



| The European Synchrotron

# mxweb for developers



# Contents

Part 1 - Introduction and working in the project

Part 2 - mxcube core for developers

Part 3 - mxcube web for developers

# Introduction and Working in the project



Daniele De Sanctis



Antonia Beteva



Olof Svensson



Axel Bociarelli



Jean Baptise Florial (EMBL  
Gr)



Loic Huder

- Project started in 2003 at ESRF and became a collaboration in 2005
- Which makes this MXCuBEs 20th birthday :)
- Today we are 15 collaborating partners !



- The aim is to provide a platform for sharing solutions and know-how
- We are striving towards making MXCuBE and easy to deploy and use application (and extend)
- During my 10 years a very friendly and collaborative spirit with a solution oriented mindset.



- A very big thanks to all of you !





## You probably already know the story

- First based on Framework2, General ESRF UI Qt3 Framework
- Framework2+ mostly used for MXCuBE Qt3
- Framework2+ ported to Qt4 (much later Qt5)
- MXCuBE web project started
- HardwareRepository previously part of Framework2 becomes mxcubecore

*So, yes !*

*There are still some very old code in mxcubecore, you have or will probably notice :)*



## Organisation

- Steering committee
- Scientific committee
- Developers committee ( That's us :) )

- Like in most projects there are some conventions and good practices
- We try to base those on what's widely used in the software community (it makes tooling easier and minimizes cognitive overhead ;) )
- Document in the CONTRIBUTING.md file  
<https://github.com/mxcube/mxcubecore/blob/develop/CONTRIBUTING.md>

### Brief summary

- Bugs, use the GitHub issues. Check for duplicates and provide as much information as possible
- Docstring are written with google style doc-strings and we have been using sphinx for generating documentation (*not used since a long time but now back working again, thanks Fabien :)*)
- Code style is PEP8, we are using flake8 (still in progress)
- Black for formatting

### Recommendation

- Consider using a editor with good support for formatting, linting and testing
- vscode, pycharm, (emacs or vi if one still is adventurous)
- There are the .vscode setting .editorconfig files committed to help setting up the environment
- Use conda or similar tool to handle your virtual environment and to install development dependencies like flake8 and pytest

### Github actions based CI

- PyTest
- Test coverage report
- Lint (still needs some work)
- Tag and publish package to PyPi
- Build of documentation (*still work in progress*)
  - <https://mxcubeweb.readthedocs.io/en/latest/>
  - <https://mxcubecore.readthedocs.io/en/latest/>

# mxcube for developers

**We never really use mxcube core on its own but you can, for fun (or for testing)**

### **Checkout repository**

**<https://github.com/mxcube/mxcubecore.git>**

**Instructions: <https://github.com/mxcube/mxcubeweb>**

### **Setup your environment**

**Favourite editor, Pytest, flake8, black, pre-commit**

**(conda-environment-dev.yml contains the development libraries/tools needed)**

### **Run the tests**

**Simply running pytest**



**What is mxcube - A control system agnostic library for sharing common routines and integration of instrumentation**

**These objects, routines and instrumentation interface logic, is implemented as HardwareObjects**

**Let's have a look at some code:**

```
import os
from gevent import monkey
monkey.patch_all(thread=False)

import mxcube_core
from mxcube_core import HardwareRepository as HWR

ROOT_DIR = os.path.abspath(os.path.dirname(mxcube_core.__file__))

hwr_config_path = "%s%s%s" % (
    os.path.join(ROOT_DIR, "configuration/mockup"),
    ":",
    os.path.join(ROOT_DIR, "configuration/mockup/test"),
)

print("Configuration path:")
print(hwr_config_path)

HWR.init_hardware_repository(hwr_config_path)

def print_value(val):
    print(val)

HWR.beamline.resolution.connect("valueChanged", print_value)
HWR.beamline.resolution.get_value()
HWR.beamline.resolution.set_value(1.4)
```

- **Gevented**
- **HardwareRepository** is accessed as **HWR**

## Simple example - output

```
DEBUG:HWR.DiffractometerMockup.moving motor {motor} to position {position}
+-----+
| role          | Class                | file                                | Time (ms) | Comment                |
+-----+
| beamline      | Beamline             | beamline_config.yml                | 100       | Start loading contents:|
| machine_info  | MachineInfoMockup   | mach-info-mockup.xml               | 37        |                        |
| transmission   | TransmissionMockup   | transmission-mockup.xml            | 2         |                        |
| energy        | EnergyMockup        | energy-mockup.xml                  | 1         |                        |
| beam          | BeamMockup          | beam-mockup.xml                    | 4         |                        |
| flux          | FluxMockup          | flux-mockup.xml                    | 268       |                        |
| detector      | DetectorMockup      | detector-mockup.xml                | 5         |                        |
| resolution    | ResolutionMockup    | resolution-mockup.xml               | 1         |                        |
| safety_shutter | ShutterMockup      | safety-shutter-mockup.xml          | 2         |                        |
| sample_changer | SampleChangerMockup | sample-changer-mockup.xml          | 9         |                        |
| diffractometer | DiffractometerMockup | diffractometer-mockup.xml          | 738       |                        |
| sample_view   | SampleView          | sample-view-mockup.xml             | 2         |                        |
| mock_procedure | ProcedureMockup     | procedure-mockup.yml                | 103       |                        |
| beamline      | Beamline            | beamline_config.yml                | 1278      | Done loading contents  |
+-----+
```

The beamline object can now be accessed via `HWR.beamline` (as seen previously)

But how does this work (in a nutshell :) ) ?

```
DEBUG:HWK.DIFFRACTOMETERMOCKUP.MOVING MOTOR (MOTOR) TO POSITION (POSITION)
+-----+
| role          | Class                | file                          | Time (ms) | Comment                |
+-----+
| beamline     | Beamline             | beamline_config.yml          | 100       | Start loading contents:|
| machine_info | MachineInfoMockup   | mach-info-mockup.xml        | 37        |                        |
| transmission | TransmissionMockup   | transmission-mockup.xml     | 2         |                        |
| energy       | EnergyMockup        | energy-mockup.xml           | 1         |                        |
| beam         | BeamMockup          | beam-mockup.xml             | 4         |                        |
| flux         | FluxMockup          | flux-mockup.xml             | 268       |                        |
| detector     | DetectorMockup      | detector-mockup.xml         | 5         |                        |
| resolution   | ResolutionMockup    | resolution-mockup.xml       | 1         |                        |
| safety_shutter | ShutterMockup      | safety-shutter-mockup.xml   | 2         |                        |
| sample_changer | SampleChangerMockup | sample-changer-mockup.xml   | 9         |                        |
| diffractometer | DiffractometerMockup | diffractometer-mockup.xml   | 738       |                        |
| sample_view  | SampleView          | sample-view-mockup.xml     | 2         |                        |
| mock_procedure | ProcedureMockup     | procedure-mockup.yml        | 103       |                        |
| beamline     | Beamline            | beamline_config.yml        | 1278      | Done loading contents  |
+-----+
```

- The configuration files are parsed and the hardware objects loaded
- Starting with the “beamline object” and traversing the “hierarchy” downwards loading the children
  - yaml files loaded via `HardwareRepository.load_from_yaml`
  - xml files loaded via `HardwareRepositoryClient._load_hardware_object`
- And yes, our plan is to (eventually) replace XML with YAML !

# HardwareObjects in action

```
└─ mxcubecore
  └─ __pycache__
  └─ .pytest_cache
  └─ .vscode
  └─ Command
  └─ configuration
  └─ HardwareObjects
    └─ __pycache__
    └─ abstract
    └─ ALBA
    └─ datamodel
    └─ DESY
    └─ EMBL
    └─ ESRF
    └─ Gphl
    └─ LNLS
    └─ MAXIV
    └─ mockup
    └─ Native
    └─ queue_entry
    └─ SOLEIL
    └─ __init__.py
    └─ Attenuators.py
    └─ autoprocessing.py
    └─ BeamInfo.py
    └─ Beamline.py
```

```
HardwareObjectsMockup.xml
└─ gphl
└─ mxcube-web
  └─ aperture.xml
  └─ beam_info.xml
  └─ beam.xml
  └─ beamcmds.xml
  └─ beamline_actions.xml
  └─ beamline_config.yml
  └─ beamstop_alignment_x.xml
  └─ beamstop_alignment_y.xml
  └─ beamstop_alignment_z.xml
  └─ beamstop.xml
  └─ capillary.xml
  └─ characterisation.xml
  └─ chip_definition.json
  └─ cryo.xml
  └─ data_analysis.xml
  └─ data_publisher.xml
  └─ detector.xml
  └─ diffractometer_beamline_action.xml
  └─ dtox.xml
  └─ edna_defaults.xml
  └─ ednaparams.xml
```

```
You, 5 months ago | 2 authors (Mikel Eguirraun and others)
1 <object class="MotorMockup">
2   <username>Omega</username>
3   <exporter_address>130.235.94.124:9001</exporter_address>
4   <motor_name>Omega</motor_name>
5   <unit>1e-3</unit>
6   <GUIstep>90</GUIstep>
7 </object>
8
```

- Each site has its own directory for site specific HardwareObjects and code
- As you now know there is a folder with yaml and xml file configuring these objects

# HardwareObjects in action

```
detectorMockup.py  detector.xml  DetectorMockup.py
```

```
oscarsson > projects > mxcube-web > test > HardwareObjectsMockup.xml > detector.xml
You, 11 seconds ago | 3 authors (Marcus Oskarsson and others)
<object class = "DetectorMockup">
  <username>detector</username>
  <object hwrId="/dtox" role="detector_distance"/>
  <tempThreshold>33.5</tempThreshold>
  <humidityThreshold>20.0</humidityThreshold>
  <tolerance>0.2</tolerance>
  <type>Eiger2</type>
  <model>9M</model>
  <manufacturer>DECTRIS</manufacturer>
  <px>0.075</px>
  <py>0.075</py>
  <width>3110</width>
  <height>3269</height>
  <hasShutterless>True</hasShutterless>
  <fileSuffix>cbf</fileSuffix>
  <roi_mode_list>["0", "C2", "C16"]</roi_mode_list>
  <beam>
    <!-- Values are in pixels/mm and pixels resp. -->
    <ax>0.0</ax>
    <ay>0.0402</ay>
    <bx>1565.715</bx>
    <by>1702.058</by>
  </beam>
  <exports>["restart"]</exports>
</object>
```

```
class DetectorMockup(AbstractDetector):
    """
    Descript. : Detector class. Contains all information about detector
               the states are 'OK', and 'BAD'
               the status is busy, exposing, ready, etc.
               the physical property is RH for pilatus, P for rayonix
    """

    def __init__(self, name):
        """
        Descript. :
        """
        AbstractDetector.__init__(self, name)

    def init(self):
        """
        Descript. :
        """
        AbstractDetector.init(self)

        self.temperature = 25
        self.humidity = 60
        self.actual_frame_rate = 50
        self.roi_modes_list = ast.literal_eval(
            self.get_property("roi_mode_list", '["0", "C2", "C16"]')
        )
        self.roi_mode = 0
        self.exposure_time_limits = eval(
            self.get_property("exposure_time_limits", "[0.04, 60000]")
        )
```

- Important: **class name and file name should be the same**
- XML is **not validated** against any schema, we often use **ast.literal\_eval** to evaluate python structures (There are some ideas on how to provide a stricter definition of the configuration for each object)
- You may see **Equipment and Device** instead of **Object** as well (those are **Equipment and Device** are deprecated and should be replaced by **Object** !)

# HardwareObjects in action

```
detectorMockup.py  detector.xml x
he > oscarsson > projects > mxcube-web > test > HardwareObjectsMockup.xml > detector.xml
You, 11 seconds ago | 3 authors (Marcus Oskarsson and others)
<object class = "DetectorMockup">
  <username>detector</username>
  <object hwrid="/dtox" role="detector_distance"/>
  <tempThreshold>33.5</tempThreshold>
  <humidityThreshold>20.0</humidityThreshold>
  <tolerance>0.2</tolerance>
  <type>Eiger2</type>
  <model>9M</model>
  <manufacturer>DECTRIS</manufacturer>
  <px>0.075</px>
  <py>0.075</py>
  <width>3110</width>
  <height>3269</height>
  <hasShutterless>True</hasShutterless>
  <fileSuffix>cbf</fileSuffix>
  <roi_mode_list>["0", "C2", "C16"]</roi_mode_list>
  <beam>
    <!-- Values are in pixels/mm and pixels resp. -->
    <ax>0.0</ax>
    <ay>0.0402</ay>
    <bx>1565.715</bx>
    <by>1702.058</by>
  </beam>
  <exports>["restart"]</exports>
</object>

HardwareObjects > mockup > DetectorMockup.py > DetectorMockup > init
class DetectorMockup(AbstractDetector):
    """
    Descript. : Detector class. Contains all information about detector
    the states are 'OK', and 'BAD'
    the status is busy, exposing, ready, etc.
    the physical property is RH for pilatus, P for rayonix
    """
    def __init__(self, name):
        """
        Descript. :
        """
        AbstractDetector.__init__(self, name)
    def init(self):
        """
        Descript. :
        """
        AbstractDetector.init(self)
        self.temperature = 25
        self.humidity = 60
        self.actual_frame_rate = 50
        self.roi_modes_list = ast.literal_eval(
            self.get_property("roi_mode_list", ["0", "C2", "C16"])
        )
        self.roi_mode = 0
        self.exposure_time_limits = eval(
            self.get_property("exposure_time_limits", "[0.04, 60000]")
        )
    )
```

- values in `<start_tag></end_tag>` are retrieved with `get_property("tag")`
- objects with a role are retrieved by `get_object_by_role("role")`
- *Future development: (might get replaced by simply adding those roles/attributes directly when object is parsed)*

HardwareObjects can communicate with control systems via something called channels and commands

```
class Microdiff(MiniDiff.MiniDiff):
    def init(self):
        self.phiMotor = self.get_object_by_role("phi")
        self.exporter_addr = self.get_property("exporter_address")

        self.update_state([HardwareObjectState.READY])

        self.x_calib = self.add_channel(
            {
                "type": "exporter",
                "exporter_address": self.exporter_addr,
                "name": "x_calib",
            },
            "CoaxCamScaleX",
        )
        self.y_calib = self.add_channel(
            {
                "type": "exporter",
                "exporter_address": self.exporter_addr,
                "name": "y_calib",
            },
            "CoaxCamScaleY",
        )
```

Channels and commands provide an abstraction for the various control systems used



Channels and commands provide an abstraction for the various control systems used

Supported are: Exporter (EMBL), Tango, Taco, Tine, Sardana, EPICS, (SPEC), \*BLISS (Not via channels and commands)

- Channels for values (and events)
- Commands for “functions”

HardwareObjects inherit **CommandContainer** meaning that we can use **add\_channel** and **add\_command**

- The library **pyDispatcher** is used for handling signals/events
- Each protocol implements a **CommandObject**, a **ChannelObject** and protocol specific “Client” for handling events and other protocol internals.
- A channel emits a **“update”** signal with the new value on a event

Antonia will talk more about this tomorrow

- The mxcube work have harmonised the API we use for these **HardwareObjects**: (summary of changes: <https://github.com/mxcube/mxcubecore/blob/develop/changelog.txt>)
- *There are still some work to be done for instance with signals and further refining the interface of certain objects and converting to YAML*
- A base class for all Hardware Objects **HardwareObject** and has a well set of state; READY, BUSY, OFF, FAULT etc.
- A set of commonly used subclasses derived from HardwareObject;
- Introduction of **BeamlineObject** for facilitating access to “well known” HardwareObjects
- Yaml configuration

- The basic objects we use are, `HardwareObject`, `AbstractActuator`,
- From these there are a number of objects derived that everybody most likely will use or can reuse: `AbstractNState`, `AbstractMotor`, `AbstractTransmission`, `AbstractResolution`, `AbstractDetector`, `AbstractShutter`, `AbstractSlits`, `AbstractBeam` and ... actually the list is quite long :)
- To keep in mind:
  - `HardwareObject` provides `set_state` and `get_state`
  - `AbstractActuator` provides `set_value`, `get_value` and `set_limits`, and `get_limits`
  - `AbstractNState` is a `AbstractActuator` where the value is in finite set of values (states)
- Thanks to Antoina:  
<https://github.com/mxcube/mxcubecore/blob/develop/Hierarchy.pdf>

# About the queue

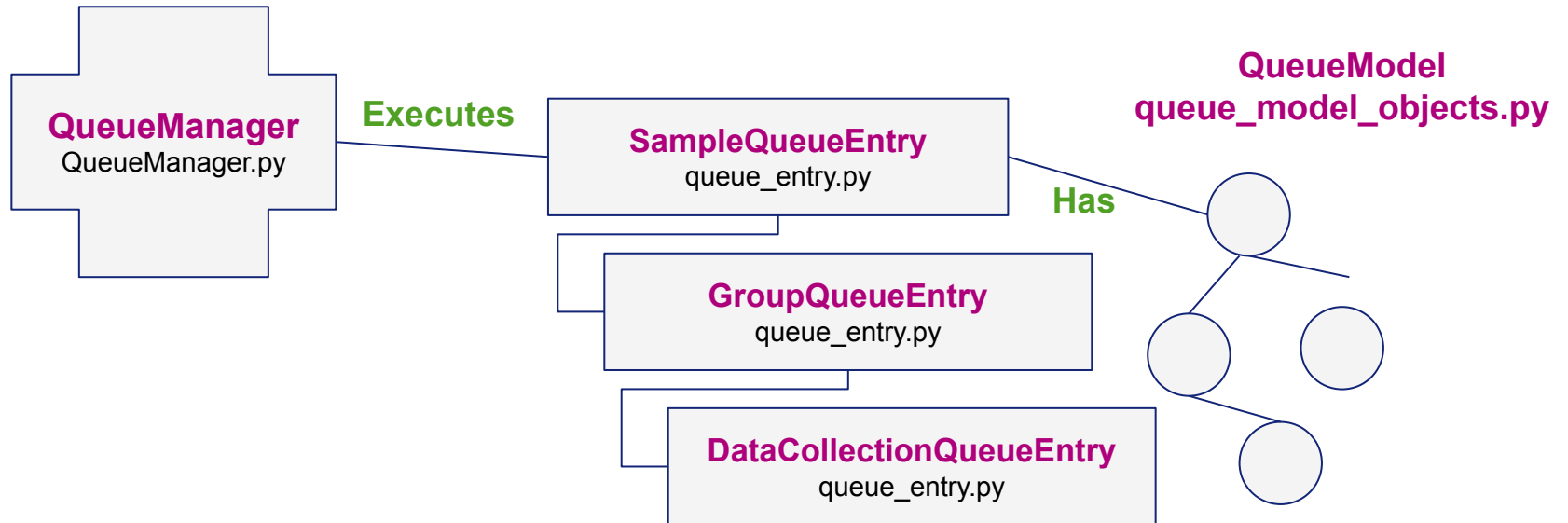
(Ok, take a deep breath ;) )

**HardwareObjects** are important for instrument control, but **WE** want to collect data !

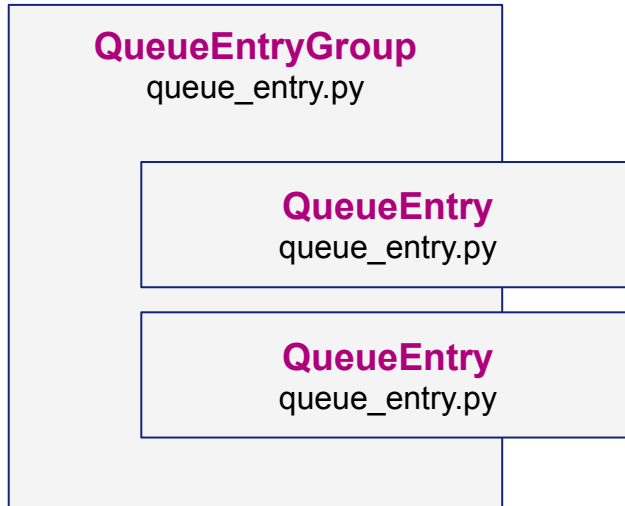
Data acquisition tasks/protocols are implemented as **QueueEntry** objects

These **QueueEntryObjects** are executed via a queue (**QueueManager**)

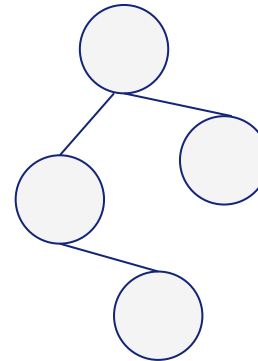
The queue further provides means for automation



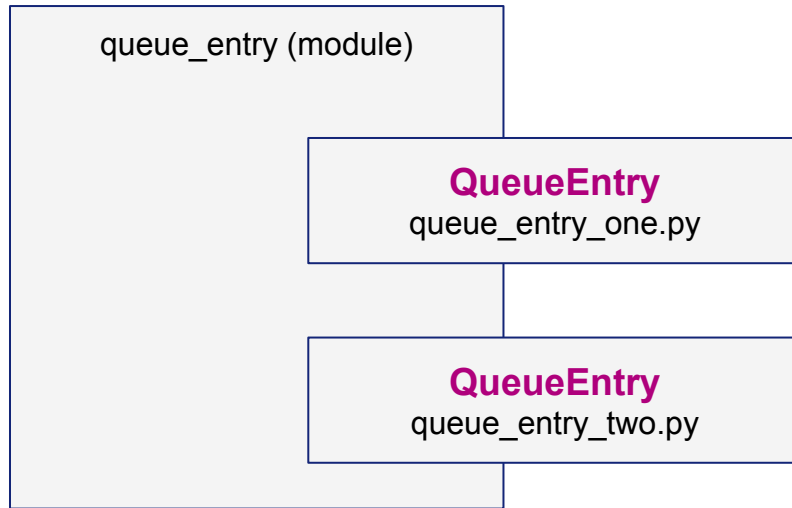
- **QueueManager** executes **QueueEntry** objects each having data models.
- Each **QueueEntry** in turn is a tree with Parent and child nodes, where the tree is traversed downwards (depth first), children gets executed after the parent
- Each entry has a **pre and post execute** that gets executed **before and main execute function**
- In **practice maximum three levels of QueueEntry** objects are used, what's called **SampleQueueEntry, GroupQueueEntry** and **QueueEntry**. (Workflows sometimes uses more) (*The use of Group might be removed/changed*)



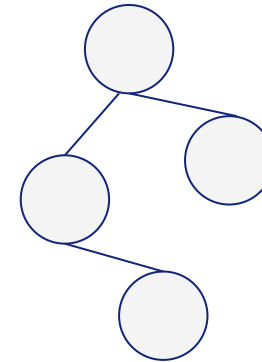
**QueueModel**  
`queue_model_objects.py`



- QueueModels are written as python objects that gets converted to dicts when passed around, hard to debug and know what to pass.
- Creation of each type of Queue entry is more or less manually wired for creation in the UI



## CommonQueueModels



- Each QueueEntry is self **contained in its own file with its specific models**
- Models are expressed as **Pydantic models**
- All common models are **shared in common queue models module**
- Each QueueEntry object in the queue\_entry module (or other search path) is **dynamically imported**
- **A generic UI interface/dialog** can be built based on the **model data associated with a QueueEntry** (an evolution of the current workflow dialog)

## Example - New style queue entry

```
EXPLORER
...
ssx_chip_collection.py 9+, M x
PyISPyBClient.py U
events_api.py 9+, U
test_collection.py 9+, U
ssx_chip_collection.py 9+, M x

MXCUBECORE
> __pycache__
__init__.py
advanced_connector.py
base_queue_entry.py
characterisation.py
data_collection.py
energy_scan.py
generic_workflow.py
import_helper.py
optical_centring.py
sample_centring.py
ssx_chip_collection.py 9+, M
test_collection.py 9+, U
xray_centering.py
xray_centering2.py
xrf_spectrum.py
SOLEIL
__init__.py
Attenuators.py
autoprocessing.py
BeamInfo.py
Beamline.py
BeamlineActions.py
BeamlineTools.py
Bliss.py
BlissActuator.py
BlissEnergy.py
BlissHutchTrigger.py
BlissMotor.py
BlissMotorWPositions.py
BlissNState.py
BlissRontecMCA.py
BlissShutter.py
Camera.py
Cats90.py
CatsBessy.py
CatsMaint.py
OUTLINE
TIMELINE

HardwareObjects > queue_entry > ssx_chip_collection.py > SsxChipCollectionQueueEntry > execute
1 import os
2
3 from pydantic import BaseModel, Field
4 from devtools import debug
5
6 from mxcubeore import HardwareRepository as HWR
7
8 from mxcubeore.HardwareObjects.queue_entry.base_queue_entry import (
9     BaseQueueEntry,
10 )
11
12 from mxcubeore.HardwareObjects.queue_model.objects import (
13     TaskNode,
14 )
15
16
17 _credits_ = ["MXCuBE collaboration"]
18 _license_ = "LGPLv3+"
19 _category_ = "General"
20
21
22 class SSXCollectionParameters(BaseModel):
23     first_image: int
24     kappa: float
25     kappa_phi: float
26     # numRows: int
27     # numCols: int
28     beam_size: float
29     shutterless: bool
30     selection: list = Field([])
31
32     class Config:
33         extra: "ignore"
34
35 class SSXUserCollectionParameters(BaseModel):
36     sub_sampling: float = Field(2, gt=0, lt=100)
37     take_pedestal: bool = Field(True)
38
39     class Config:
40         extra: "ignore"
41
42
43 class CommonCollectionParameters(BaseModel):
44     skip_existing_images: bool
45     take_snapshots: int

HardwareObjects > queue_entry > ssx_chip_collection.py > ...
75     extra: "ignore"
76
77 class SsxChipCollectionTaskParameters(BaseModel):
78     path_parameters: PathParameters
79     common_parameters: CommonCollectionParameters
80     collection_parameters: SSXCollectionParameters
81     user_collection_parameters: SSXUserCollectionParameters
82     legacy_parameters: LegacyParameters
83
84
85 class SsxChipCollectionQueueEntry(BaseQueueEntry):
86     """
87     Defines the behaviour of a data collection.
88     """
89     DATA_MODEL = SsxChipCollectionTaskParameters
90     NAME = "SSXChipCollection"
91     REQUIRES = ["point", "line", "no_shape", "chip", "mesh"]
92
93     # New style queue entry does not take view argument,
94     # adding kwargs for compatability, but they are unused
95     def __init__(self, data: SsxChipCollectionTaskParameters, view=None,
96                 super().__init__(view=view, data_model=TaskNode(data)))
97
98
99     def execute(self):
100         super().execute()
101
102         debug(self.data_model.task_data)
103
104         selected_regions = self.data_model.task_data.collection_parameters.selected_regions
105         selected_regions = selected_regions if selected_regions else []
106
107         # HWR.beamline.processing.start_something(args)
108         for region in selected_regions:
109             data_root_path = os.path.join(
110                 HWR.beamline.session.get_base_image_directory(),
111                 self.data_model.task_data.path_parameters.subdir
112             )
113
114             process_path = os.path.join(
115                 HWR.beamline.session.get_base_process_directory(),
116                 self.data_model.task_data.path_parameters.subdir
117             )
118
119             fname_prefix = self.data_model.task_data.path_parameters
```



# Example UI

The screenshot displays the MXCuBE-Web (SSX-CHIP) interface. At the top, navigation tabs include 'Samples', 'Data collection', 'Equipment', and 'System log'. The main header shows system parameters: Energy: 12.4000 KeV, Resolution: - Å, Transmission: 10.0 %, Wavelength: 1.00 Å, Detector: 10.1 mm, and Flux: 1.50e+10 ph/s. On the right, status indicators for Detector (UNKNOWN), Sample Changer (READY), Capillary (UNKNOWN), Fast Shutter (CLOSED), and Safety shutter (CLOSED) are visible. A central dialog box is open, showing the following configuration:

- Path:** /data/id29/inhouse/upid291/20220916/RAW\_DATA/Mb/Mb-Mb/
- Filename:** Mb-Mb\_[RUN#]\_[IMG#]
- Subdirectory:** Mb/Mb-Mb/
- Prefix:** Mb-Mb
- Acquisition:**
  - Exp Time:** 0.02
  - Sub Sampling:** 4
  - Align Chip:**
  - Take Pedestal:**

Buttons at the bottom of the dialog include 'Default Parameters', 'Run Now', and 'Add to Queue'. The background interface shows a 'Beamline Actions' dropdown, 'Beam size' (50), 'Chip (Diamond Chip)' with a 'Navigate' button, 'Omega' (359.90), 'X' (0.000), and 'Y' (-49.900) coordinates, and a 'Sample alignment' section with directional controls and a 'Show motors' option. A '50 μm' scale bar is present in the bottom left. On the right, a 'Sample: Mb - Mb' section shows a 'Queued Samples (0)' list with entries like 'SSXFoilCollection' and 'SSXInjectorCollection'. A 'Log messages' panel at the bottom right shows recent events: 'Opening OH2 safety shutter', 'Not mounting next sample automatically (Auto mount next)', and 'Acquired 1000 images'.

# Using mxcupeweb

MxCuBE3 (mx2112 collecting)

Samples Data collection SC tools System log

Help Remote Sign out

Energy: 14.0000 keV Resolution: 5.137 Å Transmission: 13.11 % Cryo: 100 K  
Wavelength: 0.8856 Å Detector: 872.000 mm Flux: 5.03E+11 ph/s

Detector: READY Sample Changer: READY Safety Shutter: OPEN Fast Shutter: CLOSED Capillary: IN Beamstop: IN Ring Current: 193.0 mA

Beamline Actions

Beam size: 20  
Omega: 90.00  
Kappa: 0.00  
Phi: 0.00

Sample alignment:

Grid-2

50 µm

Run Queue Unmount Settings

Sample: Sample-8:1:01 Queued Samples (0)

- Mesh Scan
- : Data Collection
- Grid-2: Mesh Scan
- Grid-2: Data Collection
- Grid-3: Mesh Scan
- Grid-3: Data Collection

Log messages:

- [14:50:21]: Not mounting next sample automatically (Auto mount next)
- [14:50:19]: Workflow: Workflow finished successfully.
- [14:50:17]: Workflow: Sample position after move: sampx=-0.546 sampy=0.468 phiy=0.189 phiz=-0.023

# Working in the project

- Same conventions and guidelines as mxcube-core
- CI Pipeline is more or less the same as mxcube-core
- Additionally running Cypress end to end (e2e) tests
- Something called ESLint for javascript
- Build of documentation (*still work in progress*)
  - <https://mxcubeweb.readthedocs.io/en/latest/>

Instructions on: <https://github.com/mxcube/mxcubeweb>

```
(mxcubewebtest) oscarsson@laposcarsson:~/tmp/mxcube$ mxcubeweb-server -r /tmp/mxcube/mxcubeweb/test/HardwareObjectsMockup.xml/ --static-folder /tmp/mxcube/mxcubeweb/ui/build/ -L debug
```

I

# Installing and configuring

Instructions on: <https://github.com/mxcube/mxcubeweb>

Designed for MX experiments - with the idea of same interface on all sites

To a certain extent configurable **instrumentation control**, **procedures / methods**

The screenshot displays the MXCuBE-Web (osc) interface. At the top, there are navigation tabs for Samples, Data collection, Equipment (highlighted with a blue box), and System log. Below these are status indicators for Camera 1, Camera 2, Detector (READY), Sample Changer (READY), Fast Shutter (CLOSED), Safety shutter (CLOSED), and Ring Current (200.0 mA). The main control area includes a Beamline Actions dropdown (highlighted with a blue box) and a central image of a sample. A context menu is open over the image, with the 'SSX Chip Collection (Lima1)' option highlighted (blue box). The left sidebar contains Phase Control (Centring), Beam size (10), Omega (311.10), Kappa (11.0), Kappa Phi (22.0), and Sample alignment controls. The right sidebar shows a Run Queue (Unmount) and a Settings dropdown. A log messages section at the bottom right shows a message: [12:29:08]: Mounting sample: test.

MXCuBE-Web (osc) | Samples | Data collection | Equipment | System log | Help | Remote | Sign out (ldtest0)

Beamline Actions ▾

Energy: 12.4000 keV | Resolution: 0.740 Å | Transmission: 10.0 % | Cryo: 200.00 k

Wavelength: 1.00 Å | Detector: 10.1 mm | Flux: 2.30e+12 ph/s

Camera 1: [ON] | Camera 2: [ON] | Detector: READY | Sample Changer: READY | Fast Shutter: CLOSED | Safety shutter: CLOSED | Ring Current: 200.0 mA

The display of available instrumentation is configurable in `ui.yaml`

To the left motor control and on the top “beamline setup”

MXCuBE-Web (os)

Beamline Actions ▾

Phase Control: Centring

Beam size: 10

Omega: 311.10 | 90.0°

Kappa: 11.0 | 0.1°

Kappa Phi: 22.0 | 0.1°

Sample alignment: [↑] [⚙️] [↓]

Show motors ▾

```
sample_view:
  id: sample_view
  components:
    -
      label: Omega
      attribute: diffractometer.phi
      role: omega
      step: 90
      precision: 2
      suffix: °
    -
      label: Kappa
      attribute: diffractometer.kappa
      role: kappa
      step: 0.1
      precision: 1
      suffix: "°"
```

```
beamline_setup:
  id: beamline_setup
  components:
    -
      label: Beamstop
      attribute: beamstop
    -
      label: Capillary
      attribute: capillary
    -
      label: Fast Shutter
      attribute: fast_shutter
    -
      label: Safety shutter
      attribute: safety_shutter
    -
      label: Detector
      attribute: detector
    -
      label: Energy
      attribute: energy
      step: 0.001
      precision: 4
      suffix: keV
```

Methods and procedures can be added in three ways:

- **Equipment view - For not so often used or temporary instrumentation commands**
- **Beamline action - For procedures that are frequently used and involves more than a simple command**
- **Queue entry / task - For collecting data**

## Equipment view - For less often or temporary instrumentation commands

MXCuBE-Web (osc)    Samples    Data collection    Equipment    System log    Help    Remote    Sign out (dtest0)

Mockup **READY**

Content

Refresh    Scan all containers

+ Mockup

Power

PowerOn    PowerOff    Regulation On

Lid

Open Lid    Close Lid

Actions

Home    Dry    Soak

Recovery

Clear Memory    Reset Message    Back    Safe

Abort

Abort

my\_fancy\_function

my\_fancy\_function    Last result

Speed\*

Num Images\*

Exp Time\*

PhaseEnum\*  
An enumeration.

Run my\_fancy\_function

detector **READY**

diffractometer **READY**

abort    my\_fancy\_function

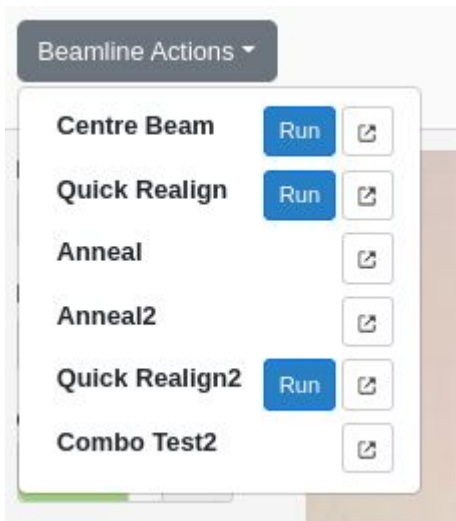
**Methods are automatically added if they are “exported” with the export tag and the method is type hinted (at least with a return type)**

```
<exports>["abort", "status", "my_fancy_function", "my_other_funny_function"]</exports>
```



**Beamline action - For procedures that are frequently used and involves more than a simple command**

**Configured as the beamline\_actions of the Beamline hardware object**



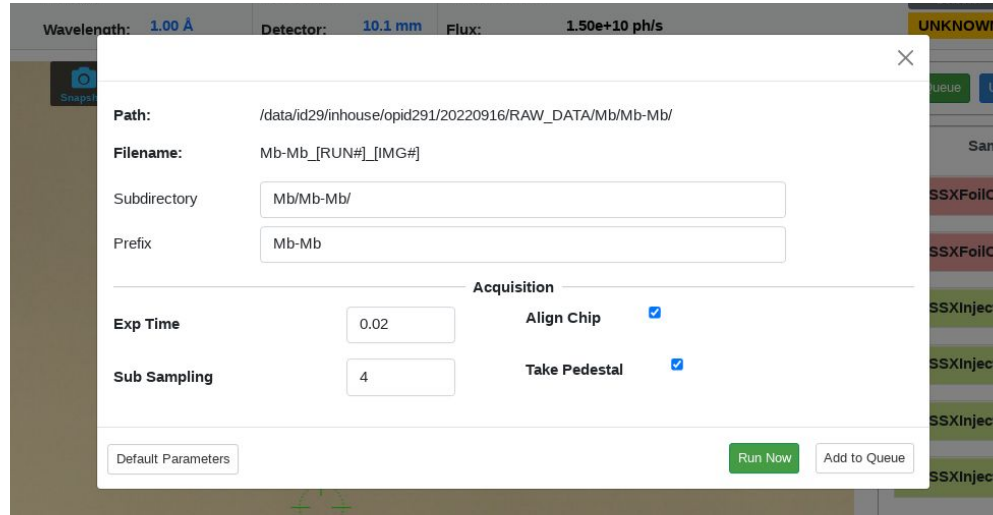
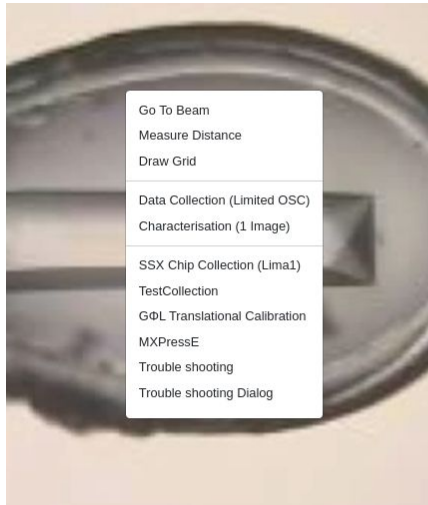
```
<object class="BeamlineActionsMockup">
  <commands>[
    {"type": "controller", "name": "Centre Beam", "command": "HardwareObjects.m
    {"type": "controller", "name": "Quick Realign", "command": "HardwareObjects
    {"type": "controller", "name": "Anneal", "command": "HardwareObjects.mockup
    {"type": "annotated", "command": "HardwareObjects.mockup.BeamlineActionsMoc
    {"type": "annotated", "command": "HardwareObjects.mockup.BeamlineActionsMoc
    {"type": "annotated", "command": "HardwareObjects.mockup.BeamlineActionsMoc
  ]
</commands>
</object>
```

The controller commands define the command arguments programmatically via the [CommandObject.add\\_argument](#) method

The annotated command uses the method typehints to define the arguments (*still work in progress, sorry the UI display is currently broken, we are working on it*)

**Mikel will make a practical about Beamline actions**

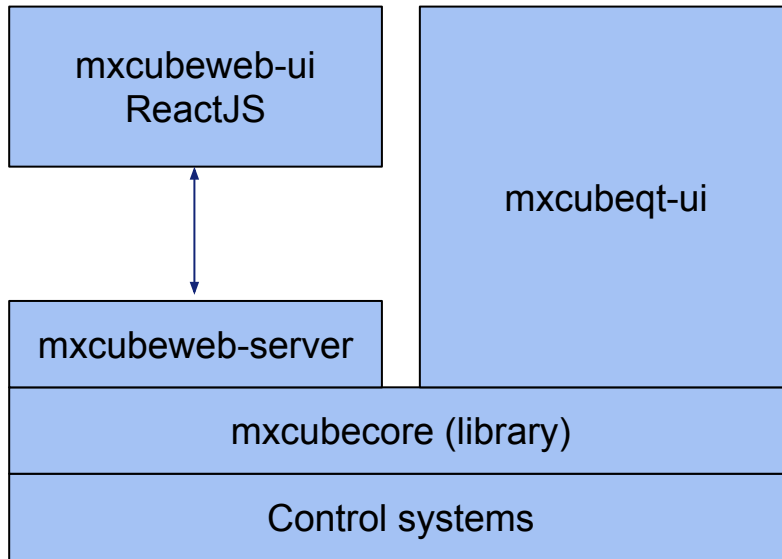
## Queue entry / task - For collecting data



Write a task that takes a Pydantic model and add it to available\_methods of Beamline object

```
82     legacy_parameters: LegacyParameters
83
84
85     class SsxChipCollectionQueueEntry(BaseQueueEntry):
86         """
87         Defines the behaviour of a data collection.
88         """
89         DATA_MODEL = SsxChipCollectionTaskParameters
90         NAME = "SSXChipCollection"
91         REQUIRES = ["point", "line", "no_shape", "chip", "mesh"]
92
93         # New style queue entry does not take view argument,
94         # adding kwargs for compatability, but they are unused
95         def __init__(self, data: SsxChipCollectionTaskParameters, view=None):
96             super().__init__(view=view, data_model=TaskNode(data))
97
98
99         def execute(self):
100             super().execute()
101
```

```
available_methods:
    datacollection: True
    characterisation: True
    helical: True
    xrf_spectrum: True
    energy_scan: True
    mesh: True
    ssx_chip_collection: True
    gphlworkflow: True
    test_collection: True
```



Frontend built using Javascript, React and bootstrap, redux ...)

Asynchronous communication done over websockets

## Some of the bigger libraries we are using:



Flask



Flask-Security



SpecTree

- Flask
- Flask security for handling user and session
- Flask-SocketIO for web sockets
- SpecTree for OpenAPI documentation, <http://localhost:8081/apidoc/swagger/>

Something called an adapter object converts (adapts) a HardwareObjects to the web world, get and set over GET and POST requests and events over websockets.

```
✓ mxcube3
  > __pycache__
  > .pytest_cache
  > .vscode
  ✓ core
    > __pycache__
    > adapter
    > components
    > models
    > util
    + __init__.py
  > routes
```

Adapters found in the adapter directory, adapts to standardised API of HardwareObjects.

Components are larger pieces of functionality, such as queue, lms, workflow with static mapping to routes. (Some might become adapters in the future)

Models for defining complex data structures and marshaling

Routes contains explicitly defined routes

### After MXCuBE Web 4

- **Update/Modernize Javascript code**
- **Using typescript ?**
- **Exchange Speectree for FlaskOpenAPI3 or other ?**
- **Possibly remove dependency on gevent**
- **Your ideas**