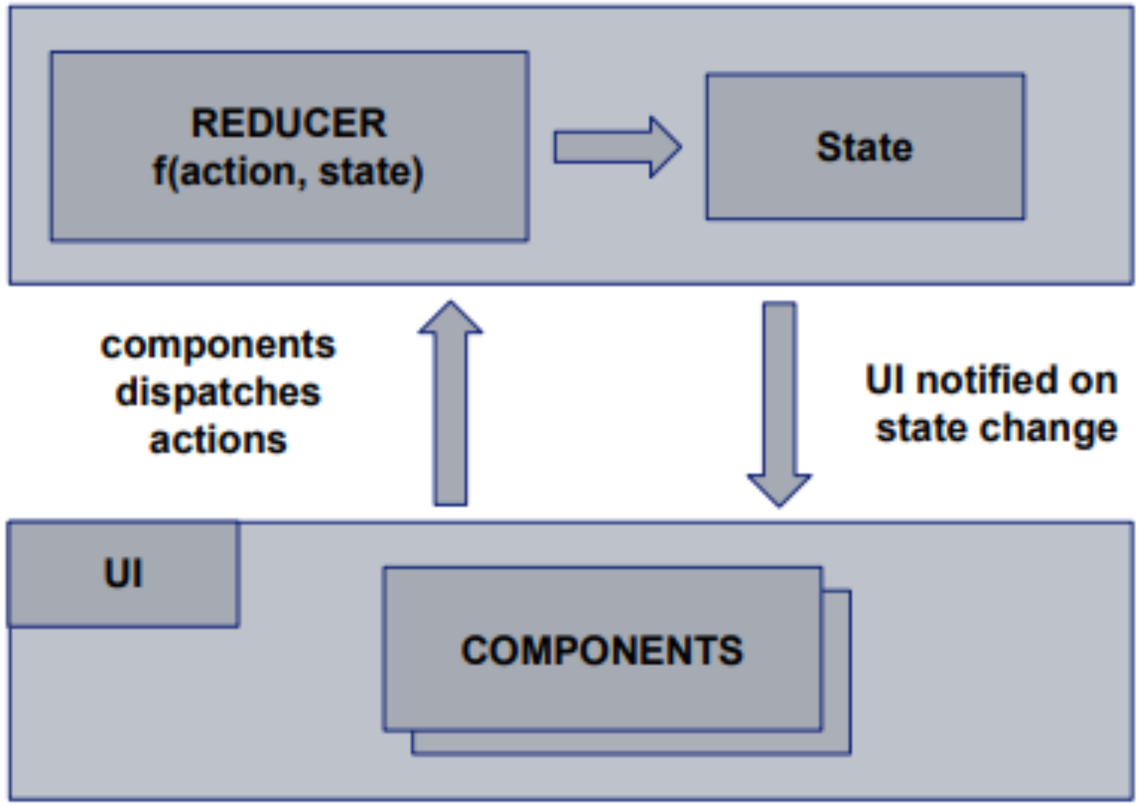
- Existing HO framework makes easy the addition of new devices
- Clear decoupling
- Steps (roughly):
    - Write your new Hardware Object
    - Configure it (xml file, specific address, range, etc.)
    - Does the current http api support the new HO?
        - if not: add new routes
    - Tell the client how to make use of the api

Frontend REACT

- **Javascript/React** library (Facebook)
- Only for the user interface (the V in MVC)
- Virtual html DOM kept as internal state
  - Different components programmed independently
- Widgets like in traditional UI development
  - Called **components**
- Reusing existing code when the layout changes
- Express the UI in a markup language called JSX (~html + javascript)

Frontend REDUX

- **Redux** application architecture/pattern
  - Predictable state container for JavaScript apps …
  - Unidirectional data flow, easy debugging
  - Changes on the internal state in a single place

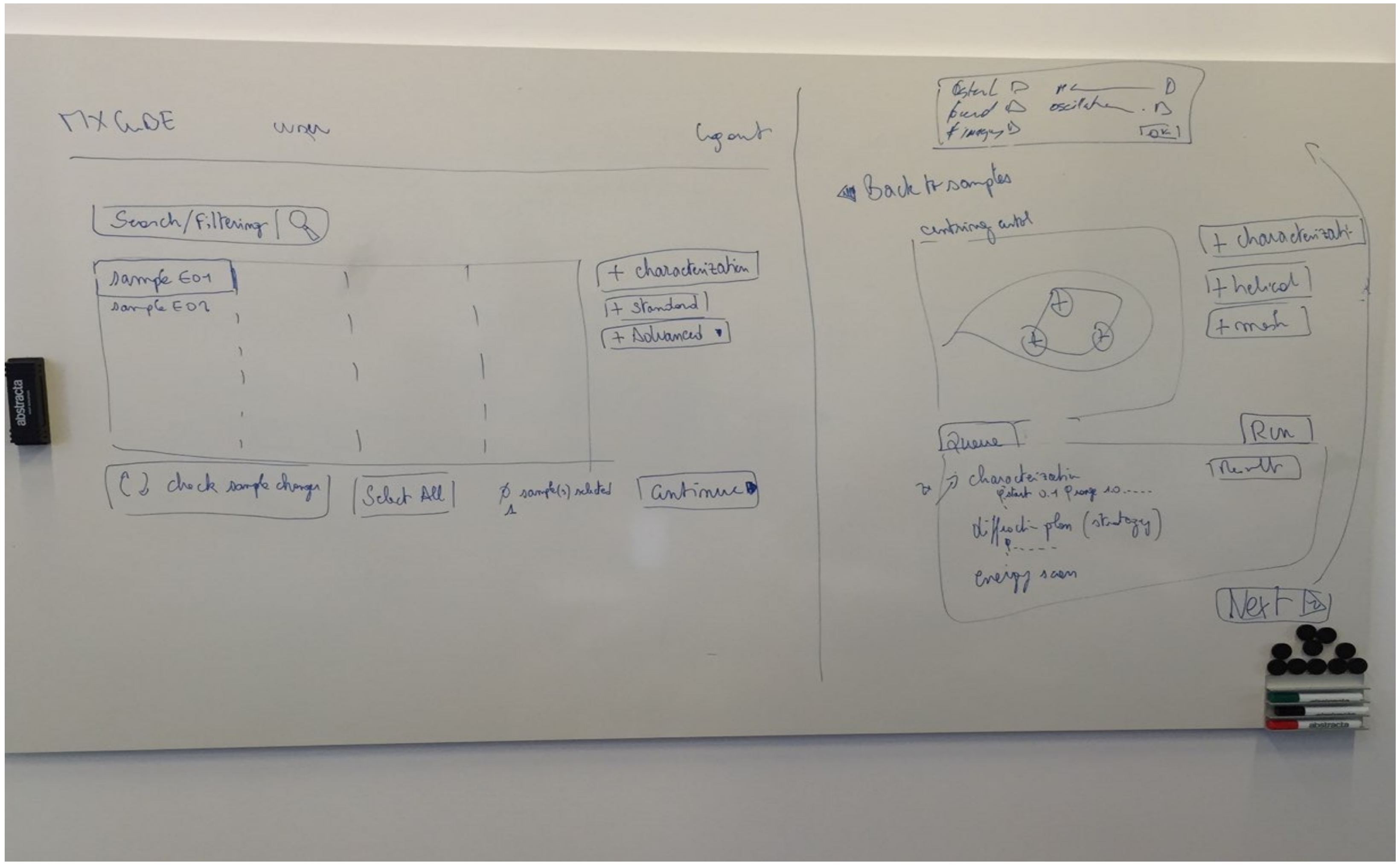


Layout

- A main objetive was identified
  - Improve the user experience

- And for that it is useful to
  - Have a clean interface
  - Use modern web technologies
  - Learn current usage and feedback

Layout - first sketch

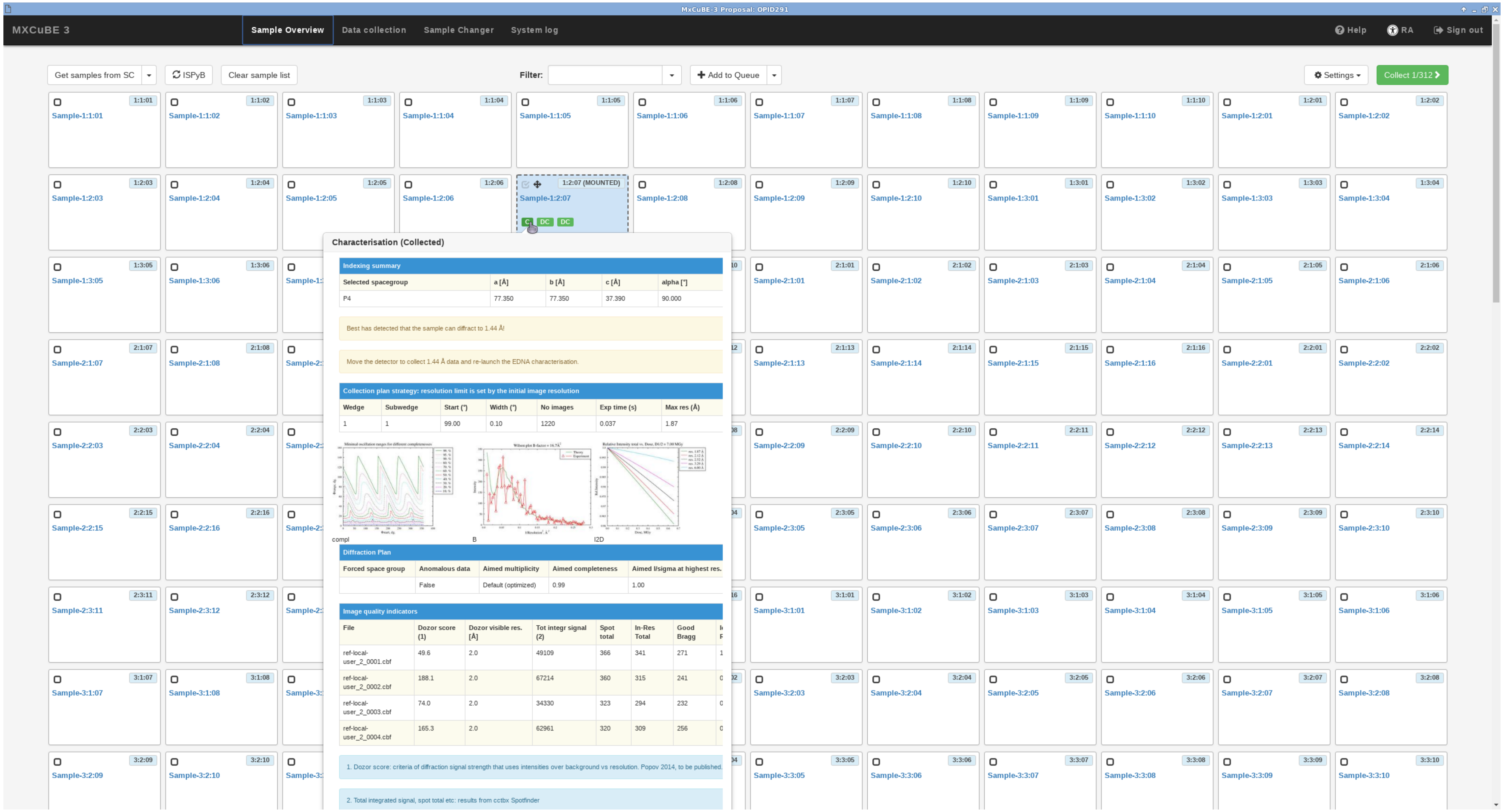


MAXIV-ESRF Sep. 2015

- Experiment configuration in a batch like mode
  - All available samples
- Experiment management for each sample
  - centring mechanism
  - should also be automatic and transparent for the user

Transitions between views to be defined

Layout - Today



Characterisation (Collected)

Indexing summary

Selected spacegroup	a [Å]	b [Å]	c [Å]	alpha [°]
P4	77.350	77.350	37.390	90.000

Best has detected that the sample can diffract to 1.44 Å!

Move the detector to collect 1.44 Å data and re-launch the EDNA characterisation.

Collection plan strategy: resolution limit is set by the initial image resolution

Wedge	Subwedge	Start (°)	Width (°)	No images	Exp time (s)	Max res (Å)
1	1	99.00	0.10	1220	0.037	1.87

Diffraction Plan

Forced space group	Anomalous data	Aimed multiplicity	Aimed completeness	Aimed I/sigma at highest res.
	False	Default (optimized)	0.99	1.00

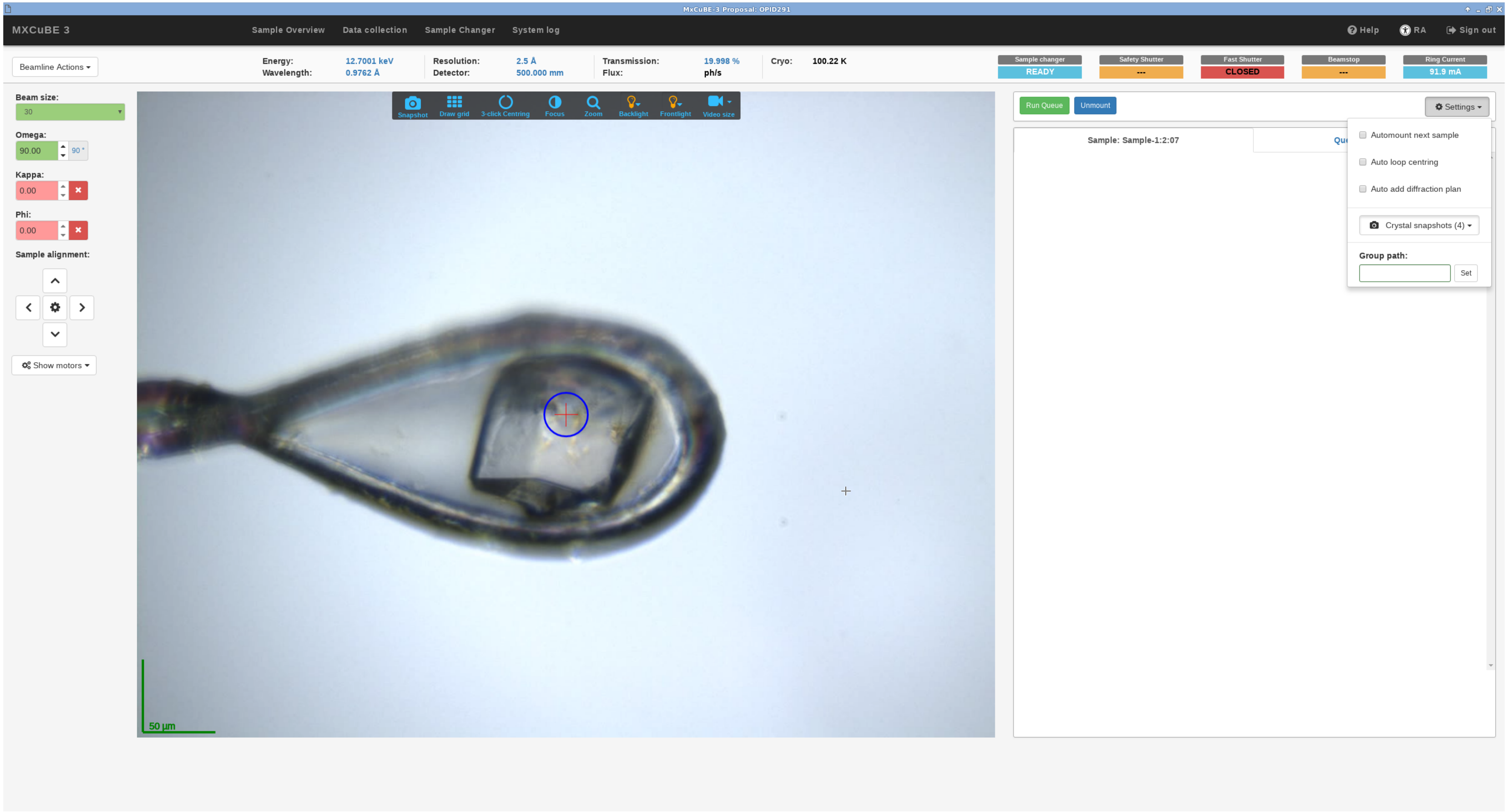
Image quality indicators

File	Dozor score (1)	Dozor visible res. [Å]	Tot integr signal (2)	Spot total	In-Res Total	Good Bragg	I F
ref-local-user_2_0001.cbf	49.6	2.0	49109	366	341	271	1
ref-local-user_2_0002.cbf	188.1	2.0	67214	360	315	241	0
ref-local-user_2_0003.cbf	74.0	2.0	34330	323	294	232	0
ref-local-user_2_0004.cbf	165.3	2.0	62961	320	309	256	0

1. Dozor score: criteria of diffraction signal strength that uses intensities over background vs resolution. Popov 2014, to be published.

2. Total integrated signal, spot total etc: results from cctbx Spotfinder

Layout - Today



MxCuBE-3 Proposal: OPID291

Energy: 12.7001 keV
Wavelength: 0.9762 Å
Resolution: 2.5 Å
Detector: 500.000 mm
Transmission: 19.998 %
Flux: ph/s
Cryo: 100.22 K

Sample changer: READY
Safety Shutter: ---
Fast Shutter: CLOSED
Beamstop: ---
Ring Current: 91.9 mA

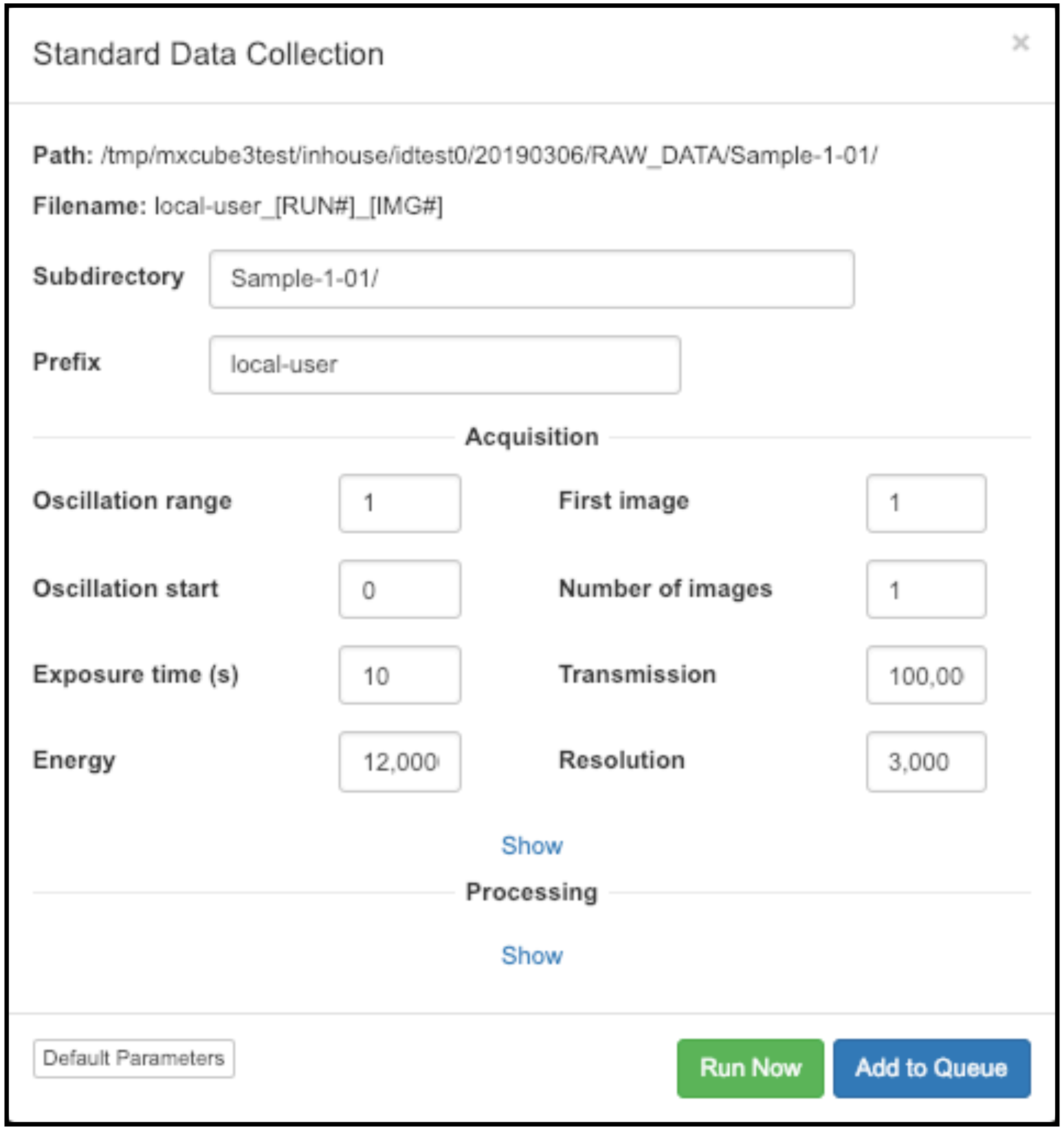
Beam size: 30
Omega: 90.00
Kappa: 0.00
Phi: 0.00

Sample: Sample-1:2:07

□ Automount next sample
□ Auto loop centring
□ Auto add diffraction plan

Group path:

DataCollections



Standard Data Collection

Path: /tmp/mxcube3test/inhouse/idtest0/20190306/RAW_DATA/Sample-1-01/

Filename: local-user_[RUN#]_[IMG#]

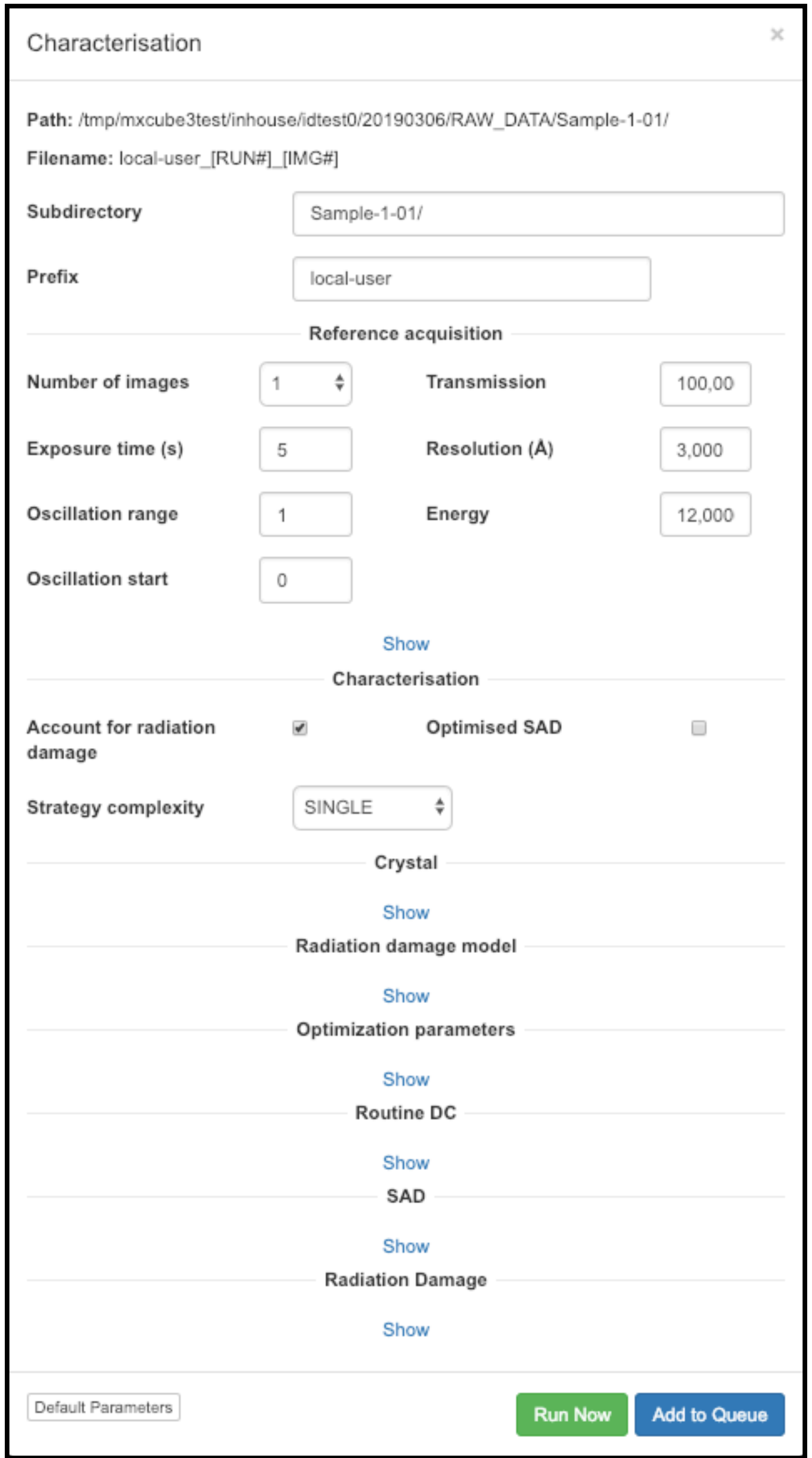
Subdirectory Sample-1-01/

Prefix local-user

Acquisition

Oscillation range	1	**First image**	1
Oscillation start	0	**Number of images**	1
Exposure time (s)	10	**Transmission**	100,00
Energy	12,000	**Resolution**	3,000

DataCollections



Characterisation

Path: /tmp/mxcube3test/inhouse/idtest0/20190306/RAW_DATA/Sample-1-01/

Filename: local-user_[RUN#]_[IMG#]

Subdirectory	Sample-1-01/
Prefix	local-user

Reference acquisition

Number of images	1	**Transmission**	100,00
Exposure time (s)	5	**Resolution (Å)**	3,000
Oscillation range	1	**Energy**	12,000
Oscillation start	0		

Characterisation

Account for radiation damage	☑	**Optimised SAD**	☐
Strategy complexity	SINGLE		

Crystal

Radiation damage model

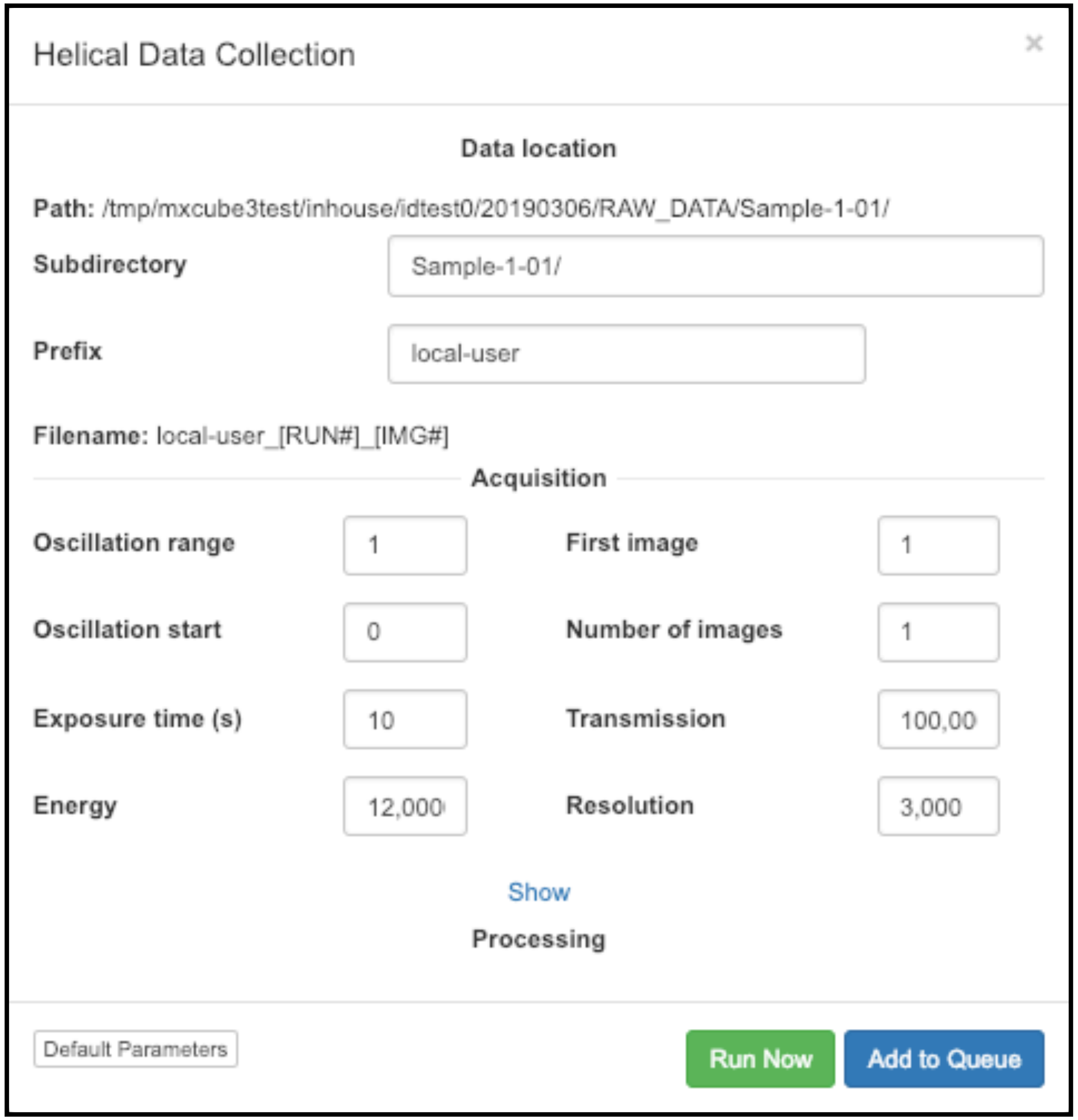
Optimization parameters

Routine DC

SAD

Radiation Damage

DataCollections



Helical Data Collection

Data location

Path: /tmp/mxcube3test/inhouse/idtest0/20190306/RAW_DATA/Sample-1-01/

Subdirectory	Sample-1-01/
Prefix	local-user

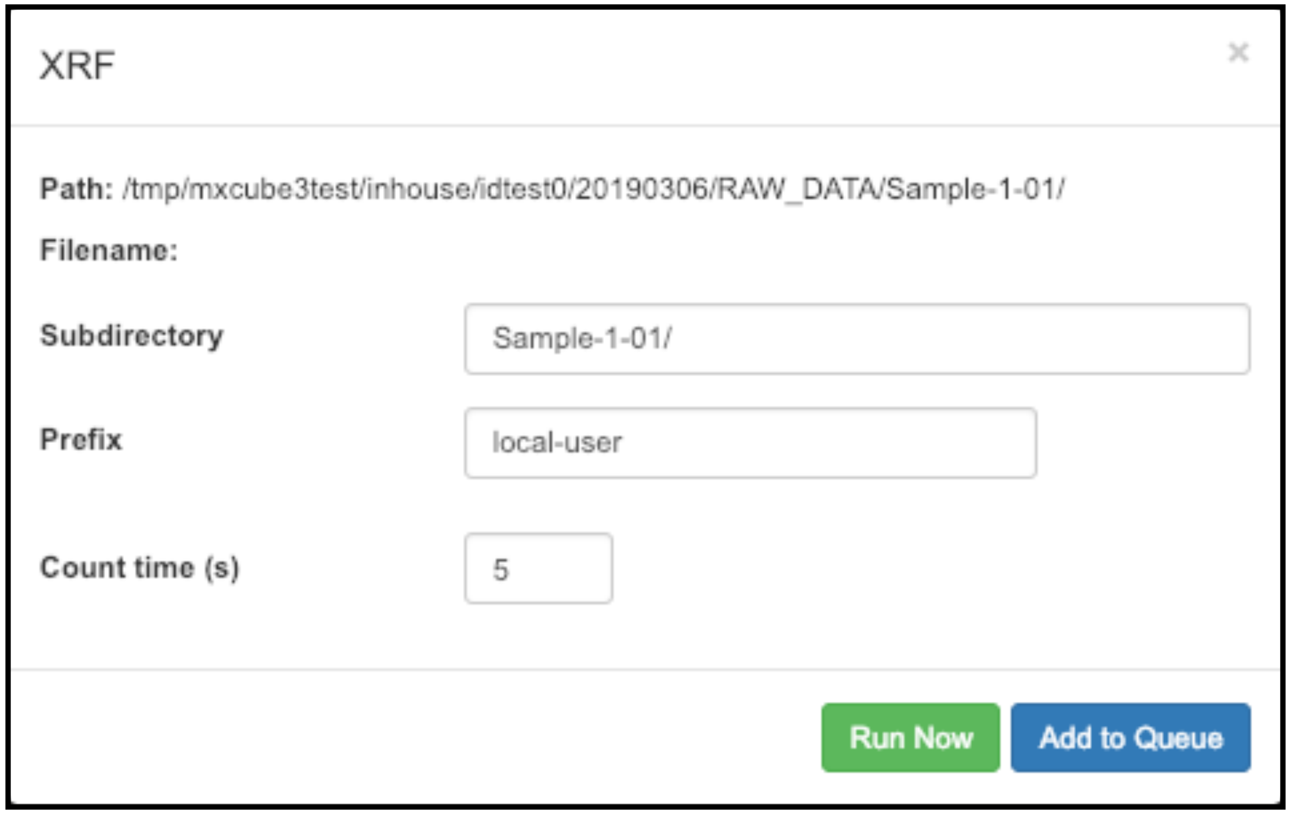
Filename: local-user_[RUN#]_[IMG#]

Acquisition

Oscillation range	1	**First image**	1
Oscillation start	0	**Number of images**	1
Exposure time (s)	10	**Transmission**	100,00
Energy	12,000	**Resolution**	3,000

Processing

DataCollections



XRF

Path: /tmp/mxcube3test/inhouse/idtest0/20190306/RAW_DATA/Sample-1-01/

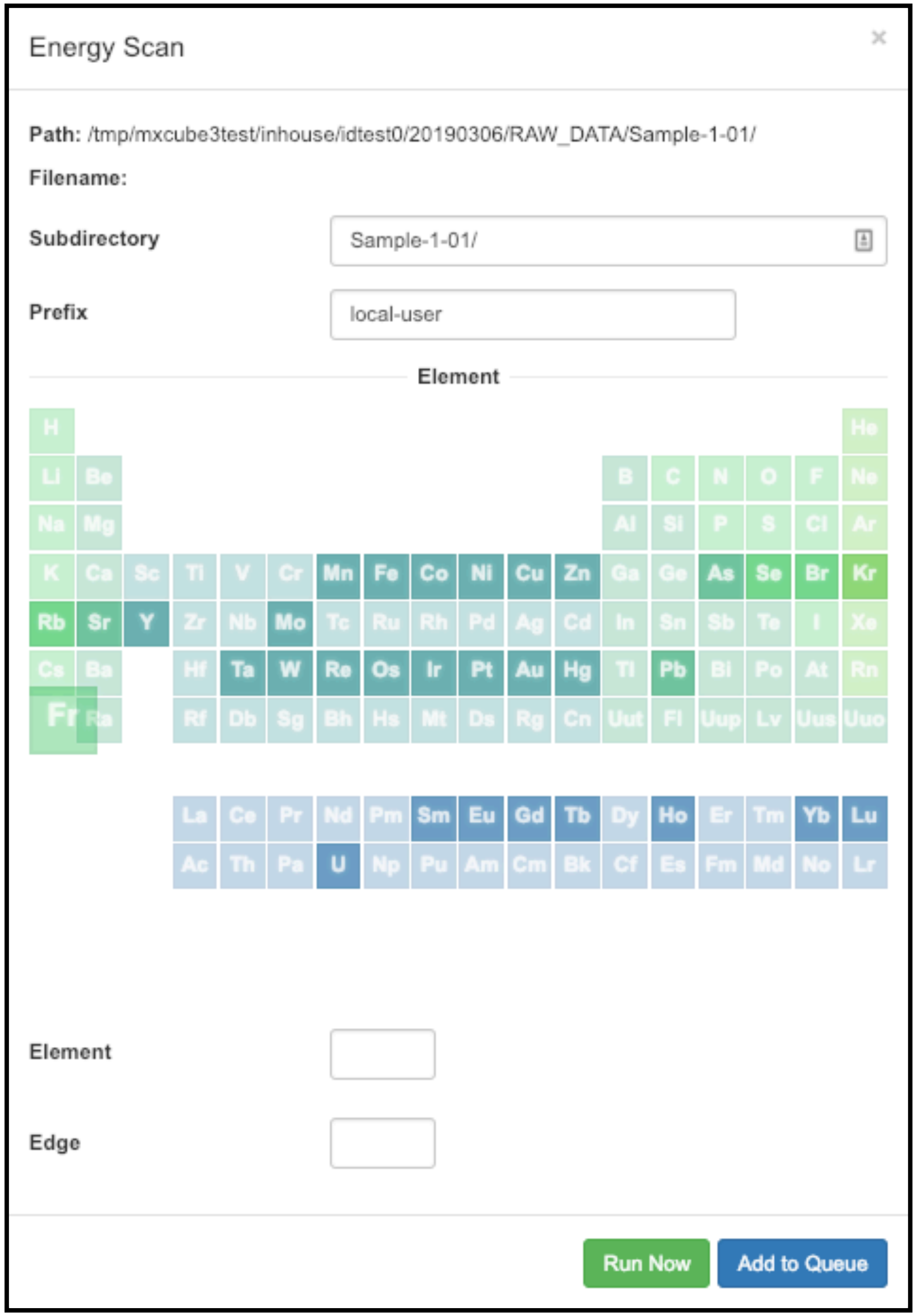
Filename:

Subdirectory Sample-1-01/

Prefix local-user

Count time (s) 5

DataCollections



Energy Scan

Path: /tmp/mxcube3test/inhouse/idtest0/20190306/RAW_DATA/Sample-1-01/

Filename:

Subdirectory Sample-1-01/

Prefix local-user

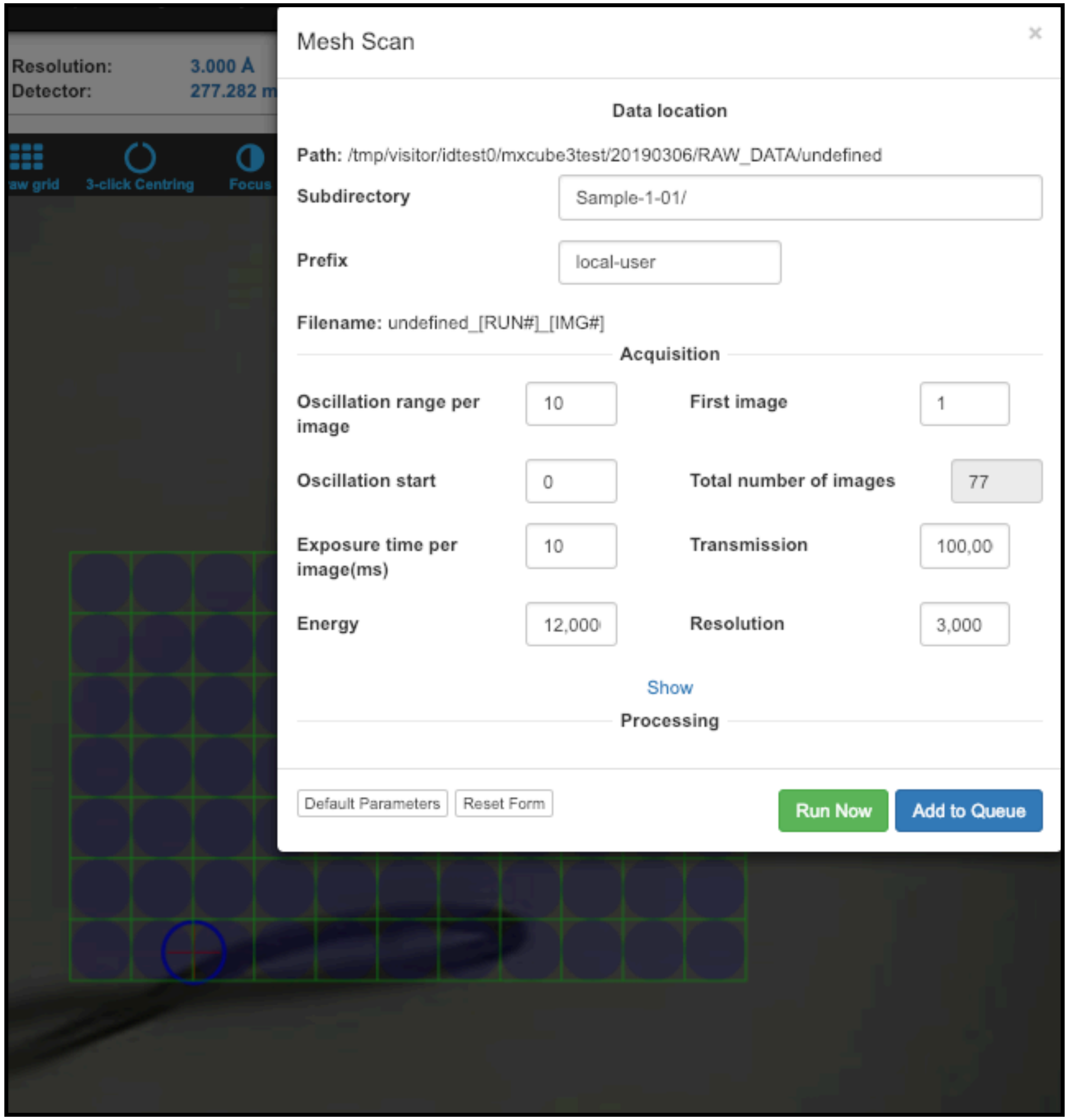
Element

×
H He
Li Be B C N O F Ne
Na Mg Al Si P S Cl Ar
K Ca Sc Ti V Cr Mn Fe Co Ni Cu Zn Ga Ge As Se Br Kr
Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cd In Sn Sb Te I Xe
Cs Ba Hf Ta W Re Os Ir Pt Au Hg Tl Pb Bi Po At Rn
Fr Ra Rf Db Sg Bh Hs Mt Ds Rg Cn Uut Fl Uup Lv Uus Uuo
La Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Lu
Ac Th Pa U Np Pu Am Cm Bk Cf Es Fm Md No Lr

Element

Edge

DataCollections



Mesh Scan

Data location

Path: /tmp/visitor/idtest0/mxcube3test/20190306/RAW_DATA/undefined

Subdirectory Sample-1-01/

Prefix local-user

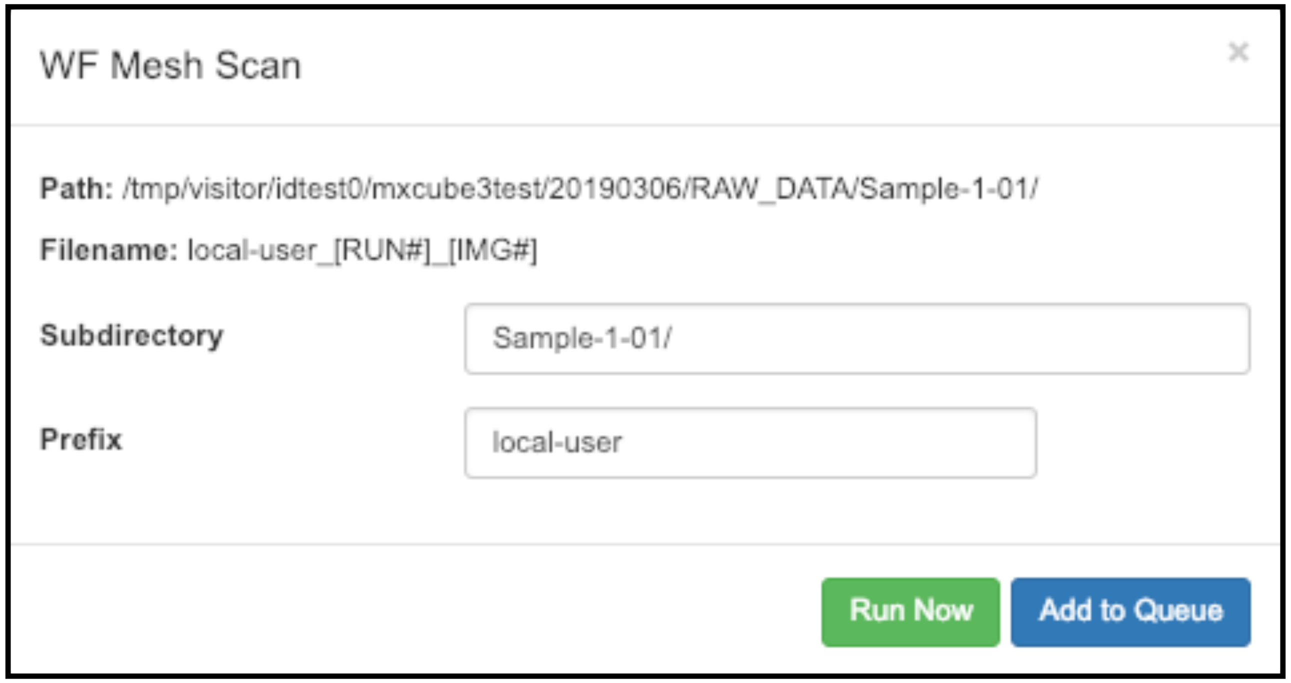
Filename: undefined_[RUN#]_[IMG#]

Acquisition

Oscillation range per image	10	First image	1
Oscillation start	0	Total number of images	77
Exposure time per image(ms)	10	Transmission	100,00
Energy	12,000	Resolution	3,000

Processing

DataCollections



WF Mesh Scan

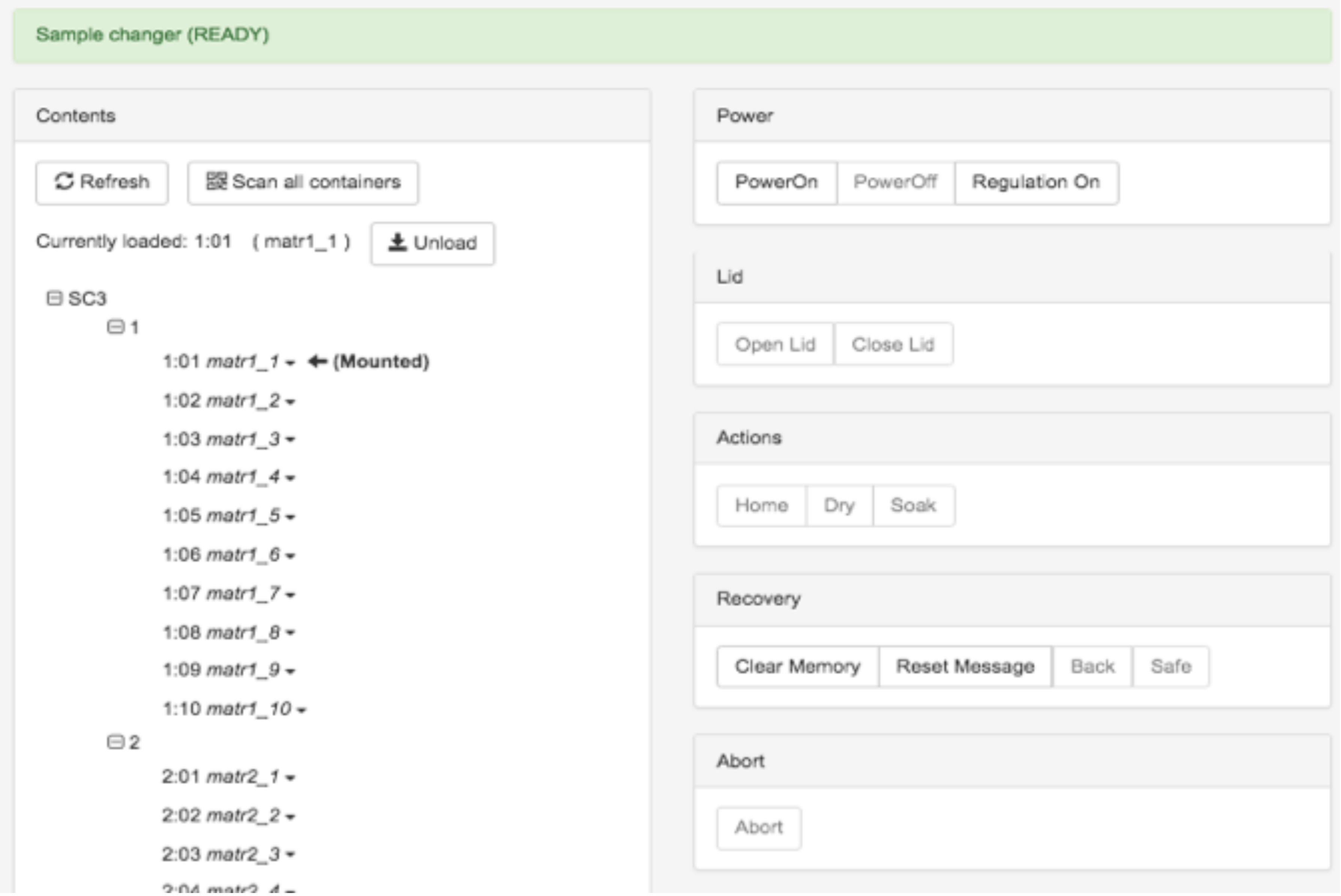
Path: /tmp/visitor/idtest0/mxcube3test/20190306/RAW_DATA/Sample-1-01/

Filename: local-user_[RUN#]_[IMG#]

Subdirectory Sample-1-01/

Prefix local-user

Sample Changer



Sample changer (READY)

Contents

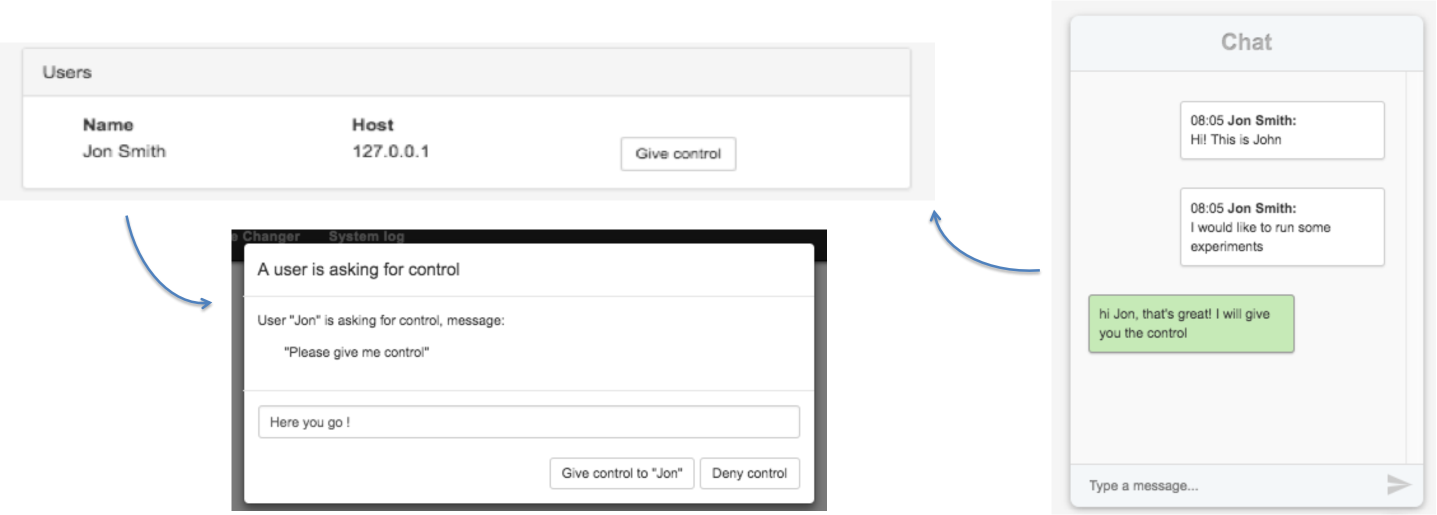
Currently loaded: 1:01 (matr1_1)

- SC3
  - 1
    - 1:01 *matr1_1* ← **(Mounted)**
    - 1:02 *matr1_2*
    - 1:03 *matr1_3*
    - 1:04 *matr1_4*
    - 1:05 *matr1_5*
    - 1:06 *matr1_6*
    - 1:07 *matr1_7*
    - 1:08 *matr1_8*
    - 1:09 *matr1_9*
    - 1:10 *matr1_10*
  - 2
    - 2:01 *matr2_1*
    - 2:02 *matr2_2*
    - 2:03 *matr2_3*
    - 2:04 *matr2_4*

Remote Operation

- Master/Slave mode
- Master is a local user/beamline staff
- Give/ask for control
  - slave cannot drive the beamline
- Screen mirroring
- In user operation at ESRF

- Demo



Users

Name	Host	
Jon Smith	127.0.0.1	Give control

A user is asking for control

User "Jon" is asking for control, message:

"Please give me control"

Here you go !

Chat

08:05 **Jon Smith:**
Hi! This is John

08:05 **Jon Smith:**
I would like to run some experiments

hi Jon, that's great! I will give you the control

LIMS integration

- Our LIMS is Ispyb
- The user configures the samples in Ispyb
- Mxcube retrieves the samples info (name, location in the SC, etc.)
- Data collection results are posted to Ispyb (beamline parameters, data collection info, file paths…)
- Auto triggering of data analysis (EDNA) performed by mxcube
    - feedback of data collection proposal (crystal characterisation) (diff plan)
- Results are displayed in the interface

Simulated beamline

- Extensive set on mockups equipment
  - Diffratometer
  - Detector
  - Motors/movables
  - Lims interface
  - …
- (almost) all the functionality of the interface can be tested without beamline (to certain degree)
- The xml files defines which components to use
- You can mix real and simulated equipment

MXCuBE 3 - demo

http://localhost:8090

MXCuBE3 People

Team:

MAX IV: M. Eguiraun, J. Nan, U. Muller, A. Gonzalez
ESRF: M. Oscarsson, A. Beteva, D. de Sanctis
Do not forget: M. Guijarro, F. Bolmsten, A. Milan-Otero, M. Thunissen, …

Supported by:

MXCuBE collaboration
MAX IV MX and KITS teams
ESRF BCU team

Publications:

MXCuBE 3 web application, on the way to next generation experiment control: NOBUGS16
Bringing MX experiments to the web MXCuBE 3: ICALEPCS17
MXCuBE 3 web application for MX experiment control; elease update and user experience: NOBUGS18

Thanks for your attention!

Exercises

Environment (docker)

1. **Getting mxcube**: (put it somewhere you like, do not use you existing mxcube3 folder)

- git clone https://github.com/meguiraun/mxcube3.git
- cd mxcube3
- git checkout -b v3.0.1 origin/v3.0.1
- cd mxcube3
- git clone https://github.com/meguiraun/HardwareRepository.git
- cd HardwareRepository
- git checkout -b 2.2 origin/2.2

2. **Running**: change the firt part of the -v to where you downloaded mxcube3 in the step above
3. docker pull mikeleguiraun/mxcube:mxcube3_workshop

- Terminal1: *docker run -v <YOUR_PATH_TO_MXCUBE>:/mxcube/mxcube3 -p 8081:8081 -p 8090:8090 --name mxcube3_workshop mikeleguiraun/mxcube:mxcube3_workshop*
- Terminal2: *docker exec -it mxcube3_workshop python mxcube3-server -w True -r test/HardwareObjectsMockup.xml*
- Terminal3:   *docker exec -it mxcube3_workshop npm install* -> this is only needed once, it takes a while
- Terminal3: *docker exec -it mxcube3_workshop npm start* (leave this running all the time)

```
8081: backend server
8090: web pack dev server (UI)
```

3. go to **localhost:8090**, username: idtest0, whatever password

For getting into the container: *docker exec -it mxcube3_workshop /bin/bash*

Environment (native)

1. **Getting mxcube**: (put it somewhere you like, do not use you existing mxcube3 folder)
   - git clone https://github.com/mxcube/mxcube3.git
   - git checkout -b v3.0.1 origin/v3.0.1
   - cd mxcube3
   - git clone https://github.com/mxcube/HardwareRepository.git
   - cd HardwareRepository
   - git checkout -b 2.2 origin/2.2

2. **Running**:
   1. Install conda: https://docs.conda.io/en/latest/miniconda.html
   2. conda create -n mxcube3 python=2.7
   3. conda activate mxcube3
   4. Install and run redis (check for your platform)
   5. pip install -r requirements.txt (from the requirements.txt file in mxcube3 main folder)
   6. python mxcube3-server -w True -r test/HardwareObjectsMockup.xml

   8. Install NODE from https://nodejs.org/en/
   9. In mxcube3 folder
      1. npm install
      2. npm start

Remember

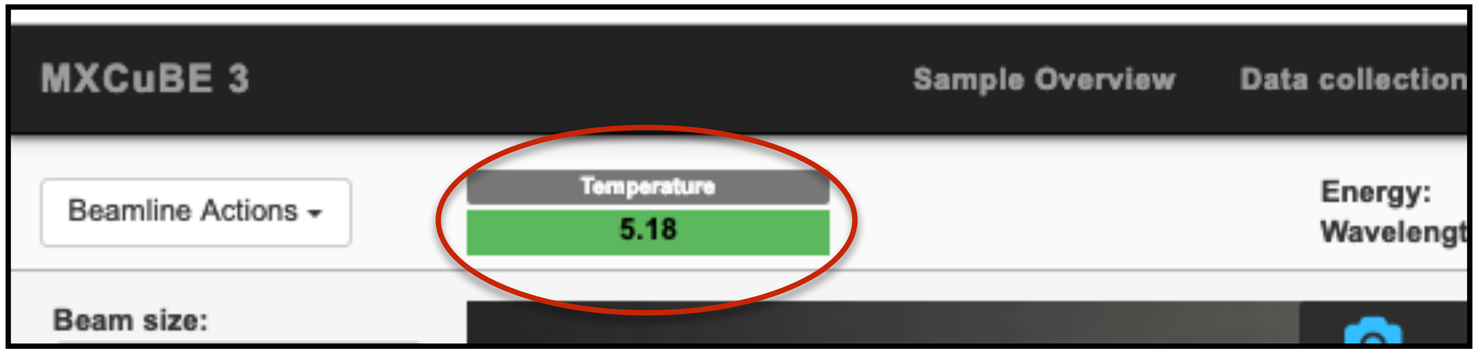
```
from HardwareRepository import HardwareRepository as hwr
hwr_dir = '../test/HardwareObjectsMockup.xml/'
hwr = hwr.HardwareRepository(hwr_dir)
hwr.connect()
ctrl = hwr.getHardwareObject(‘temp_controller’)
```

Exercise 1

A new hardware object in the interface

- On the hardware repository folder (repo):
  - git checkout -b temp_controller origin/temp_controller_template
  - New TemperatureController.py file
    - Finish it with random temperature value
    - New xml configuration file for it

- Mxcube 3 folder:
  - git checkout -b temp_controller origin/temp_controller_template
  - Several new files, check and finish them:
    - Load the previous hwobj (beamline-setup.xm)
    - First: API endpoint (new url for GET)
      - Test calling the url in a browser
    - React component: *mxcube3/ui/components/TemperatureController/TemperatureController.jsx*
    - Temperature actions and reducers
    - Forward temperature change events to the UI:
      - mxcube3/routes/signals.py
      - mxcube3/ui/serverIO.js

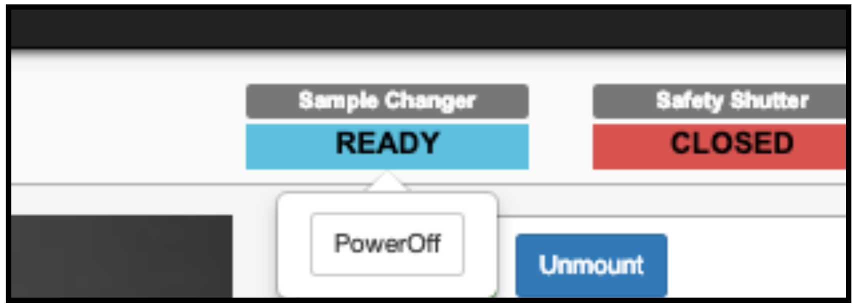


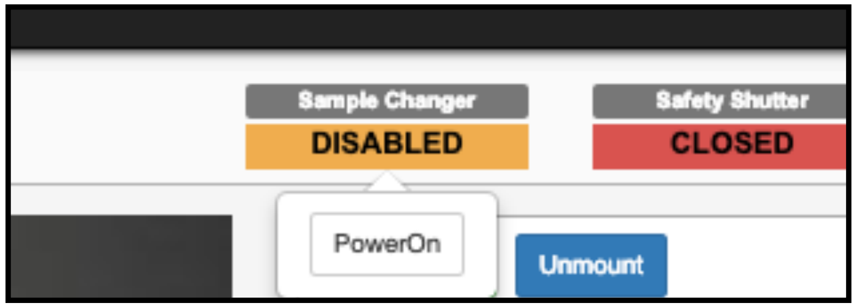
Exercise 2

Modify UI component

The current sample changer indicator in the data collection view only displays the state, change it so that you can power on and off the sample changer.

- On the mxcube3 folder
  - git checkout -b sample_changer_switch_template origin/sample_changer_switch_template
  - InOutSwitch2 does something very similar… just saying…
  - The sample changer maintenance hwobj already knows how to power it


Sample Changer
READY
Safety Shutter
CLOSED
PowerOff
Unmount


Sample Changer
DISABLED
Safety Shutter
CLOSED
PowerOn
Unmount