User Documentation

Release 1.0

European XFEL

Jan 27, 2023

CONTENTS

The data analysis group at European XFEL can help you process and analyse your data during and after your experiment. Please email us at da-support@xfel.eu with any questions, or about corrections to the information below.

The documentation is available online at https://rtd.xfel.eu/docs/data-analysis-user-documentation/en/latest/

CHAPTER

ONE

OVERVIEW OF EUROPEAN XFEL DATA

1.1 Trains and pulses

European XFEL generates a "pulse train" of up to 2700 individual X-ray pulses, at a rate of 10 trains per second. Within a train, pulses arrive with a maximum frequency of 4.5 MHz (220 ns between pulses).

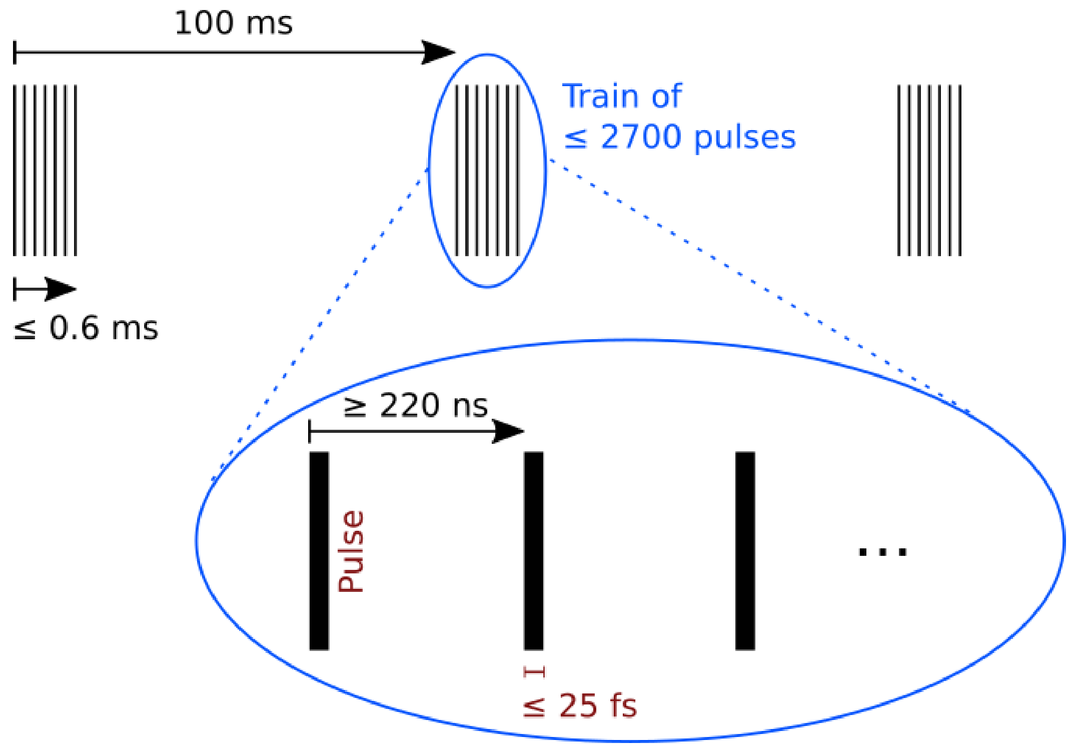


Fig. 1.1: The structure of trains and pulses at European XFEL.

Each train receives a unique integer train ID, which is used to find and match up data. Some kinds of data are recorded once per train, while others may be per pulse, or even higher sampling rates.

Information about the pulses generated in each SASE can be recorded with a `BUNCH_DECODER` device. This is a subset of the raw 'bunch pattern' data available through a `TIMESERVER` device. An example notebook illustrates how to read this information.

1.2 Sources and keys

Data is recorded from various sources. This includes the X-ray detectors which are the main data sources for many experiments, as well as separate sensors such as temperature sensors or devices to measure parameters of the beam. Controllable devices can also be data sources, to record things like motor positions.

Data within each source is organised by keys. For instance, an XGM is an apparatus for measuring the beam (see [Maltezopoulos2019] for details). `SA1_XTD2_XGM/XGM/DOOCS` is the source name of a specific XGM, and `beamPosition.ixPos` is one of its keys, recording the X position of the beam.

Note: Multi-module detectors typically have a separate source for each module. Various other devices (such as XGMs) are split for technical reasons into one 'control' and one 'instrument' source, although not all keys of a 'control' source can be controlled.

1.3 Tools and services provided by European XFEL

Broadly speaking, data analysis at EuXFEL is separated into two categories: online and offline. 'Online analysis' refers to realtime analysis of data being streamed during an experiment, and the programs we have for this run on the *Online cluster*. 'Offline analysis' refers to analysis of data that has been saved to files and migrated to the *Offline cluster*.

The facility provides:

- display of key experimental data during experiment (see *Karabo scenes*) through the Karabo control system ([Hauf2019]).
- near real-time data streams ("Karabo Bridge") for user software (see *Streaming from Karabo bridge*)
- access to saved *data files* through the Maxwell cluster at DESY
- detector calibration of both streaming (online) and saved (offline) data (see *Detector calibration*).
- software tools for online and offline analysis (see *Data analysis software*)
- *compute resources*, including:
  - *Online cluster* nodes for processing to provide feedback during data collection
  - The *Offline cluster* ('Maxwell'), managed by DESY, including JupyterHub access

[Fangohr2017] provides some context about the data analysis provisions.

1.3.1 Detector calibration

The fast X-ray detectors at European XFEL have some unusual features which pose challenges for processing their output into meaningful scientific data, including on-sensor memory cells and multi-gain-stage architectures. They are also capable of producing on the order of 10 GB per second, so calibrating the data is computationally intensive.

European XFEL aims to provide facility users with a fully corrected and calibrated dataset as the primary data product [KusterCal2014], so the burden of dealing with this calibration falls on the facility, not the users. This concept has been successfully deployed in other scientific communities such as astronomy, space science, and high-energy physics for more than a decade.

Users neither have to provide large amounts of computing resources nor have to have in-depth expertise on detector physics to obtain state-of-the-art corrected and calibrated datasets for their experiments and can thus focus on their scientific analysis. Additionally, comparisons between and data aggregation of different experiments and instruments are simplified as calibration becomes user-independent.

Within the proposal data folder, the `proc/` subfolder contains calibrated data, and `raw/` contains uncalibrated data. The *Karabo Bridge* data streams can also offer both calibrated or raw data - see *Streaming from Karabo bridge* for details.

1.3.2 Data policies

The following summarise the policies around certain kinds of data.

Raw data Raw data represents digitized detector signal, not altered by detector-specific corrections or calibrations; e.g., it is in the form of detector units such as analogue digital units (ADU). Vetoing, either by hard- or software triggers and zero-value suppression (e.g., by transferral to event lists), may have been performed and is irreversible. Raw data is the main archival data product at the European XFEL. It is not foreseen to be exported outside the facility [KusterCal2014].

Calib. data Calibrated data is generated from raw data by applying detector-specific corrections and transformation to physical units (calibration) - e.g., photons per pixel. Calibrated data is the standard data product with which users will be provided. It is not archived; instead, if a calibrated dataset is requested but not accessible through the online-cache or user-space anymore, it will be reprocessed on the fly from the raw data repository using the appropriate calibration parameters provided by the calibration database.

Alignment data Alignment data is generated from dedicated alignment measurements, providing the position of each detector pixel and detector module in three-dimensional space. It is stored in the detector coordinate system (i.e., as pixel coordinates) and no additional interpolation or coordinate transformation will be applied. Alignment data is part of the standard data products with which users will be provided.

See also:

Scientific data & data retention policies

1.4 Citations

CHAPTER

TWO

ONLINE ANALYSIS

Broadly speaking, there are four ways to analyse data in real-time:

1. *Karabo scenes*
2. *EXtra-foam*
3. *EXtra-metro*
4. *Streaming from Karabo bridge*

Each of them have different tradeoffs when it comes to usability and flexibility. Karabo scenes and EXtra-foam are designed to be easy to use, but extending them can be difficult. On the other hand, EXtra-metro and writing your own tools offer the most flexibility, but it usually takes a lot more work.

Note that given enough prior notice, custom analysis code can be created in collaboration with the users and made available for European XFEL experiments.

2.1 Karabo scenes

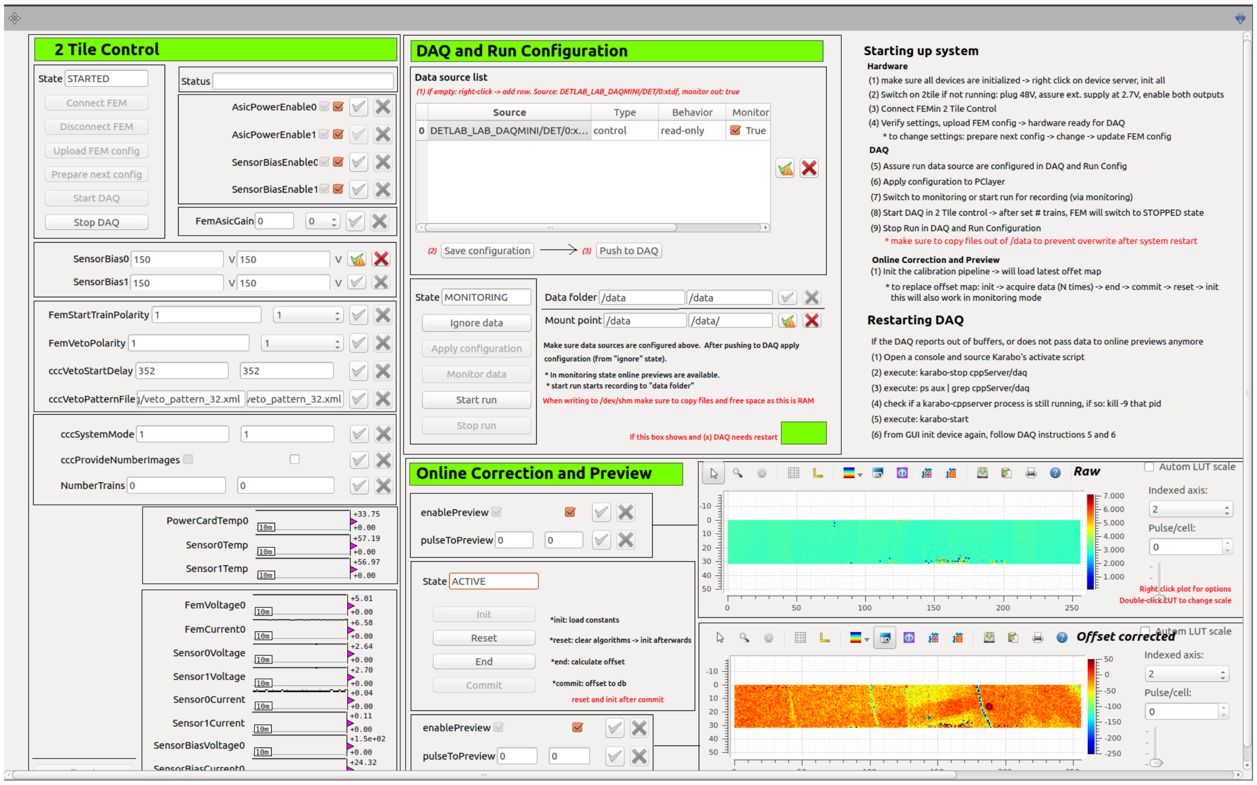
Karabo (the control system at EuXFEL) provides a customised GUI scene that displays relevant data during an experiment. E.g. the last captured corrected-Xray image from the detector, and some crucial other parameters such as train and pulse ID, beam energy, detector position, and for pump-probe experiments the pump-probe laser state, pump-probe laser delay, etc.

Specific scenes can be developed for each experiment's needs. An example is shown in Figure Fig. 2.1.

2.2 EXtra-foam

EXtra-foam is a tool that provides online data analysis and visualization for experiments, primarily for 2D detectors. Some features it provides are:

- Live preview of the detector. For multi-module detectors it is possible to specify the geometry for an accurate preview.
- Azimuthal integration.
- Analysis of ROIs on the detector, and the ability to correlate features of a ROI (e.g. intensity) with other data (e.g. time, motor position, etc).
- Recording and subtracting dark images.



State STARTED
Status

AsicPowerEnable0
AsicPowerEnable1
SensorBiasEnable0
SensorBiasEnable1
FemAsicGain 0 | 0

SensorBias0 150 V 150 V
SensorBias1 150 V 150 V

FemStartTrainPolarity 1 | 1
FemVetoPolarity 1 | 1
cccVetoStartDelay 352 | 352
cccVetoPatternFile /veto_pattern_32.xml | veto_pattern_32.xml

cccSystemMode 1 | 1
cccProvideNumberImages
NumberTrains 0 | 0

PowerCardTemp0 +33.75
Sensor0Temp +57.19
Sensor1Temp +56.97

FemVoltage0 +5.01
FemCurrent0 +6.58
Sensor0Voltage +2.64
Sensor1Voltage +2.70
Sensor0Current +0.04
Sensor1Current +0.11
SensorBiasVoltage0 +1.5e+02
SensorBiasCurrent0 +24.32

DAQ and Run Configuration

Data source list

(1) If empty: right-click -> add row. Source: DETLAB_LAB_DAQMINI/DET/0:xtdf, monitor out: true

	Source	Type	Behavior	Monitor
0	DETLAB_LAB_DAQMINI/DET/0:x...	control	read-only	True

(2) Save configuration → (3) Push to DAQ

State MONITORING

Data folder /data | /data
Mount point /data | /data/

Make sure data sources are configured above. After pushing to DAQ apply configuration (from "ignore" state).

* In monitoring state online previews are available.
* start run starts recording to "data folder"

When writing to /dev/shm make sure to copy files and free space as this is RAM

If this box shows and (x) DAQ needs restart

Starting up system

Hardware

(1) make sure all devices are initialized -> right click on device server, init all
(2) Switch on 2tile if not running: plug 48V, assure ext. supply at 2.7V, enable both outputs
(3) Connect FEMin 2 Tile Control
(4) Verify settings, upload FEM config -> hardware ready for DAQ
    * to change settings: prepare next config -> change -> update FEM config

DAQ

(5) Assure run data source are configured in DAQ and Run Config
(6) Apply configuration to PClayer
(7) Switch to monitoring or start run for recording (via monitoring)
(8) Start DAQ in 2 TIle control -> after set # trains, FEM will switch to STOPPED state
(9) Stop Run in DAQ and Run Configuration
    * make sure to copy files out of /data to prevent overwrite after system restart

Online Correction and Preview

(1) Init the calibration pipeline -> will load latest offet map
    * to replace offset map: init -> acquire data (N times) -> end -> commit -> reset -> init
    this will also work in monitoring mode

Restarting DAQ

If the DAQ reports out of buffers, or does not pass data to online previews anymore
(1) Open a console and source Karabo's activate script
(2) execute: karabo-stop cppServer/daq
(3) execute: ps aux | grep cppServer/daq
(4) check if a karabo-cppserver process is still running, if so: kill -9 that pid
(5) execute: karabo-start
(6) from GUI init device again, follow DAQ instructions 5 and 6

Online Correction and Preview

enablePreview
pulseToPreview 0 | 0

State ACTIVE

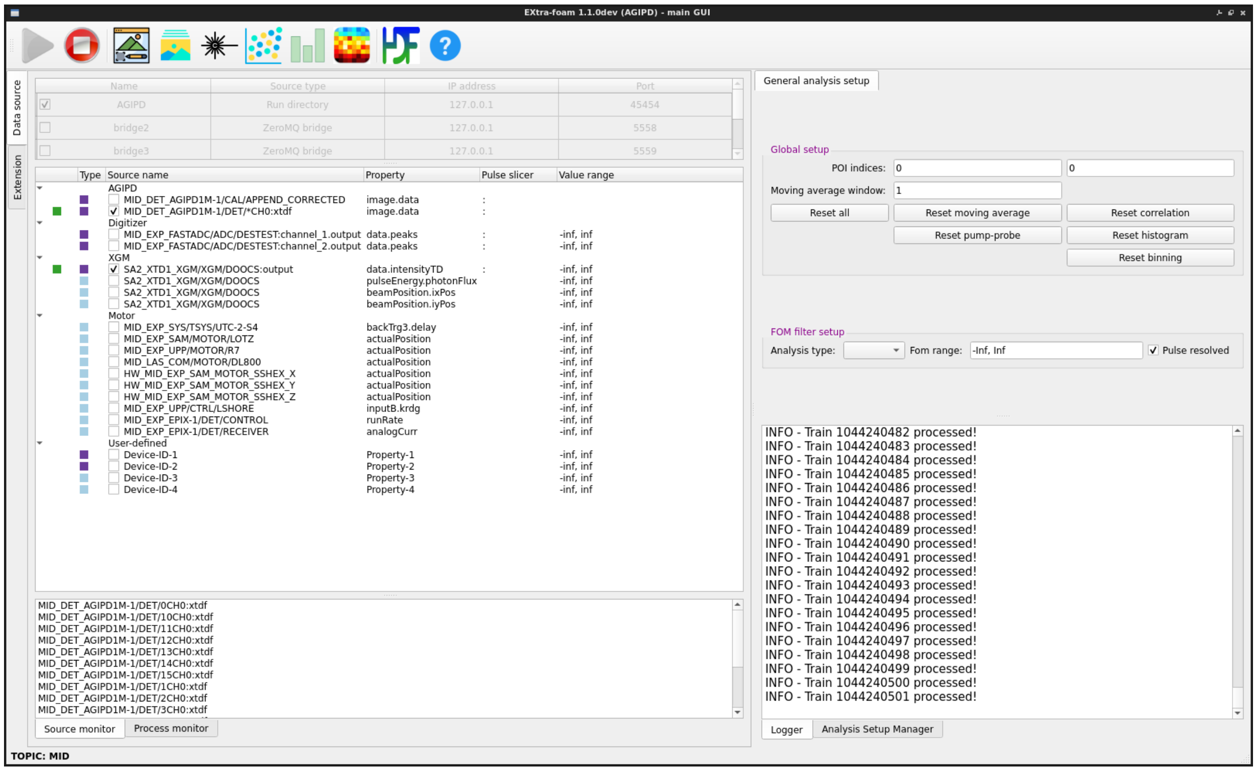
*init: load constants
*reset: clear algorithms -> init afterwards
*end: calculate offset
*commit: offset to db
reset and init after commit

enablePreview
pulseToPreview 0 | 0

Right click plot for options
Double-click LUT to change scale

Indexed axis: 2
Pulse/cell: 0

Fig. 2.1: Example of scene in Karabo for LPD detector.



Name	Source type	IP address	Port
AGIPD	Run directory	127.0.0.1	45454
bridge2	ZeroMQ bridge	127.0.0.1	5558
bridge3	ZeroMQ bridge	127.0.0.1	5559

Type	Source name	Property	Pulse slicer	Value range
AGIPD				
	MID_DET_AGIPD1M-1/CAL/APPEND_CORRECTED	image.data	:	
	✓ MID_DET_AGIPD1M-1/DET/*CH0:xtdf	image.data	:	
Digitizer				
	MID_EXP_FASTADC/ADC/DESTEST:channel_1.output	data.peaks	:	-inf, inf
	MID_EXP_FASTADC/ADC/DESTEST:channel_2.output	data.peaks	:	-inf, inf
XGM				
	✓ SA2_XTD1_XGM/XGM/DOOCS:output	data.intensityTD	:	-inf, inf
	SA2_XTD1_XGM/XGM/DOOCS	pulseEnergy.photonFlux		-inf, inf
	SA2_XTD1_XGM/XGM/DOOCS	beamPosition.ixPos		-inf, inf
	SA2_XTD1_XGM/XGM/DOOCS	beamPosition.iyPos		-inf, inf
Motor				
	MID_EXP_SYS/TSYS/UTC-2-S4	backTrg3.delay		-inf, inf
	MID_EXP_SAM/MOTOR/LOTZ	actualPosition		-inf, inf
	MID_EXP_UPP/MOTOR/R7	actualPosition		-inf, inf
	MID_LAS_COM/MOTOR/DL800	actualPosition		-inf, inf
	HW_MID_EXP_SAM_MOTOR_SSHEX_X	actualPosition		-inf, inf
	HW_MID_EXP_SAM_MOTOR_SSHEX_Y	actualPosition		-inf, inf
	HW_MID_EXP_SAM_MOTOR_SSHEX_Z	actualPosition		-inf, inf
	MID_EXP_UPP/CTRL/LSHORE	inputB.krdg		-inf, inf
	MID_EXP_EPIX-1/DET/CONTROL	runRate		-inf, inf
	MID_EXP_EPIX-1/DET/RECEIVER	analogCurr		-inf, inf
User-defined				
	Device-ID-1	Property-1		-inf, inf
	Device-ID-2	Property-2		-inf, inf
	Device-ID-3	Property-3		-inf, inf
	Device-ID-4	Property-4		-inf, inf

```
MID_DET_AGIPD1M-1/DET/0CH0:xtdf
MID_DET_AGIPD1M-1/DET/10CH0:xtdf
MID_DET_AGIPD1M-1/DET/11CH0:xtdf
MID_DET_AGIPD1M-1/DET/12CH0:xtdf
MID_DET_AGIPD1M-1/DET/13CH0:xtdf
MID_DET_AGIPD1M-1/DET/14CH0:xtdf
MID_DET_AGIPD1M-1/DET/15CH0:xtdf
MID_DET_AGIPD1M-1/DET/1CH0:xtdf
MID_DET_AGIPD1M-1/DET/2CH0:xtdf
MID_DET_AGIPD1M-1/DET/3CH0:xtdf
```

Global setup

POI indices: 0 | 0
Moving average window: 1

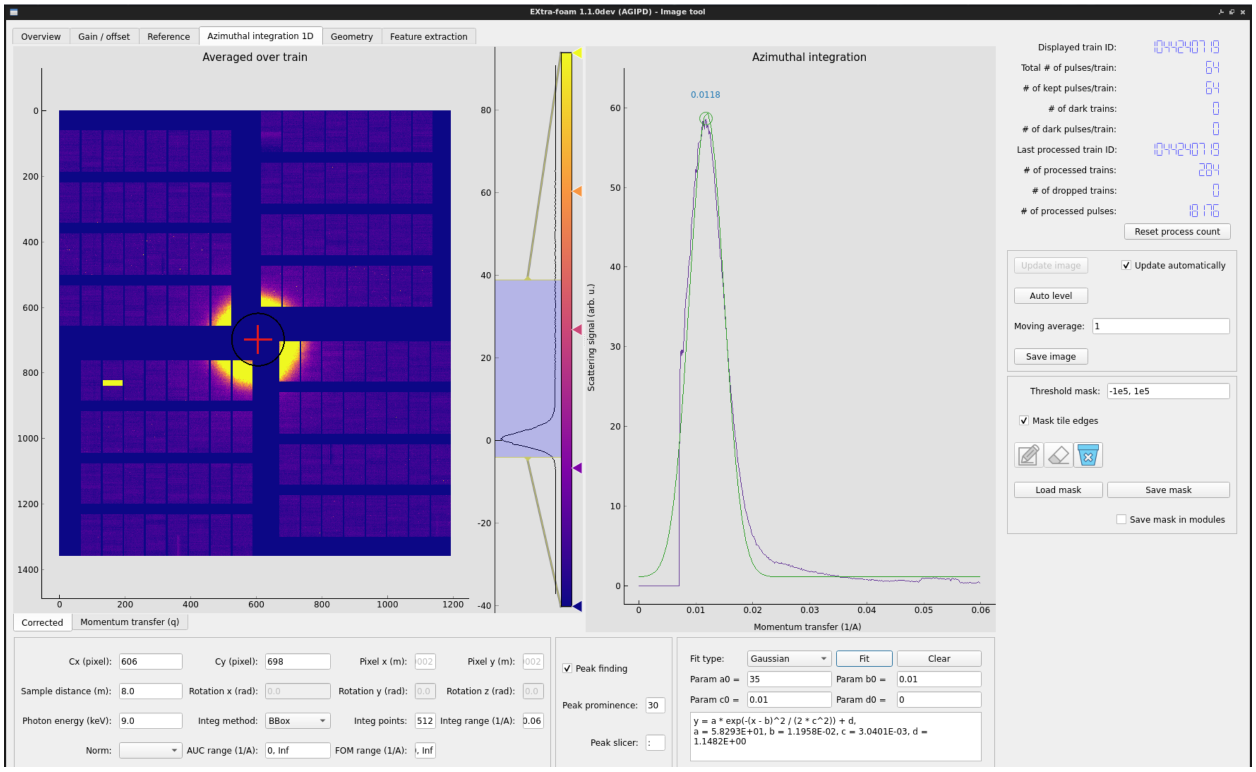
FOM filter setup

Analysis type: | Fom range: -Inf, Inf | ✓ Pulse resolved

```
INFO - Train 1044240482 processed!
INFO - Train 1044240483 processed!
INFO - Train 1044240484 processed!
INFO - Train 1044240485 processed!
INFO - Train 1044240486 processed!
INFO - Train 1044240487 processed!
INFO - Train 1044240488 processed!
INFO - Train 1044240489 processed!
INFO - Train 1044240490 processed!
INFO - Train 1044240491 processed!
INFO - Train 1044240492 processed!
INFO - Train 1044240493 processed!
INFO - Train 1044240494 processed!
INFO - Train 1044240495 processed!
INFO - Train 1044240496 processed!
INFO - Train 1044240497 processed!
INFO - Train 1044240498 processed!
INFO - Train 1044240499 processed!
INFO - Train 1044240500 processed!
INFO - Train 1044240501 processed!
```

TOPIC: MID

Fig. 2.2: The main window of EXtra-foam, where different data sources can be selected for processing.



Displayed train ID: 1044240719
Total # of pulses/train: 64
# of kept pulses/train: 64
# of dark trains: 0
# of dark pulses/train: 0
Last processed train ID: 1044240719
# of processed trains: 284
# of dropped trains: 0
# of processed pulses: 18176

✓ Update automatically
Moving average: 1
Threshold mask: -1e5, 1e5
✓ Mask tile edges
Save mask in modules

Cx (pixel): 606
Cy (pixel): 698
Pixel x (m): 002
Pixel y (m): 002
Sample distance (m): 8.0
Rotation x (rad): 0.0
Rotation y (rad): 0.0
Rotation z (rad): 0.0
Photon energy (keV): 9.0
Integ method: BBox
Integ points: 512
Integ range (1/A): 0.06
Norm:
AUC range (1/A): 0, Inf
FOM range (1/A): , Inf

✓ Peak finding
Peak prominence: 30
Peak slicer: :

Fit type: Gaussian
Param a0 = 35
Param b0 = 0.01
Param c0 = 0.01
Param d0 = 0

```
y = a * exp(-(x - b)^2 / (2 * c^2)) + d,
a = 5.8293E+01, b = 1.1958E-02, c = 3.0401E-03, d =
1.1482E+00
```

Fig. 2.3: Example of computing and fitting the $I(q)$ curve.

It works with AGIPD, LPD, JUNGFRAU, DSSC, ePix, and any train-resolved camera (e.g. Basler/Zyla cameras, etc).

In addition to the main program, there are separate programs called special suites that implement more specialized analysis for different experiments or instruments.

See also:

The EXtra-foam documentation has more details on its features.

EXtra-foam was designed for doing very well-defined kinds of analysis, so in some cases it's quite inflexible. If more flexibility is needed for an experiment, EXtra-metro is an option.

2.3 EXtra-metro

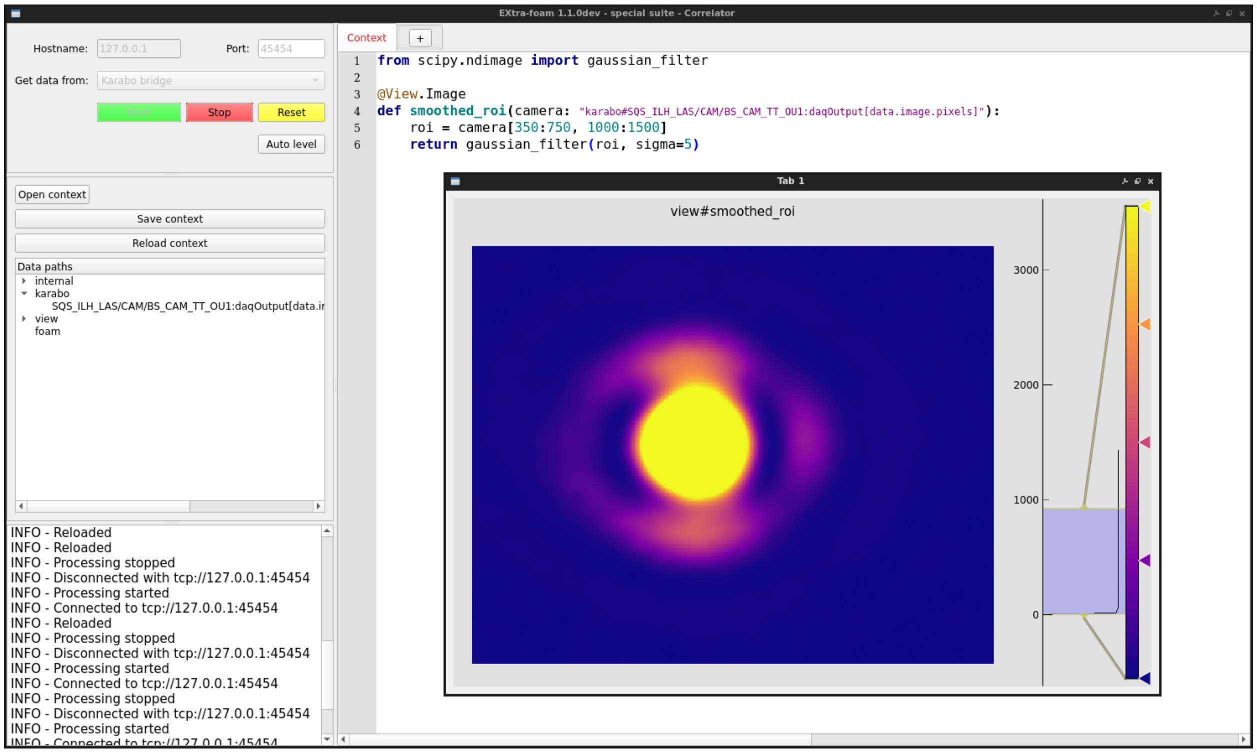
EXtra-metro is a processing framework designed to let you implement your own analysis. For example, let's say that you want to select an ROI from a camera and apply some smoothing to it:

```python
from scipy.ndimage import gaussian_filter

@View.Image
def smoothed_roi(camera: "karabo#SQS_ILH_LAS/CAM/BS_CAM_TT_OU1:output[data.image.pixels]
↪"):
    roi = camera[350:750, 1000:1500]
    return gaussian_filter(roi, sigma=5)
```

Here we've defined a function which takes in an `ndarray` of image data from a camera. It selects an ROI from the image, smooths it using a Gaussian filter from `scipy`, and returns the result. The function is decorated with `@View.Image`, which is a decorator that tells EXtra-metro that this function is a 'view' into some data, and it should be visualized as an image.

If this loaded into a program that embeds EXtra-metro and data is streamed to it, this function will be executed by EXtra-metro for each train and the results will be streamed out for visualization. For example, EXtra-foam embeds EXtra-metro in a special suite and if you run the above code you might see something like this:



```
from scipy.ndimage import gaussian_filter

@View.Image
def smoothed_roi(camera: "karabo#SQS_ILH_LAS/CAM/BS_CAM_TT_OU1:daqOutput[data.image.pixels]"):
    roi = camera[350:750, 1000:1500]
    return gaussian_filter(roi, sigma=5)
```

```
INFO - Reloaded
INFO - Reloaded
INFO - Processing stopped
INFO - Disconnected with tcp://127.0.0.1:45454
INFO - Processing started
INFO - Connected to tcp://127.0.0.1:45454
INFO - Reloaded
INFO - Processing stopped
INFO - Disconnected with tcp://127.0.0.1:45454
INFO - Processing started
INFO - Connected to tcp://127.0.0.1:45454
INFO - Processing stopped
INFO - Disconnected with tcp://127.0.0.1:45454
INFO - Processing started
INFO - Connected to tcp://127.0.0.1:45454
```

Fig. 2.4: Example of online analysis using EXtra-metro.

If you want to modify the analysis, such as tweak the `sigma` factor or add another `View`, it's possible to modify the code and reload it on-the-fly. This makes EXtra-metro excellent for anything that requires a lot of flexibility.

EXtra-metro has also been integrated with Karabo, so it's possible to create scenes with plots and parameters to control analysis code.

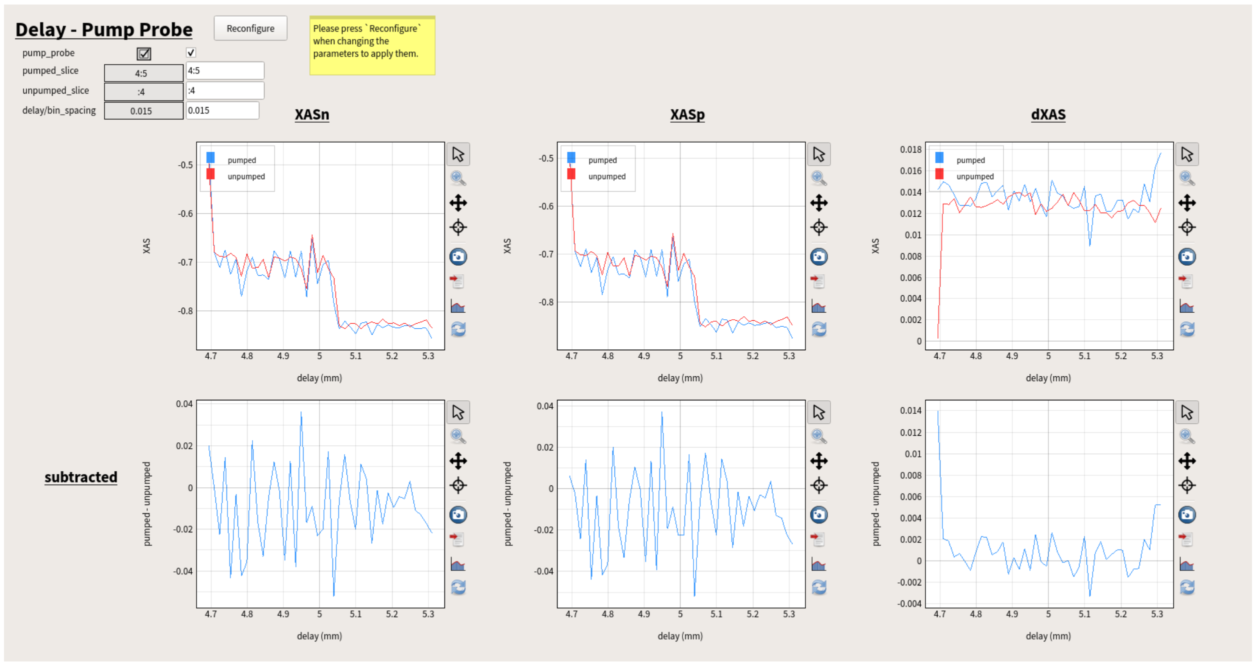
See also:

Check out the EXtra-metro documentation for more information on how to write analysis code, and the MetroProcessor documentation for information on the integration with Karabo.

2.4 Streaming from Karabo bridge

A *Karabo bridge* is a proxy interface to stream data to tools that are not integrated into the Karabo framework. It can be configured to provide any detector or control data. This interface is primarily for online data analysis (near real-time), but the extra_data Python package can also stream data from files using the same protocol, which may be useful for testing.

We provide *Karabo Bridge Clients* to receive this data in Python and C++, but you can also write your own code to receive the data if necessary. A custom client will need to implement the *Karabo Bridge protocol*.



Please press `Reconfigure` when changing the parameters to apply them.

pump_probe	✓	✓
pumped_slice	4:5	4:5
unpumped_slice	:4	:4
delay/bin_spacing	0.015	0.015

Fig. 2.5: A scene with analysis implemented in EXtra-metro, but controlled and visualized through a custom Karabo scene.

Any user tool will need to be running on the *Online cluster* to access the Karabo bridge devices that are streaming data. Some instruments already have bridges running and configured for sending detector data, but if not it is possible to set them up. Please tell your contact person at the instrument if you will be bringing custom tools that require a bridge.

Note: Data from MHz detectors at the EuXFEL (AGIPD, LPD, DSSC) might have a different data shape in file and in data streams. For each train data the shape in file follows: (pulse index, slow scan axis, fast scan axis). For performance constraints, the default shape offered for this data via data stream is: (module number, fast scan, slow scan, pulse index). This configuration can be changed on demand during beamtime and be switched between 2 shapes:

- online: (module number, fast scan, slow scan, pulse index)
- file-like: (pulse index, module number, slow scan, fast scan)

If you wish to use one or the other, communicate it to your contact person at the instrument.

Having file-like data shape online will add delay to the data stream, you might want to avoid this option if faster feedback is important for your experiment.

2.4.1 Simulation

In order to test tools with Karabo bridge, we provide a simulated server in the `karabo_bridge` Python package. This sends nonsense data with the same structure as real data. To start a server, run the command:

```
karabo-bridge-server-sim 4545
```

The number (4545) must be an unused local TCP port above 1024.

CHAPTER
THREE

DATA FILES

3.1 Data policy

The data policy of European XFEL is available at https://www.xfel.eu/users/policies/index_eng.html

3.2 Data folders

On both the *Online cluster* and the *Offline cluster*, European XFEL data is stored in `/gpfs/exfel/exp`. Each proposal has its own directory, so your data will be available at a path something like:

```
/gpfs/exfel/exp/SPB/201830/p900022/
```

The raw data from each run goes in a subfolder such as `raw/r0104`. Once this has been migrated to the offline cluster, corrected detector data will be automatically produced in another subfolder such as `proc/r0104`.

See also:

The contents of the proposal folder

3.3 Reading data in Python

We provide a Python package `extra_data` to read data from European XFEL.

See also:

extra_data documentation

3.4 Combining detector data from multiple modules

The pixel detectors (AGIPD and LPD) record data in separate files for each of their 16 modules.

The EXtra-data Python library can combine detector modules into a numpy array (example)

Alternatively, the modules can be combined in a single view as an HDF5 virtual dataset with the `extra-data-make-virtual-cxi` command, allowing the data to be processed by CrystFEL, for instance.

3.4.1 How To Make Virtual CXI data files

These commands are available in our Anaconda installation on Maxwell. To use them, first run:

```
module load exfel exfel_anaconda3
```

`extra-data-make-virtual-cxi` creates a file in CXI format with virtual datasets for the detector data. This can be used as input for CrystFEL, for example. Run it with a run directory:

```
extra-data-make-virtual-cxi /gpfs/exfel/exp/SPB/201830/p900022/proc/r0034
```

The file with the virtual dataset will be written in the `scratch` directory for the proposal. You can also use the `-o` option to specify an output file.

See also:

extra-data-make-virtual-cxi reference docs

Virtual datasets are a feature introduced with HDF5 1.10. If you are getting HDF5 errors trying to read these files, it may mean that you are using an older version of HDF5.

Checking access

Reading data from a virtual dataset doesn't fail if it can't access the source data files; it reads 0s (the fill value) instead. Use `hdf5-vds-check` to check that the source data is accessible:

```
hdf5-vds-check /gpfs/exfel/exp/SPB/201830/p900022/scratch/r0034_detectors_virt.cxi
```

Installing tools elsewhere

These commands are available in the module on Maxwell. If you need to install them elsewhere, they are available as Python packages:

```
pip install extra-data hdf5-vds-check
```

3.5 Geometry files

Geometry files specify the location of the detector modules in real space. Please contact your instrument scientist regarding obtaining geometry files for the detector at each instrument.

EXtra-geom is a Python library used to describe the physical layout of multi-module detectors at European XFEL, and to assemble complete detector images.

One geometry file for the FXE LPD detector is available online.

One geometry file for the SPB/SFX AGIPD1M detector is available from https://cxidb.org/id-83.html (Experiment by Anton Barty).

EXtra-geom can read both file formats, see for example Assembling detector data into images.

GeoAssembler can be used to create or adjust geometry files by visually moving detector quadrants around.

A more systematic provision of geometry files is in preparation.

3.6 Data format

Experimental data are taken in the context of the following categories:

- **instruments**: each instrument has their own label. For each instrument, there are multiple *cycles*:
- **cycle**: a scheduling period in which multiple user experiments will take place (of the order of months). Within each cycle, there are multiple *proposals*:
- **proposals**: (a.k.a. beamtimes) each user experiment proposal gets a number here. Within that proposal, users may carry out *runs* which can be logically associated to different experiments and samples:
- **runs**: when a user starts acquiring data, then a new run starts, until that data acquisition is stopped. Within a run, there are trains labelled by a unique *train Id*:
- **train id**: there are 10 pulse trains per second. Each train can carry multiple *pulses*
- **pulse id**: up to 2700 pulses per train, individually counted.Counter starts from zero for every train.

We distinguish different types of data:

- **Control data** has one entry for each train, even if the value changes less often than that.
- **Instrument data** may have zero, one or multiple entries per train. Your main experimental results, e.g. from X-ray detectors, will usually be instrument data.
- **Run data** is a superset of Control data, captured once per run.

Data is stored in HDF5 files; there may be tens to thousands of files in a single run. We aim to enable you to analyse data without needing to know the details of the file structure, e.g. by using EXtra-data in Python, or by *generating a CXI file* to represent a run. If you do need to read the EuXFEL HDF5 files yourself, however, the structure is described in the data files format page of the EXtra-data docs.

3.6.1 HDF5 chunking & compression

Both raw and corrected data may be stored using the HDF5 chunked layout. Some parts of the corrected data are compressed using the gzip compression filter in HDF5. In particular, detector gain stage and mask datasets compress well, saving a lot of disk space.

You can examine compression and chunk sizes using the GUI *HDF View* tool, our h5glance command line tool, or `h5ls -v`:

```
$ h5glance /gpfs/exfel/exp/XMPL/201750/p700000/raw/r0803/RAW-R0803-AGIPD00-S00000.h5 \
  INSTRUMENT/SPB_DET_AGIPD1M-1/DET/0CH0:xtdf/image/data
/gpfs/exfel/exp/XMPL/201750/p700000/raw/r0803/RAW-R0803-AGIPD00-S00000.h5/INSTRUMENT/SPB_
↪DET_AGIPD1M-1/DET/0CH0:xtdf/image/data
      dtype: uint16
      shape: 16000 × 2 × 512 × 128
   maxshape: Unlimited × 2 × 512 × 128
     layout: Chunked
      chunk: 16 × 2 × 512 × 128
compression: None (options: None)
...

$ h5ls -v /gpfs/exfel/exp/XMPL/201750/p700000/raw/r0803/RAW-R0803-AGIPD00-S00000.h5/
↪INSTRUMENT/SPB_DET_AGIPD1M-1/DET/0CH0:xtdf/image/data
Opened "/gpfs/exfel/exp/XMPL/201750/p700000/raw/r0803/RAW-R0803-AGIPD00-S00000.h5" with␣
↪sec2 driver.
```

(continues on next page)

(continued from previous page)

```
data                     Dataset {16000/Inf, 2/2, 512/512, 128/128}
    Location:  1:12333
    Links:     1
    Modified:  2017-11-20 04:57:44 CET
    Chunks:    {16, 2, 512, 128} 4194304 bytes
    Storage:   4194304000 logical bytes, 4194304000 allocated bytes, 100.00% utilization
    Type:      native unsigned short
```

The compressed datasets are stored with a single detector frame per chunk, to minimise the impact on analysis code reading the data.

If you observe pathologically slow reading, check whether you are accessing a compressed dataset with a chunk size larger than one frame. HDF5 decompresses an entire chunk at once, and it may be redoing this for each frame you read. You can avoid this by setting a cache size large enough to hold one complete chunk. The necessary C code looks something like this:

```
hid_t dapl = H5Pcreate(H5P_DATASET_ACCESS);
// Set a 32 MB cache size (calculate at least the size of one chunk)
H5Pset_chunk_cache(dapl, H5D_CHUNK_CACHE_NSLOTS_DEFAULT, 32 * 1024 * 1024, 1);
hid_t h5_dataset_id = H5Dopen(h5_file_id, ".../image/gain", dapl);
```

To benefit from chunk caching, you need to reuse the opened dataset ID for successive reads, instead of opening and closing it to read each frame.

3.7 Example data

Some example datasets are available so you can try reading the files before your experiment. There may be differences, e.g. in naming, when you collect new data, so it's a good idea to talk to the relevant instrument group and the data analysis group at European XFEL as well.

3.7.1 Example runs on Maxwell

We prepared an environment to mimic real experiment data cycle at the European XFEL. For this, we have a fake instrument called `XMPL` which contains runs giving an overview of the data to expect. This data is made available on Maxwell:

```
/gpfs/exfel/exp/XMPL/201750/p700000
```

It follows the same structure that each experiment have (see *Storage* for more details), and will be used to share different example of file format generated at the facility, from all instrument and detectors. These datasets are also linked to the Metadata catalog and information about the data (instrument, detector, sample, date, . . .) can be found there (MDC). Each run datasets comprise raw data (in `.../p700000/raw/run_id`) calibrated data (in `.../p700000/proc/run_id`) and a set of sample script to read the data (in `.../p700000/usr/run_id`).

List of sample data sets:

run id	description	Date	comments
r0001	instrument: SPB detector: AGIPD sample: Water	2018-04-03	commissioning
r0002	instrument: SPB detector: AGIPD sample: Lysozyme (liquid)	2018-04-03	commissioning
r0003	instrument: SPB detector: AGIPD sample: Lysozyme (liquid)	2018-04-03	commissioning
r0004	instrument: SPB detector: AGIPD sample: Lysozyme (liquid)	2018-04-03	commissioning
r0005	instrument: SPB detector: AGIPD sample: Lithium titanate	2018-08-18	AGIPD calibration
r0006	instrument: SPB detector: AGIPD sample: Lithium titanate[1]	2017-11-20	commissioning
r0007	instrument: FXE detector: LPD sample: aqueous solution of [Fe(bpy)3]2+	2017-09-18	User Run
r0008	tunnel: SA1_XTD2 device: XGM sample: n/a	2019-02-15	commissioning (XPD)
r0009	tunnel: SA3_XTD10 device: XGM sample: n/a	2019-02-15	commissioning (XPD)
r0010		2019-08-10	commissioning

Note: Mock data can be generated using the extra_data package, e.g.:

```
>>> from extra_data.tests.make_examples import make_agipd_example_file
>>> make_agipd_example_file('agipd_example.h5')

>>> from extra_data.tests.make_examples import write_file, Motor, ADC, XGM
>>> write_file('test_file.h5', [
..:     XGM('SPB_XTD1_XGM/XGM/MAIN'),
..:     Motor('SPB_DET_MOT/MOTOR/AGIPD_X'),
..:     Motor('SPB_DET_MOT/MOTOR/AGIPD_Y'),
..:     Motor('SPB_DET_MOT/MOTOR/AGIPD_Z'),
..:     ADC('SA1_XTD2_MPC/ADC/1', nsample=0, channels=(
..:         'channel_3.output/data',
..:         'channel_4.output/data',
..:         'channel_5.output/data'))
..:     ], ntrains=500, chunksize=50)
```

This only creates the structure of the files; the data will all be zeros.

3.7.2 Public data from EuXFEL in the CXIDB

The following entries at https://cxidb.org/ stem from user experiments done at our facility:

[1] Lithium titanate, spinel; nanopowder, <200 nm particle size (BET), >99%; CAS Number 12031-95-7; Empirical formula Li4Ti5O12; https://www.sigmaaldrich.com/catalog/product/aldrich/702277

CXIDB id	description	Deposition date	Publication DOI
id80	Authors: Wiedorn et al. instrument: SPB/SFX sample: Lysozyme wavelength: 1.33 Å (9.30 keV)	2018-08-13	10.1038/s41467-018-06156-7
id83	Authors: Wiedorn et al. instrument: SPB/SFX sample: -lactamase wavelength: 1.33 Å (9.30 keV)	2018-08-13	10.1038/s41467-018-06156-7
id87	Authors: Grünbein et al. instrument: SPB/SFX sample: Urease, Concanavalin A/B wavelength: 1.66 Å (7.47 keV)	2018-09-12	10.1038/s41597-019-0010-0
id98	Authors: Yefanov et al. instrument: SPB/SFX sample: Lysozyme wavelength: 1.33 Å (9.30 keV)	2020-02-07	10.1063/1.5124387
id100	Authors: Pandey et al. instrument: SPB/SFX sample: Photoactive Yellow Protein wavelength: 1.33 Å (9.30 keV)	2019-08-12	10.11577/1577287
id111	Authors: Gisriel et al. instrument: SPB/SFX sample: Photosystem I wavelength: 1.33 Å (9.30 keV)	2020-11-21	10.1038/s41467-019-12955-3
id152	Authors: Echelmeier et al. instrument: SPB/SFX sample: KDO8PS wavelength: 1.33 Å (9.30 keV)	2021-07-22	10.1038/s41467-020-18156-7

3.8 Downloading experiment data

Experiments at European XFEL typically generate large amounts of data - from around 10 TB up to petabytes from one beamtime. Because of this, we recommend that you analyse data on the Maxwell cluster rather than downloading it.

If you do need to download experimental data, there are two options:

- Using Globus (see Globus' How To). The endpoint is `euxfel#euxfel`, and you should use your XFEL credentials to authenticate. The metadata catalogue has links to Globus for each proposal & run.
- Using FTP from `ftp.xfel.eu`. You can use this with `lftp` (command line), or FileZilla (GUI). TLS encryption ('explicit FTPS') is required. The FTP server is not considered a critical service, so it may be unavailable at times.

CHAPTER

FOUR

COMPUTE ENVIRONMENT

4.1 User account

User accounts are created during the initial registration step in the UPEX portal. At this point the account can only be used for the UPEX itself. If the user account is associated to the accepted and scheduled proposal, then the account is upgraded 4 weeks before the first scheduled beamtime of the given user. For the first early user period, the time between the upgrade of the accounts and the start of the experiment can be shorter. The upgraded account allows the user to access additional services such as the online safety training, the metadata catalog , and the computing infrastructure of the European XFEL.

By default upgraded user accounts are kept in this state for 1 year after the user's last beamtime. An extension can be requested by the PI.

On-site guest WLAN (WiFi) is provided for all users. For those users with eduroam accounts provided by their home institute, access is straightforward. For those without eduroam account, a special registration procedure must be conducted to obtain guest access for the limited time period. After connecting to the XFEL-Guest network (also when using a network patch cable) and opening a web browser, the user will be able to register for the usage if guest network. The registration is valid for 10 days and 5 devices.

4.1.1 Tools

At different stages of the proposal, users are granted access to different services:

Stage	Access provided	Comments
Proposal submission	Access to User portal (UPEX)	
Approval of proposal and scheduling	Lightweight account	ca. 2 months before beam-time start
Preparation phase	Access to Metadata catalog and beamtime store filesystem. LDAP account upgraded for members of all accepted proposals.	First-time users: once A-form is submitted and accepted. Deadline for A-form submission is normally 4 weeks before beam-time start.
Beam time	Access to catalogs and dedicated online and offline services	
Data analysis	Access to catalogs and shared offline computing resources, initially limited to 1 year time period after beamtime.	

Importantly, first-time users should aim for a timely A-form submission. This ensures that they will have a time window of several weeks prior to the start of their beam-time when access to the Maxwell computing resources and the associated storage system (GPFS) is granted. An additional benefit of such access is that working with *example data* becomes possible, in order to get accustomed to the peculiarities of EuXFEL data and workflows.

4.2 Online cluster

During beam time, exclusive access to a dedicated online cluster (ONC) is available only to the experiment team members and instrument support staff.

European XFEL aims to keep the software provided on the ONC identical to that available on the *offline cluster* (which is the Maxwell cluster).

4.2.1 Online cluster nodes in SASE 1

Beamtime in SASE 1 is shared between the FXE and the SPB/SFX instruments, with alternating shifts: when the FXE shift stops, the SPB/SFX shift starts, and vice versa.

Within SASE1, there is one node reserved for the SPB/SFX experiments (`sa1-onc-spb`), and one node is reserved for the FXE experiments (`sa1-onc-fxe`). These can be used by the groups at any time during experiment period (i.e. during shifts and between shifts).

Both the SPB/SFX and the FXE users have *shared* access to another 7 nodes. The default expectation is that those nodes are using during the shift of the users, and usage stops at the end of the shift (so that the other experiment can start using the machines during their shift). These are `sa1-onc-01, sa1-onc-02`, `sa1-onc-03`, `sa1-onc-04, sa1-onc-05`, `sa1-onc-06`, `sa1-ong-01`.

Overview of available nodes and usage policy:

name	purpose
`sa1-onc-spb`	reserved for SPB/SFX
`sa1-onc-fxe`	reserved for FXE
`sa1-onc-01` to `sa1-onc-06`	shared between FXE, SPB use only during shifts
`sa1-ong-01`	shared between FXE, SPB GPU: Tesla V100 (16GB)

These nodes do not have access to the Internet.

The name `sa1-onc-` of the nodes stands for SAse1-ONlineCluster.

4.2.2 Online cluster nodes in SASE 2

Beamtime in SASE 2 is shared between the MID and the HED instruments, with alternating shifts: when the MID shift stops, the HED shift starts, and vice versa.

Within SASE2, there is one node reserved for the MID experiments (`sa2-onc-mid`), and one node is reserved for the HED experiments (`sa2-onc-hed`). These can be used by the groups at any time during experiment period (i.e. during shifts and between shifts).

Both the MID and the HED users have *shared* access to another 7 nodes. The default expectation is that those nodes are using during the shift of the users, and usage stops at the end of the shift (so that the other experiment can start using the machines during their shift). These are `sa2-onc-01, sa2-onc-02`, `sa2-onc-03`, `sa2-onc-04, sa2-onc-05`, `sa2-onc-06`, `sa2-ong-01`.

Overview of available nodes and usage policy:

name	purpose
`sa2-onc-mid`	reserved for MID
`sa2-onc-hed`	reserved for HED
`sa2-onc-01` to `sa2-onc-06`	shared between MID, HED use only during shifts
`sa2-ong-01`	shared between HED, MID GPU: Tesla V100 (16GB)

These nodes do not have access to the Internet.

The name `sa2-onc-` of the nodes stands for SAse2-ONlineCluster.

4.2.3 Online cluster nodes in SASE 3

Beamtime in SASE 3 is shared between the SQS and the SCS instruments, with alternating shifts, when the SQS shift stops, the SCS shift starts, and vice versa.

Within SASE3, there is one node reserved for the SCS experiments (`sa3-onc-scs`), and one node is reserved for the SQS experiments (`sa3-onc-sqs`). These can be used by the groups at any time during experiment period (i.e. during and between shifts).

Both SASE3 instrument users have *shared* access to another 7 nodes. The default expectation is that those nodes are used during users shift, and usage stops at the end of the shift (so that the other experiment can start using the machines during their shift). These are `sa3-onc-01`, `sa3-onc-02`, `sa3-onc-03`, `sa3-onc-04`, `sa3-onc-05`, `sa3-onc-06`, `sa3-ong-01`.

Overview of available nodes and usage policy:

name	purpose
`sa3-onc-scs`	reserved for SCS
`sa3-onc-sqs`	reserved for SQS
`sa3-onc-01` to `sa3-onc-06`	shared between SCS, SQS use only during shifts
`sa3-ong-01`	shared between SCS, SQS GPU: Tesla V100 (16GB)

These nodes do not have access to the Internet.

The name `sa3-onc-` of the nodes stands for SAse3-ONlineCluster.

Note that the usage policy on shared nodes is not strictly enforced. Scientists across instruments should liaise for agreement on usage other than specified here.

4.2.4 Access to online cluster

During your beamtime, you can SSH to the online cluster with two hops:

```
# Replace <username> with your username
ssh <username>@max-exfl-display003.desy.de  # 003 or 004
ssh sa3-onc-scs   # Name of a specific node - see above
```

This only works during your beamtime, not before or after. You should connect to the reserved node for the instrument you're using, and then make another hop to the shared nodes if you need them.

Workstations in the control hutches, and dedicated access workstations in the XFEL headquarters building (marked with an X in the map below) can connect directly to the online cluster.

`Location of the ONC workstations`

From these access computers, one can ssh directly into the online cluster nodes and also to the Maxwell cluster (see *Offline cluster*). The X display is forwarded automatically in both cases.

There is no direct Internet access from the online cluster possible.

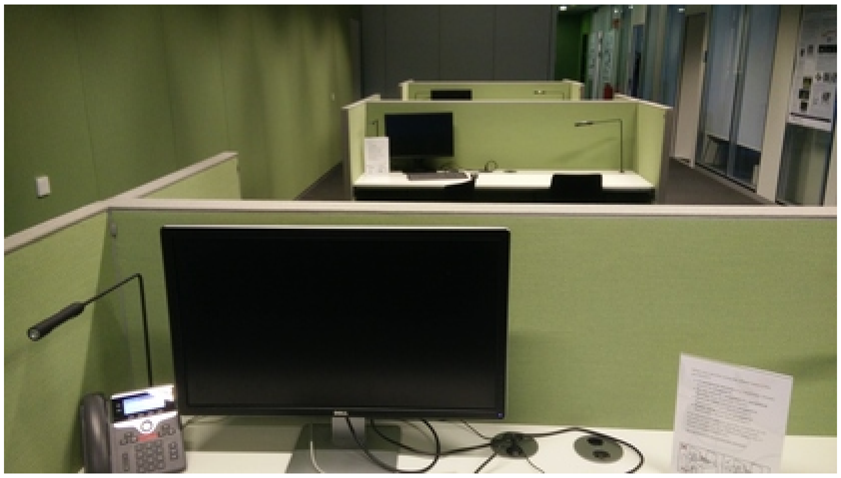


Fig. 4.1: Workstations at Level 1

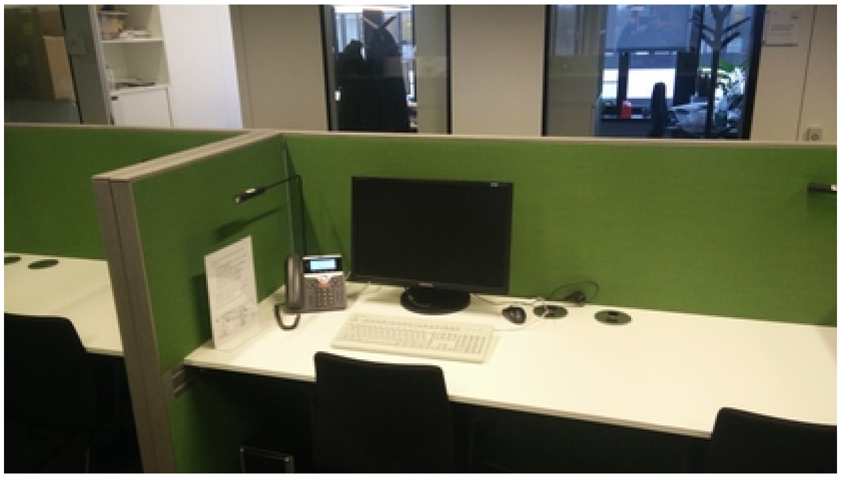


Fig. 4.2: Workstation at Level 2

4.2.5 Storage

The following folders are available on the online cluster for each proposal:

- **raw**: data files recorded from the experiment (read-only).
- **usr**: beamtime store. Into this folder users can upload some files, data or scripts to be used during the beamtime. This folder is mounted and thus immediately synchronised with a corresponding folder in the offline cluster. There is not a lot of space here (5TB).
- **scratch**: folder where users can write temporary data, i.e. the output of customized calibration pipelines etc. This folder is intended for large amounts of processed data. If the processed data is small in volume, it is recommended to use **usr**. Data in scratch is considered temporary, and may be deleted after your experiment.
- **proc**: Not currently used on the online cluster.

These folders are accessible at the same paths as on the offline cluster:

```
/gpfs/exfel/exp/<instrument>/<instrument_cycle>/p<proposal_id>/(raw|usr|proc|scratch)
```

Your home directory (`/home/<username>`) on online nodes is only shared between nodes within each SASE (e.g. `sa1-` nodes for SPB & FXE). It is also entirely separate from your home directory on the offline (Maxwell) cluster.

To share files between the online and the offline cluster, use the `usr` directory for your proposal, e.g. `/gpfs/exfel/exp/FXE/202122/p002958/usr`. All users associated with a proposal have access to this directory.

4.2.6 Access to data on the online cluster

Tools running on the online cluster can get streaming data from *Karabo Bridge*.

Data is also available in HDF5 files in the proposal's `raw` folder, and can be read using EXtra-data. But there are some limitations to reading files on the online cluster:

- You can't read the files which are currently being written. You can read complete runs after they have been closed, and you can read partial data from the current run because the writers periodically close one 'sequence file' and start a new one.
- Once runs are migrated to the offline cluster, the data on the online cluster may be deleted to free up space. You can expect data to remain available during a shift, but don't assume it will stay on the online cluster for the long term.
- *Corrected detector data* is not available in files on the online cluster, because the corrected files are generated after data is migrated to the offline cluster. You can get corrected data as a stream, though.

4.3 Offline cluster

The Maxwell cluster at DESY is available for data processing and analysis during and after the experiment. Users are welcome and encouraged to make themselves familiar with the Maxwell cluster and its environment well in advance of the beam time.

In the context of European XFEL experiments, the Maxwell cluster is also referred to as the "offline" cluster. Despite this name, you *can* connect to the internet from Maxwell. It is offline in that it can't stream data directly from the experiments, unlike the "*online cluster*".

See also:

General information about Maxwell (DESY wiki)

Details of hardware for EuXFEL

4.3.1 Getting access

When a proposal is accepted, the main proposer will be asked to fill out the "A-form" which, among information on the final selection of samples to be brought to the experiment, also contains a list of all experiment's participants. At time of submission of the A-form, all the participants have to have an active account in UPEX. This is the prerequesite for getting access to the facility's computing and data resources. After submission of the A-form, additional participants can be granted access to the experiment's data by PI request.

Users have access to:

- HPC cluster
- beamtime store, data repository and scratch space
- web based tools

4.3.2 Graphical login

To use Maxwell with a remote desktop, you can either:

- Go to https://max-exfl-display.desy.de:3443 in a web browser
- Or install FastX and connect to `max-exfl-display.desy.de`

See also:

Interactive login (DESY wiki)

4.3.3 Jupyter

Jupyter notebooks can be used through https://max-jhub.desy.de

See also:

Jupyter hub and notebooks

4.3.4 SSH access

```
ssh username@max-exfl-display.desy.de
```

Replace `username` with your EuXFEL username. Unlike most of the cluster, `max-exfl-display` is directly accessible from outside the DESY/EuXFEL network.

4.3.5 Running jobs

When you log in, you are on a 'login node', shared with lots of other people. You can try things out and run small computations here, but it's bad practice to run anything for a long time or use many CPUs on a login node.

To run a bigger job, you should submit it to SLURM, our queueing system. If you can define your job in a script, you can submit it like this:

```
sbatch -p upex -t 8:00:00 myscript.sh
```

- `-p` specifies the 'partition' to use. External users should use `upex`, while EuXFEL staff use `exfel`.

- `-t` specifies a time limit: `8:00:00` means 8 hours. If your job doesn't finish in this time, it will be killed. The default is 1 hour, and the maximum is 2 weeks.
- Your script should start with a 'shebang', a line like `#!/usr/bin/bash` pointing to the interpreter it should run in, e.g.:

  ```
  #!/usr/bin/bash

  echo "Job started at $(date) on $(hostname)"

  # To use the 'module' command, source this script first:
  source /usr/share/Modules/init/bash
  module load exfel exfel_anaconda3

  python -c "print(9 * 6)"
  ```

To see your running and pending jobs, run:

```
squeue -u $USER
```

Once a job starts, a file like `slurm-4192693.out` will be created - the number is the job ID. This contains the text output of the script, which you would see if you ran it in a terminal. The programs you run will probably also write data files.

SLURM is a powerful tool, and this is a deliberately brief introduction. If you are submitting a lot of jobs, it's worth spending some time exploring what it can do.

See also:

Running batch jobs (DESY wiki)

SLURM Quick Start User Guide

During beamtime

A reservation can be made for a beamtime so that your user group has exclusive access to a small subset of nodes. This is helpful if the computing cluster is busy, as you can always run some prioritised jobs, without waiting for nodes to become free in the general user partition.

As how many nodes to reserve, or if a reservation is needed, is dependent on the requirements of the experiment, speak to your local contact in the instrument group if you want to request a reservation.

The name of the reservation is `upex_NNNNNN` where `NNNNNN` is a 6-digit zero-padded proposal number, e.g. `upex_002416` would be the reservation for proposal number `2416`. Access to this reservation is permitted to anybody defined as a team member on myMdC.

Note that the reservation is only valid for 6 hours before and after a scheduled beamtime.

During your beamtime, if a reservation has been made, members of your group can submit jobs to the reservation with the `--reservation` flag on slurm commands. For example:

```
sbatch --reservation=upex_002416 ...
```

You can check the details of your reservation like this:

```
scontrol show res upex_002416
```

The output of this command tells you the period when the reservation is valid, the reserved nodes, and which usernames are allowed to submit jobs for it:

```
[@max-exfl001]~/reservation% scontrol show res upex_002416
ReservationName=upex_002416 StartTime=2019-03-07T23:05:00 EndTime=2019-03-11T14:00:00␣
↪Duration=3-14:55:00
Nodes=max-exfl[034-035,057,166] NodeCnt=4 CoreCnt=156 Features=(null) PartitionName=upex␣
↪Flags=IGNORE_JOBS
TRES=cpu=312
Users=bob,fred,sally Accounts=(null) Licenses=(null) State=ACTIVE BurstBuffer=(null)␣
↪Watts=n/a
```

To view all of the current reservations:

```
scontrol show reservations --all
```

4.3.6 Software available

The EuXFEL data analysis group provides a number of relevant tools, described in *Data analysis software*. In particular, a Python environment with relevant modules can be loaded by running:

```
module load exfel exfel_anaconda3
```

See also:

List of applications provided by DESY on Maxwell

4.3.7 Storage

Users will be given a single experiment folder per beam time (not per user) through which all data will be accessible, e.g:

```
/gpfs/exfel/exp/<instrument>/<instrument_cycle>/p<proposal_id>/(raw|usr|proc|scratch)
```

Storage	Quota	Permission	Lifetime	comments
raw	None	Read	5 years	Raw experiment data
proc	None	Read	6 months	Processed data e.g. calibrated
usr	5TB	Read/Write	2 years	User data, results
scratch	None	Read/Write	6 months	Temporary data (lifetime not guaranteed)

The data lifetimes above are minima set by the data policy - data may be kept for longer than this. However, data may be moved from storage designed for fast access to slower archival systems, even within these minimum lifetimes.

4.3.8 Synchronisation

The data in the `raw` directories are moved from the online cluster (at the experiment) to the offline (Maxwell) cluster as follows:

- when the run stops (user presses button), the data is flagged that it can be copied to the Maxwell cluster, and is queued to a copy service (provided by DESY). The data will be copied without the user noticing.
- Once the data is copied, the data is 'switched' and becomes available on the offline cluster.

  The precise time at which this switch happens after the user presses the button cannot be predicted: if the data is copied already (in the background), it could be instantaneous, otherwise the copy process needs to finish first.

- The actual copying process (before the switch) could take anything between minutes to hours, and will depend on (i) the size of the data and (ii) how busy the (DESY) copying queue is.
- The `usr` folder is mounted from the Maxwell cluster, and thus always identical between the online and offline system. However, it is not optimised for dealing with large files and thus potentially slow for lager files. There is a quota of 5TB.

4.4 Running containers

Singularity is available on both the online and offline cluster. It can be used to run containers built with Singularity or Docker.

Running containers with Docker is experimental, and there are some complications with filesystem permissions. We recommend using Singularity to run your containers, but if you need Docker, it is available.

- On the online cluster, Docker needs to be enabled for your account. Please email it-support@xfel.eu to request it.
- On the offline cluster, Docker only works on nodes allocated for SLURM jobs (see *Running jobs*), not on login nodes.

See also:

Docker on Maxwell (DESY wiki)

4.5 Compute environment FAQ

Frequently asked questions

tbd

CHAPTER

FIVE

JUPYTER HUB AND NOTEBOOKS

Documentation based on frequently asked questions

5.1 Usage of Jupyter running on the Maxwell cluster

The easiest way to run Jupyter Notebooks on the Maxwell cluster is to use the *JupyterHub* portal.

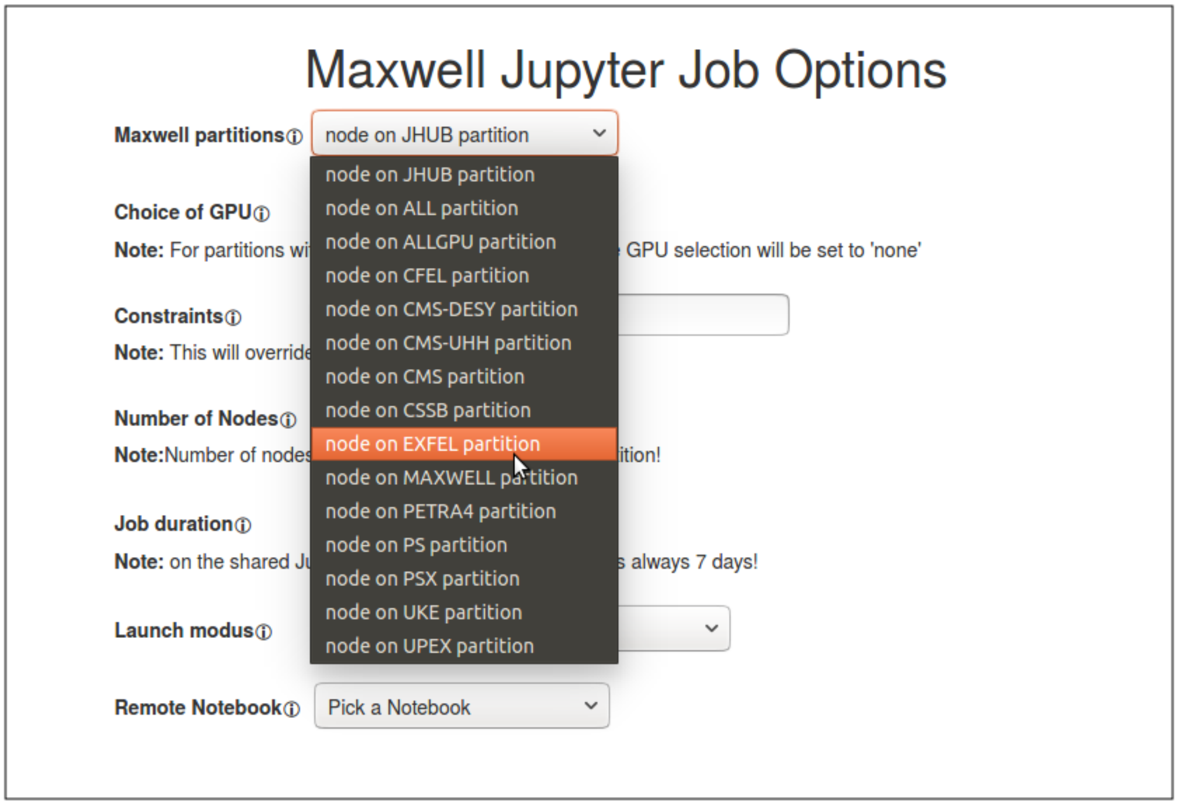
Alternatively, a notebook server can be manually started on Maxwell, and port forwarding can be used to direct a local web-browser to that server.

Both approaches will be explained in the following.

5.1.1 How to access Max-Jhub and select a partition

Log in to Jupyter Hub on the Maxwell cluster using your DESY account. Once logged in, perform the following steps:

- Choose a suitable partition on Maxwell from the upper drop-down menu; the choice should target one that your group has access to. Note that if you experience start-up problems with one partition, you may want to try another one.



Maxwell Jupyter Job Options

Maxwell partitions: node on JHUB partition

node on JHUB partition
node on ALL partition
node on ALLGPU partition
node on CFEL partition
node on CMS-DESY partition
node on CMS-UHH partition
node on CMS partition
node on CSSB partition
node on EXFEL partition
node on MAXWELL partition
node on PETRA4 partition
node on PS partition
node on PSX partition
node on UKE partition
node on UPEX partition

Choice of GPU

Note: For partitions wi GPU selection will be set to 'none'

Constraints

Note: This will override

Number of Nodes

Note: Number of nodes ition!

Job duration

Note: on the shared J s always 7 days!

Launch modus

Remote Notebook: Pick a Notebook

- JHUB (Shared Jupyter partition): the default, it will assign a shared node with a maximum time limit of 7 days. This partition is suited for everyone, but your code may run slower and won't be able to use as much memory as on a dedicated node.
- EXFEL: a partition suited for EuXFEL staff. Like all partitions except JHUB, it will assign a dedicated node with a maximum time limit of 8 hours.
- UPEX: a partition suited for EuXFEL users.

- Select a `Job duration` (ignored for the JHUB partition).
- Click on `Spawn`.
- Browse through your home (`/home/<username>`) directory to select or start a new Jupyter project.
- Jupyter projects stored e. g. on:

```
/gpfs/exfel/exp/<instrument>/<instrument_cycle>/p<proposal_id>/(usr|scratch)
```

  are also accessible via symbolic link in user's home directory.

Remember to disconnect by clicking on `Control Panel` and then `Stop My Server` to free the reserved node.

This is the easiest way to run Jupyter Notebooks on the Maxwell cluster (it does not require port forwarding or machine reservations). The *xfel* kernel includes many useful modules for working with EuXFEL data; this should be available as an option when you create a new notebook.

Some details are explained at https://confluence.desy.de/display/MXW/JupyterHub+on+Maxwell .

5.1.2 How to use Jupyter notebooks on Maxwell with a manual server

The following protocols are useful if you cannot use JupyterHub, or want to have more fine grained control about the resources you use in your notebook session.

To make the Jupyter notebook-server tool available on Maxwell, it is recommended to employ the Python anaconda distribution provided by EuXFEL: To get that Anaconda distribution (based on Python 3) and the Jupyter notebook as an executable into the PATH, the command is:

```
module load exfel exfel_anaconda3
```

For example (as of 15 January 2018):

```
[user@max-exfl001]~% which python
/usr/bin/python                                # this is the linux system python
[user@max-exfl001]~% module load exfel exfel_anaconda
[user@max-exfl001]~% which python
/gpfs/exfel/sw/software/xfel_anaconda3/X.X/bin/python   # xfel anaconda's python
[user@max-exfl001]~% which ipython
/gpfs/exfel/sw/software/xfel_anaconda3/X.X/bin/ipython
[user@max-exfl001]~% which jupyter-notebook
/gpfs/exfel/sw/software/xfel_anaconda3/X.X/bin/jupyter-notebook
[user@max-exfl001]~% jupyter-notebook --version
6.0.3
```

The `` `module load exfel exfel_anaconda3` `` will create a so called jupyter kernel that makes the all exfel python modules available in jupyter. If you prefer, you can create your own anaconda installation in your user account.

Procedure from inside the DESY network

1. Request a node just for you to carry out the analysis.

   To do this, we need to login to `max-exfl.desy.de`, and then use the `salloc` command to request - for example - one Node (N=1) for 8 hours (t=08:00:00) in the upex partition (p=upex):

   ```
   [MYLOCALUSERNAME@COMP]$ ssh USERNAME@max-exfl.desy.de
   [USERNAME@max-exfl001]$ salloc -N 1 -p upex -t 08:00:00
   ```

   The system will respond (if it can allocate a node for you) with something like:

   ```
   salloc: granted job allocation ....
   salloc: ...
   salloc: Nodes max-exfl072 are ready for job
   ```

   The node we can now use (for the next 8 hours) is `max-exfl072`.

   In the commands above replace `USERNAME` with your username on the Maxwell cluster (same as XFEL/DESY user name).

   See also:

   Running jobs for more information on slurm commands.

2. Now we open a new terminal on our local machine `COMP` and ssh directly to that node `max-exfl072`. We also forward port 8432 on node `max-exfl072` (or whichever one is assigned) to the port 8432 on our local machine `COMP`):

```
[MYLOCALUSERNAME@COMP]$ ssh -L 8432:localhost:8432 max-exfl072
```

3. Now we can start the Jupyter Notebook and need to tell it to use port 8432, and not to start a browser automatically:

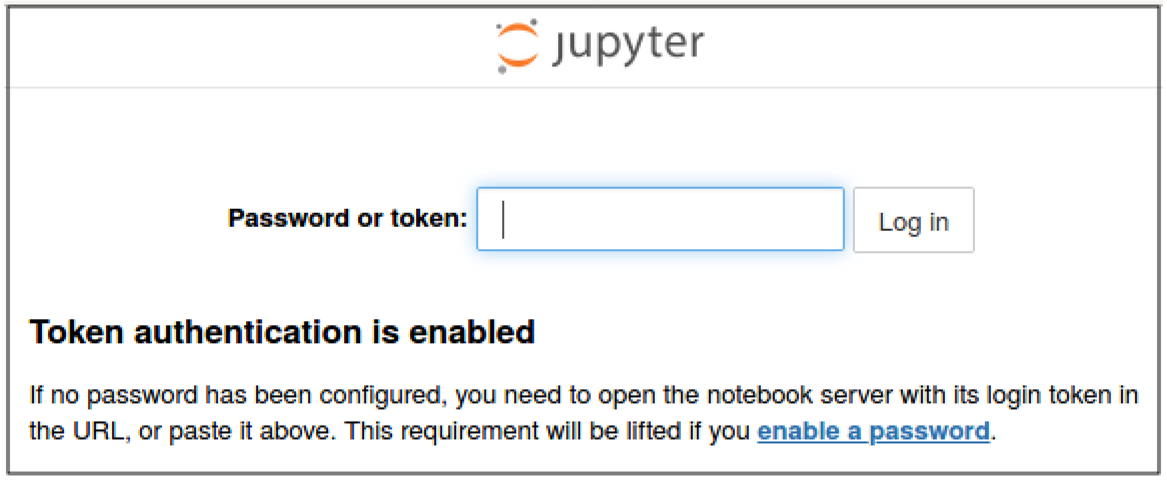
```
[USERNAME@max-exfl072]$ module load exfel exfel_anaconda3
[USERNAME@max-exfl072]$ jupyter-notebook --port 8432 --no-browser
```

4. Then open a browser window on your machine `COMP` and ask it to connect to port 8432 `https://localhost:8432`

5. If you haven't set a password before, the standard output in the terminal will end with a text block like:

```
To access the notebook, open this file in a browser:
   file:///home/dallanto/.local/share/jupyter/runtime/nbserver-14812-open.html
Or copy and paste one of these URLs:
   http://localhost:8432/?token=eac81f1482be0112caee9aff8eb13745f976109f0c80f9c6
or http://127.0.0.1:8432/?token=eac81f1482be0112caee9aff8eb13745f976109f0c80f9c6
```

Only then, after having connected to the server as per (4)., you will get an authentication dialogue like this:



Token authentication is enabled

If no password has been configured, you need to open the notebook server with its login token in the URL, or paste it above. This requirement will be lifted if you enable a password.

and you have to enter the token (e. g. `c81f1482be0112caee9aff8eb13745f976109f0c80f9c6`) by copy-paste from the terminal message.

Alternatively, you can enter the entire URL `http://localhost:8432/?token=c81f...` into the browser address field, at step (4.), by copy-paste, in the first place. This saves you the extra authentication page.

If for some reason the terminal text from the server startup is not available, one can also retrieve the token by typing `jupyter-notebook list` to a command-line prompt of the node where the notebook server runs.

However, if a password has been set, the authentication dialogue will simplify and you will be asked for that password only.

Summary

- request node
- then forward port and start jupyter there:

```
ssh -L 8432:localhost:8432 USERNAME@max-exfl072
module load exfel exfel_anaconda3
jupyter-notebook --port 8432 --no-browser
```

- open browser on local machine (`https://localhost:8432`)

Procedure from outside the DESY network

If your machine `LAPTOP` is outside the XFEL/DESY network, you need to get into the DESY network via `bastion.desy.de`. In this case, we recommend that you first ssh to `max-exfl` (via `bastion.desy.de`) to create a node allocation (in our example below for 8 hours):

```
[MYLOCALUSERNAME@LAPTOP]$ ssh USERNAME@bastion.desy.de
[USERNAME@bastion01]$ ssh max-exfl
[USERNAME@max-exfl001]$ salloc -N 1 -p upex -t 08:00:00
```

Once this is done, we need to connect a port on your local machine with the port that the jupyter notebook listens to on `max-exfl072`. We need to go via bastion, i.e.

```
[MYLOCALUSERNAME@LAPTOP]$ ssh -L 8432:localhost:8432 bastion.desy.de -t ssh -L 8432:
↪localhost:8432 max-exfl072
[USERNAME@max-exfl072] module load exfel exfel_anaconda3
[USERNAME@max-exfl072] jupyter-notebook --port=8432 --no-browser
```

Then open a browser with URL `https://localhost:8432` on local machine `LAPTOP`.

Procedure from hutch computers - proxy issue

The hutch computers are isolated from the internet, but it is possible to set a proxy to access the internet - including max-jhub.

To fix this you must disable the SOCKS proxy, the correct settings are:

1. Open preferences
2. Scroll down, click network settings
3. Tick "Manual proxy configuration"
4. Set the HTTP Proxy to `exflproxy01.desy.de` and the port to 3128
5. Make sure that the "SOCKS Host" and "Port" entry is empty
6. Tick "Also use this proxy for FTP and HTTPS"
7. Select OK to save these settings

Note that there is an additional issue with Firefox where, even though you deleted the url and the setting was applied, the UI still shows an entry. If this happens you can check the actual setting by going to `about:config` in Firefox, searching for `network.proxy.socks`, and deleting the values for `network.proxy.socks` and `network.proxy.socks_port`.

Technical comments

You can ignore the comments below unless you run into difficulties, or what to get more background information.

- If your laptop is connect to eduroam, you are *outside* the XFEL/DESY network, and need to follow instructions in *From outside the DESY network*.
- We have used 8432 as the port in the examples above. There is no particular reason for doing so, other than the port not being used by any other famous software (see list on Wikipedia), and the port number being greater than 1024.

You can chose other ports as you like. Using different ports also allows to run multiple Jupyter Notebook servers on the same node (each listening to a particular node).

By default (i.e. if we don't use the `--port` switch when starting the Jupyter Notebook), port 8888 is used.

5.2 Usage of Jupyter running on the EuXFEL online cluster

Once logged into a machine in one of the online cluster nodes available (check *Online cluster* for the proper host aliases), you can use Jupyter either:

- Directly from the node of the instrument itself by running Firefox from a terminal and going to the node name + .desy.de (e.g. `sa1-onc-spb.desy.de`). It should take you directly to the Jupyterhub page from where you can login with your credentials.

  The Jupyter instance will run from the node itself, meaning that the rendering could not be optimal.

- On your local computer after you forward the necessary ports. This port-forwarding will be slightly different depending if you are inside or outside the control network:

  – Outside the control network (e.g. from your office pc), a double port-forwarding will be necessary:

    ```
    ssh -L 8000:localhost:8000 exflgateway -t ssh -L 8000:localhost:8000 sa1-onc-fxe
    ```

    By using the port 8000, you will be able to open an instance of Jupyterhub service directly on the browser of your local machine if you go to:

    `http://localhost:8000`

    and login with your credentials.

  – Inside the control network (e.g. from a control room computer):

    ```
    ssh -L 8000:localhost:8000 sa1-onc-fxe
    ```

5.2.1 Manual Jupyter notebook server on online cluster nodes

If you are using a *shared* access node during shifts, it is also possible to manually start a Jupyter instance from inside the control network (e.g. from a control room computer).

The more straight-forward way is to render Jupyter on the node itself. In order to do so, firstly you need to load the notebook-server tool available on Maxwell (see *Start manual server*) by typing:

```
module load exfel exfel_anaconda3
```

Once loaded, you can simply type in the terminal:

```
jupyter-notebook
```

to start a Jupyter session.

If you need better rendering, you must forward ports in order to use the browser of your local machine. To do so, type on your local machine:

```
ssh -L 8008:localhost:8008 sa1-onc-03
```

After loading the notebook-server tool (see above), you will be able to render the Jupyter notebook session, running on the remote node (sa1-onc-03, on the browser of your local machine by typing:

```
jupyter-notebook --port 8008 --no-browser
```

And opening in the browser the following URL:

```
http://localhost.8008
```

CHAPTER

SIX

DATA ANALYSIS SOFTWARE

6.1 Software to access and inspect data

6.1.1 EXtra Data

`extra_data` is a Python library for accessing and working with data produced at European XFEL. It can:

- Conveniently access data from an experimental run, which is often spread across dozens of 'sequence' files.
- Read data into pandas and xarray, two popular Python libraries which support powerful, efficient data analysis.
- Assemble image data for multi module detectors like LPD and AGIPD, using geometry files in different formats.
- Stream data from files over a ZeroMQ socket. This stream of data can then be accessed using *Karabo Bridge Clients* clients, to test live-processing tools with data from real experiments.

See also:

extra_data documentation

6.1.2 Karabo Bridge Clients

We provide client libraries in Python and C++ to receive data from the *karabo bridge*, allowing users to integrate their tools with the karabo framework and receive live data during an experiment run.

Karabo Bridge Python client

The `karabo_bridge` Python package provides a client interface to receive data from the Karabo bridge.

`class karabo_bridge.Client`(*endpoint*, *sock='REQ'*, *ser='msgpack'*, *timeout=None*, *context=None*)

Karabo bridge client for Karabo pipeline data.

This class can request data to a Karabo bridge server. Create the client with:

```
from karabo_bridge import Client
krb_client = Client("tcp://153.0.55.21:12345")
```

then call `data = krb_client.next()` to request next available data container.

Parameters

- **endpoint** (*str*) – server socket you want to connect to (only support TCP socket).
- **sock** (*str, optional*) – socket type - supported: REQ, SUB.

- **`ser`** (*`str`*, *DEPRECATED*) – Serialization protocol to use to decode the incoming message (default is msgpack) - supported: msgpack.
- **`context`** (*`zmq.Context`*) – To run the Client's sockets using a provided ZeroMQ context.
- **`timeout`** (*`int`*) – Timeout on **:method:`next`** (in seconds)

  Data transfered at the EuXFEL for Mega-pixels detectors can be very large. Setting a too small timeout might end in never getting data. Some example of transfer timing for 1Mpix detector (AGIPD, LPD):

  > 32 pulses per train (125 MB): ~0.1 s 128 pulses per train (500 MB): ~0.4 s 350 pulses per train (1.37 GB): ~1 s

Raises

- `NotImplementedError` – if socket type or serialization algorythm is not supported.
- **`ZMQError`** – if provided endpoint is not valid.

`next`()

Request next data container.

This function call is blocking.

Returns

- **data** (*dict*) – The data for this train, keyed by source name.
- **meta** (*dict*) – The metadata for this train, keyed by source name.

  This dictionary is populated for protocol version 1.0 and 2.2. For other protocol versions, metadata information is available in *data* dict.

Raises `TimeoutError` – If timeout is reached before receiving data.

Data container

The data object returned by *`Client.next()`* is a tuple of two nested dictionaries, holding data and metadata respectively.

The first level of keys in both dictionaries are source names, which typically correspond to Karabo device names.

In the data dictionary, each source has a dictionary representing a flattened Karabo hash, with dots delimiting successive key levels. The values are either basic Python types such as strings and floats, or numpy arrays.

In the metadata dictionary, each source has a dictionary with keys: `source`, `timestamp`, `timestamp.tid`, `timestamp.sec`, `timestamp.frac` and `ignored_keys`, which is a list of strings identifying keys which were filtered out of the data by configuration options on the bridge server.

```
(
    {   # Metadata
        'SPB_DET_AGIPD1M-1/DET/0CH0:xtdf': {
            'source': 'SPB_DET_AGIPD1M-1/DET/0CH0:xtdf',
            'timestamp': 1526464869.4109755,
            # Timestamps can have attosecond resolution: 18 fractional digits
            'timestamp.frac': '410975500000000000',
            'timestamp.sec': '1526464869',
            'timestamp.tid': 10000000001,
            'ignored_keys': []
        },
```

(continues on next page)

(continued from previous page)

```
    },
    {   # Data
        'SPB_DET_AGIPD1M-1/DET/0CH0:xtdf': {
            'image.data': array(...),
            'header.pulseCount': 64,
            # ... other keys
        }
    }
)
```

Karabo Bridge C++ client

The *karabo bridge* C++ client provides a client interface to receive data from *karabo bridge*.

Example Usage for accessing image data from the stream.

```
#include "kb_client.hpp"
using namespace std;

int main (int argc, char* argv[]) {
         std::string addr;

         if (argc >= 2) addr = argv[1];

         else throw std::invalid_argument("Server address required!");

         karabo_bridge::Client client;
         client.connect(addr);

         auto livedata = client.next()

         for (auto it = livedata.begin(); it != livedata.end(); ++it)
         {
           if (it -> first == "SPB_DET_AGIPD1M-1/DET/detector")
           {
             karabo_bridge::kb_data data(it->second);
             auto images = data.array["image.data"].as<std::vector<uint32_t>>();
           }
         }

 }
```

Details on installation and usage are available here

Karabo Bridge Recorder

Karabo bridge recorder is a utility to record and replay data sent through the Karabo bridge.

This is for software engineering purposes, to conveniently test code with the exact data format sent by the bridge. We maintain a simulator for the same purpose, but sometimes the data format changes. Making a new recording is much quicker than updating the simulator.

Do not use this to store scientific data for later access! The file format is undocumented, and we will not support such use. EuXFEL has much better ways to store data; ask your contact people if you're not sure how to record or access data.

Usage:

```
# Record some data (creates a new folder)
karabo-bridge-record tcp://10.253.0.51/4510

# Play back recorded data
karabo-bridge-replay tcp://0.0.0.0:4545 karabo-bridge-recording-2019-02-19T16.49.25Z/
```

It is **recommended** to use this application from pre-installed path on the on- and offline cluster (See *EXtra-xwiz*)

Karabo Bridge protocol

The Karabo bridge protocol is based on ZeroMQ and msgpack.

The connection can use one of two kinds of ZeroMQ sockets. The client and the server must be configured to use matching socket types.

- REQ-REP: The client sends the raw ASCII bytes `next` (not using msgpack) as a request, and the reply is a message as described below. This is the default option.
- PUB-SUB: The client subscribes to messages published by the server. This is simpler, but can create a lot of network traffic.

Note: The data messages are documented here in terms of *msgpack*. The code can also use Python's *pickle* serialisation format, but since this is Python specific, it is not recommended for new code.

Message format 1.0

In the original message format (version `1.0`), data is sent in a single ZMQ message part containing a nested dictionary (a msgpack map).

The first level of keys are source names, which typically correspond to Karabo device names. Each source has a data dictionary representing a flattened Karabo hash, with dots delimiting successive key levels. Arrays are serialised using msgpack_numpy.

Each source data dictionary also has a key `metadata`, which contains a further nested dictionary with keys: `source`, `timestamp`, `timestamp.tid`, `timestamp.sec`, `timestamp.frac`, and `ignored_keys`, which is a list of strings identifying keys which were filtered out of the data by configuration options on the bridge server.

```
{
    'SPB_DET_AGIPD1M-1/DET/detector': {
        'image.data': array(...),
```

(continues on next page)

(continued from previous page)

```
        # ... other keys
        'metadata': {
            'source': 'SPB_DET_AGIPD1M-1/DET/detector',
            'timestamp': 1526464869.4109755,
            'timestamp.frac': '410975500000000000',
            'timestamp.sec': '1526464869',
            'timestamp.tid': 10000000001,
            'ignored_keys': []
        },
    }
}
```

Message format 2.2

We have skipped describing message formats 2.0 and 2.1, as we don't know of any code that used them before version 2.2 was defined.

The data is split up into a series of pieces, allowing arrays to be serialised more efficiently. Each piece has two ZeroMQ message parts: a msgpack-serialised header, followed by a data part. A full message is therefore a multipart ZeroMQ message with an even number of message parts.

Each header part is a dictionary (a msgpack map) containing at least the keys `source` and `content`. The former is a source name such as `'SPB_DET_AGIPD1M-1/DET/detector'`. The latter is one of:

- `'msgpack'`: The following data part is a msgpack map containing the data for this source, excluding any arrays. The header map also includes the `metadata` information as described for message format 1.0 above.
- `'array'`: The following data part is a raw array. The header has additional keys describing the array:
  - `path`: The key of this data, e.g. `'image.data'`.
  - `dtype`: A string naming a (numpy) dtype, such as `'uint16'` for 16-bit unsigned integers.
  - `shape`: An array of integers giving the dimensions of the array.

A multipart message might contain data from several sources. For each source, there is one header-data pair with `'msgpack'` content, followed by zero or more header-data pairs for arrays.

Changed in version 2.2: Moved metadata from the data to the header.

Image data

Karabo `ImageData` objects, holding images from cameras, are represented by a number of keys with a common prefix. This keys following this prefix include:

- `.data`: numpy array
- `.bitsPerPixels` int
- `.dimensions` list of int
- `.dimensionScales` str
- `.dimensionTypes` list of int
- `.encoding` str
- `.geometry.alignment.offsets` list of float

- `.geometry.alignment.rotations` list of float
- `.geometry.pixelRegion` list of int - seems optional
- `.geometry.subAssemblies` list of hashes - seems optional
- `.geometry.tileId` int
- `.header` user defined Hash
- `.ROIOffsets` list of int
- `.binning` list of int

Minor changes to this list may occur without a new protocol version.

Protocol implementations

Clients:

- Python client
- C++ client

Servers:

- PipeToZeroMQ Karabo device: sends data from a live Karabo system.
- extra_data Python module: can stream data from XFEL HDF5 files.
- The Python client includes a server to send simulated random data.

See also:

Streaming from Karabo bridge

6.1.3 HDF5 command line tools

We hope most users will be able to access data through existing tools such as *EXtra Data*, or by converting it to standard formats such as *CXI*. But if none of these options work for you, you may need to look directly at the HDF5 files.

Basic HDF5 command line tools are available by default:

```
h52gif         h5copy         h5fc-64        h5perf_serial  h5unjam
h5c++          h5debug        h5import       h5redeploy
h5c++-64       h5diff         h5jam          h5repack
h5cc           h5dump         h5ls           h5repart
h5cc-64        h5fc           h5mkgrp        h5stat
```

You can inspect HDF5 files in the terminal with the `h5glance` tool, available in the `exfel_anaconda3` module (see *EXtra-xwiz*).

The `hdfview` interactive viewer is available in the `xray` module (see https://confluence.desy.de/display/MXW/hdfview). To use it, first run `module load xray`.

6.2 Software with scientific purposes

6.2.1 EXtra-foam

One of our main programs for online analysis is *EXtra-foam*, which is a GUI program for easily doing common kinds of analysis.

6.2.2 GeoAssembler

This tool provides a tool to calibrate AGIPD detector geometry.The tool can be seen as an alternative to the calibration mode of CrysFEL's hdfsee. The calibration can either be based on a starting geometry that needs to be refined or a completely new geometry. In the latter case the initial conditions for the geometry are defined so that all modules are 29px apart from each other and 4px gap between asics within a module.

The geometry calibration is supported by two modes of graphical user interfaces. A Qt-based and a jupyter notebook based interface.

Using the Qt-Gui

It is **recommended** to use this Gui application through the pre-installed path on the on- and offline cluster (See *EXtra-xwiz*). The command is:

```
geoAssembler
```

The following optional arguments can be set via the command line:

`-h, --help`
: Show help about these options

`-nb, --notebook`
: Do not start gui, create a notebook

`-nb_dir`
: Set default directory to save notebooks

`-nb_file`
: Set file name of the notebook

`-r` `<run_dir>`, **`--run`** `<run_dir>`
: The path to a run folder

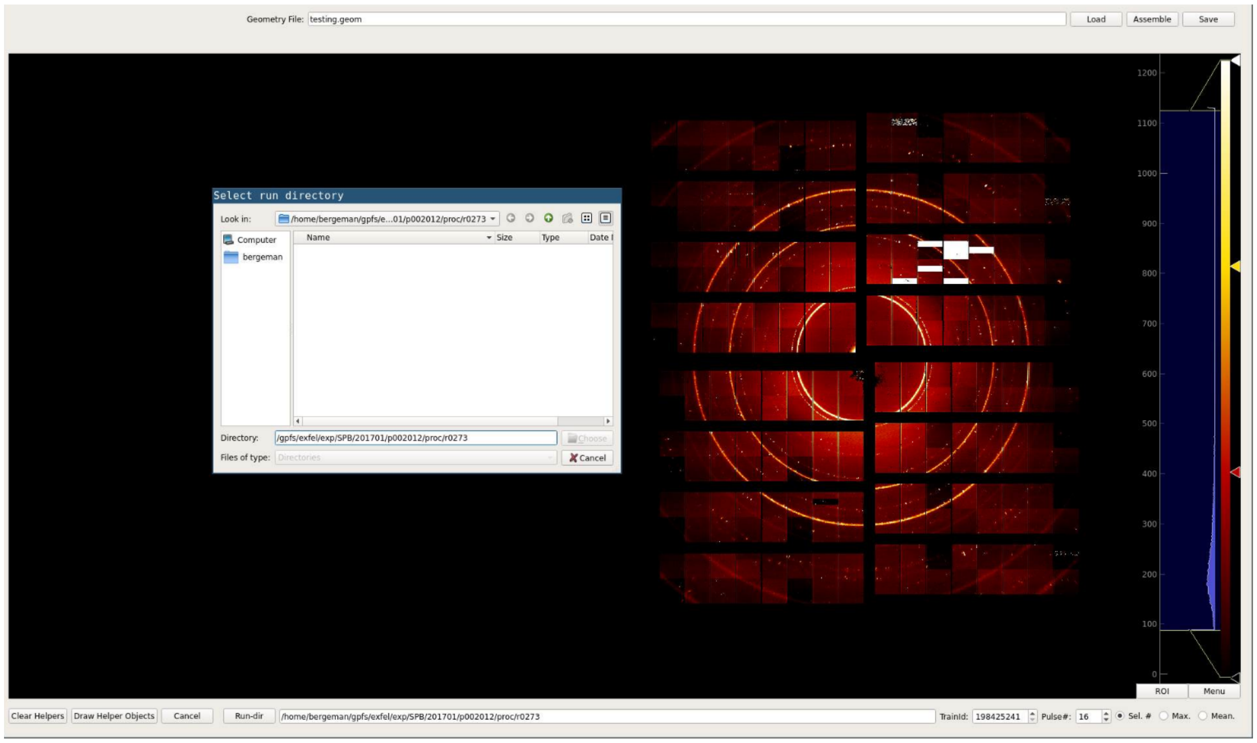
`-g` `<geomfile>`, **`--geometry`** `<geomfile>`
: Path to a CrystFEL format geometry file

`-c` `<clen>`, **`--clen`** `<clen>`
: Detector distance [m]

`-e` `<energy>`, **`--energy`** `<energy>`
: Photon energy [eV]

`-l` `<min> <max>`, **`--level`** `<min> <max>`
: Display range for plotting

If no run directory has been preselected using the `-r/--run` option, a directory can to be set by clicking the Run-dir button. Train IDs can be selected after a run has been selected. The user can either choose to display images by pulses or if the signal is to week/noisy by applying a Maximum or Mean across the entire train to all images. To do so the user can just select the *Max* or *Mean* button instead of the default *Sel #*. After an image number / function has been


Geometry File: testing.geom

Select run directory

Look in: /home/bergeman/gpfs/e...01/p002012/proc/r0273

Directory: /gpfs/exfel/exp/SPB/201701/p002012/proc/r0273

Files of type: Directories

/home/bergeman/gpfs/exfel/exp/SPB/201701/p002012/proc/r0273

TrainId: 198425241 Pulse#: 16

selected the image can be assembled using the *Assemble* button. Optionally a pre-defined geometry file can be loaded using the *Load* button.

After the image is displayed quadrants can be selected by clicking on them. They can be moved by using the *Ctrl+arrow-up/down/left/right* key combination. Circles that can help to align quadrants are added by the *Draw Helper Objects* button. The radii of the circles an be adjusted using the radius *spin box* in the top left.

Once the quadrants have been positioned a geometry file can be saved by using the *Save* button.

Calibration Using Jupyter The `-nb`, `--notebook` flag creates a Jupyter notebook in the home directory. This notebook is self explanatory.

Dependencies If the user doesn't want or cannot use the xfel module and wants to install the tool the following python packages should be available:

- `numpy`
- `cfelpyutils`
- `pyqtgraph`
- `matplotlib`
- `ipywidgets`
- `pyqt5`
- `pyFAI`

6.2.3 XasTim

A toolchain for real-time data analysis and visualization of XAS (X-ray Absorption Spectroscopy) experiments using the TIM (Transmission Intensity Monitor) device.

See *XasTim* in the SCS toolbox for more details.

6.2.4 Cheetah

The CFEL group at DESY provides Cheetah on the Maxwell cluster. See their documentation for how to get started using it.

See also:

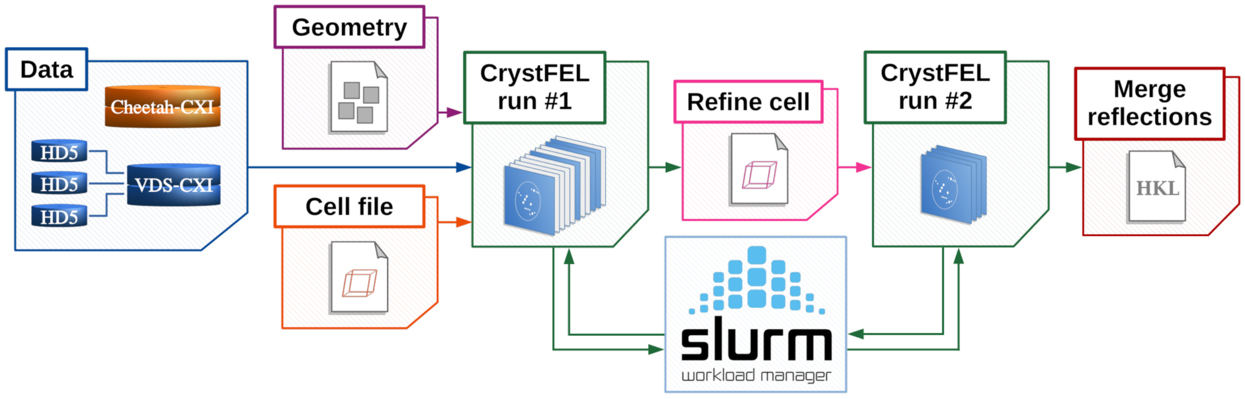
Cheetah documentation on European XFEL

6.2.5 EXtra-xwiz

A command-line tool for managing serial femtosecond crystallography (SFX) workflows using CrystFEL on EuXFEL-calibrated detector data. It is currently in a beta-testing phase.

Getting started with EXtra-xwiz

EXtra-xwiz is a command-line tool for automated processing of the calibrated serial crystalography data. It is a semi-automatic pipeline wrapped around CrystFEL and it utilises SLURM for parallel computing.



The basic functionality of the tool is explained here.

Installation

EXtra-xwiz is pre-installed in the EuXFEL software folder on Maxwell, currently in a virtual environment. To activate it, simply type

```
module load exfel EXtra-xwiz/beta
```

Getting and setting a configuration

The workflow configuration determines how the steps of a typical SFX workflow shall be run. The first call of the tool:

```
xwiz-workflow
```

will do nothing except writing a template configuration file to the current folder.

Luckily, there are only a few configuration items that have to be adapted by the user, because no reasonable default exists. Open the freshly created file `xwiz_conf.toml` with a text editor and change the right-hand parts of the definitions as required, e. g. proposal number, run and so on.

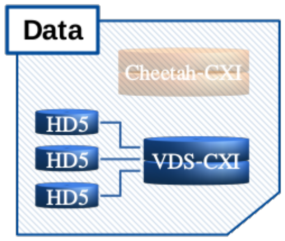
If configuration file with more parameters is desired `-adv / --advance-config` option can be used:

```
xwiz-workflow --advance-config
```

The configuration file explained

Generally the configuration is structured into groups, addressing different categories or aspects of the pipeline usage, such as input file retrieval, CrystFEL program parameters or SLURM (distributed computation) set-up.

A few specific aspects of the configuration values may not be obvious, let's look at the "data", "crystfel", "geom, and "slurm" groups.



```
[data]
proposal = 900145
runs = 752
n_frames_total = 10000
vds_names = ["jf_752_vds.cxi"]
list_prefix = "p900145_r0752"
```

proposal Proposal number which identifies your beam-time. Accessing calibrated/corrected detector data, Xwiz will expand this to the path of the proposal "proc" folder containing the runs you will work with.

runs Single data collection run or a list of run numbers that shall be processed within one Xwiz session. To add another run to the above example, change to e. g.: `[752,753]`

vds_names List of one or more virtual data set (VDS) file names. Any valid/existing path - absolute or relative - may be used. If the string(s) are only names, the path is interpreted as the current working directory (i. e. the folder from where Xwiz was started).

There are two ways to work with VDS files:

1. **create them from the original data in runs, typically at the first processing session.**
    - the **runs** parameter is essential
    - **vds_names** point to target file names of VDS files to be created
    - there must be one name for each run

   - e.g. in case of two runs: `["jf_752_vds.cxi", "jf_753_vds.cxi"]`

2. **use existing VDS files.**

   - The software first looks if the given files are already present
   - if so, the **runs** setting is ignored and the VDS will be re-used

n_frames_total The total number of frames to be processed, over all runs. Having `n_frames_total < actual sum of frames` is a way to truncate processing. The case `n_frames_total > actual sum of frames` will take the actual sum.

list_prefix All output files of Xwiz will start with this tag, it thus identifies the session.

CrystFEL

```
[crystfel]
# Available versions: '0.8.0', '0.9.1', '0.10.0', 'cfel_dev'
version = 'cfel_dev'
```

version Version number of the CrystFEL release to use. The need to switch to older releases may be given by the reproduction or comparison purposes. `cfel_dev` points to the latest builds, prior to the upcoming release.

Geometry

```
[geom]
file_path = "jungfrau8_p900145_v1_vds.geom"
```

file_path relative or absolute path to an existing CrystFEL-format geometry file appropriate for the detector geometry at the time of data collection.

Geometry files can come in two fashions, with either Cheetah-compatible or VDS-compatible pixel data references. The main difference concerns fused vs. stacked pixel data from multi-module detectors (AGIPD, JF4M). As per Cheetah layout, all pixels of an image-frame are in one set with just the dimensions (ss, fs), whereas in VDS layout there is an additional dimension containing the module index. If you have the correct geometry in terms of metrics, but the file refers to the wrong pixel layout, Xwiz will auto-correct these layout references to match the used data file. For instance, a geometry file stemming from usage with Cheetah will be auto-converted to VDS layout if Xwiz is run with a VDS file (default).

slurm
workload manager

```
[slurm]
# Available partitions: 'all', 'upex', 'exfel'
partition = "all"
duration_all = "2:00:00"
n_nodes_all = 10
duration_hits = "0:30:00"
n_nodes_hits = 4
```

partition points to the corresponding Maxwell partition of HPC nodes to be used. Default "all" would suite most needs. External users could also choose "upex" - or in case of dedicated beam-time reservations something like "upex_002697" (example).

duration_all (resp. **duration_hits**) The format reads "H:MM:SS" and defines for how long the HPC nodes shall be allocated.

In practice, processing that takes longer to finish than the allocated time will be aborted!

n_nodes_all (resp. **n_nodes_hits**) specifies how many HPC nodes to employ. Hard limits are set by the partition resp. reservation.

The `_all` vs. `_hits` distinction allows to have separate values for the processing of all frames in the first phase versus the re-processing of a subset of crystal "hit" frames (indexable due to Bragg diffraction patterns contained) in the second phase of processing.

Running the fully automatic mode

Provided that a good starting estimate for the crystal unit cell is available, one can create a CrystFEL unit cell file in the current folder, like this:

```
CrystFEL unit cell file version 1.0

lattice_type = tetragonal
centering = P
unique_axis = c
a = 79.1 A
b = 79.1 A
c = 37.9 A
al = 90.00 deg
be = 90.00 deg
ga = 90.00 deg
```

and specify the saved file name in the configuration.

To process the diffraction data, one can then enter

```
xwiz-workflow -a
```

and let everything run based on the configuration, ending up with the merged structure factor intensities and their crystallographic FOM vs. resolution- shell tables.

Running the interactive mode

Essentially, interactive mode means that all items of the configuration file (including advanced configuration parameters) are displayed for confirmation at a prompt at run-time. This is meant to allow for quick re-confirmation or modification of the workflow parameters, without the need of changing the configuration persistently by file editing.

The interactive mode is actually the default, meaning it is chosen when the tool is called without command-line argument. Hence, type

```
xwiz-workflow
```

and you will enter the interactive mode.

Output files

The following files can be found in the current folder, from where the tool was run, after the workflow has finished:

- a virtual data set for indexamajig (only in case that a non-existing file was referred to in the config)
- stream files from indexamajig
- a file <prefix>_hits.lst containing the numbers of detector frames that could be indexed (a sub-set of all frames)
- the unique set of structure factor intensities after scaling and averaging symmetry-equivalent observations with `partialator`, in three versions: fully merged as well as merged from the two complementary half-sets of equivalents
- text files *.dat with crystallographic FOMs (S/N, CC_1/2, CC*, R_split) in resolution shells
- a summary file written by the workflow tool, wrapping up indexing rates, unit cell refinement and FOMs

The summary file looks like this:

```
SUMMARY OF XWIZ WORKFLOW

Session time-stamp: 2020-07-22T12:18:48.711769
Operation mode:
  interactive (run-time parameter confirm/override)
Input type:
  virtual data set referring to EuXFEL-corrected HDF5

BASE CONFIGURATION USED
 Group: data
   path        : /gpfs/exfel/exp/XMPL/201750/p700000/proc/
   runs        : 29
   n_frames    : 630000

... <ECHO OF CONFIGURATION FILE> ...

Step #   d_lim   source      N(crystals)    N(frames)    Indexing rate [%%]
  1        3.5   indexamajig     28553      630000          4.53
                 cell_check      28553      630000          4.53
  2        2.0   indexamajig     27581       28553         96.60
                 OVERALL         27581      630000          4.38

Crystal unit cells used:

File                 Symmetry/axis, a, b, c, alpha, beta, gamma
```

(continues on next page)

(continued from previous page)

```
hewl.cell           tetragonal  P  c  79.1  79.1  37.9  90.00  90.00  90.00
hewl.cell_refined   tetragonal  P  c  79.7  79.7  38.0  90.00  90.00  90.00

Crystallographic FOMs:
                          overall    outer shell
Completeness              100.00         100.0
Signal-over-noise           5.84          2.54
CC_1/2                    0.6338        0.1394
CC*                       0.8363        0.4948
R_split                    33.68         72.59
```

6.3 Access on online and Maxwell cluster

The tools described above are readily available on both the online cluster and the Maxwell cluster. We recommend using the already setup applications available in the XFEL specific Anaconda3 distribution:

```
module load exfel exfel_anaconda3
```

This will provide access to all these libraries and tools for data analysis:

- extra_data
- EXtra-foam
- karabo_bridge
- extra_data_interactive
- geoAssembler
- pyFAI
- xas-tim-view
- karabo-bridge-record
- karabo-bridge-replay
- h5glance
- extra-data-validate

There are many other applications and libraries available on the Maxwell cluster, maintained by DESY. There is a listing at: https://confluence.desy.de/display/IS/Alphabetical+List+of+Packages

Not all of this software is on the online cluster (maintained by EuXFEL).

6.4 Adding extra software

Both the *Offline cluster* and the *Online cluster* environment feature a set of data analysis tools. If an experiment requires access to additional analysis packages or applications, this user requirement should be discussed and agreed ahead of the experiment; please contact `da-support@xfel.eu` in such a case.

In addition, users can bring their own tools and install them in their user space. This will make them available for immediate use in both offline and online environment.

6.5 Make your software available to all users at EuXFEL

In general, users who create their own data analysis packages are encouraged to share the progress on such data analysis development with European XFEL and seek help to make them functional at EuXFEL.

EuXFEL is keen to make such existing tools available to all users, and to provide the corresponding software as a service as part of a collaboration with the authors of the package. Please approach `da-support@xfel.eu` for further information and discussion if this is of interest to you.

CHAPTER

SEVEN

SPECIFIC USE CASES & DEPLOYMENTS

7.1 SCS toolbox

This section provides guidelines for data analysis softwares / toolchains which are deployed for the SCS instrument.

See also:

SCS Instrument documentation

SCS analysis toolbox

7.1.1 EXtra-foam (FastCCD)

Toolchain for real-time data analysis and visualization of FastCCD related data analysis. In this tool chain, FastCCDProcessor correlates and processes data from FastCCD and other data sources (XGM, digitizer, motors, etc.). The processed data is then sent out via the karabo bridge to *EXtra-foam*, which provides a rich interface for complicated data analysis and visualization.

Usage

1. Open the project *ZMQ_BRIDGE* within the **SCS** topic;

2. Instantiate two devices: *SCS_XAS_FASTCCD/DA/PROC* and *SCS_XAS_FASTCCD/DA/BRIDGE*;

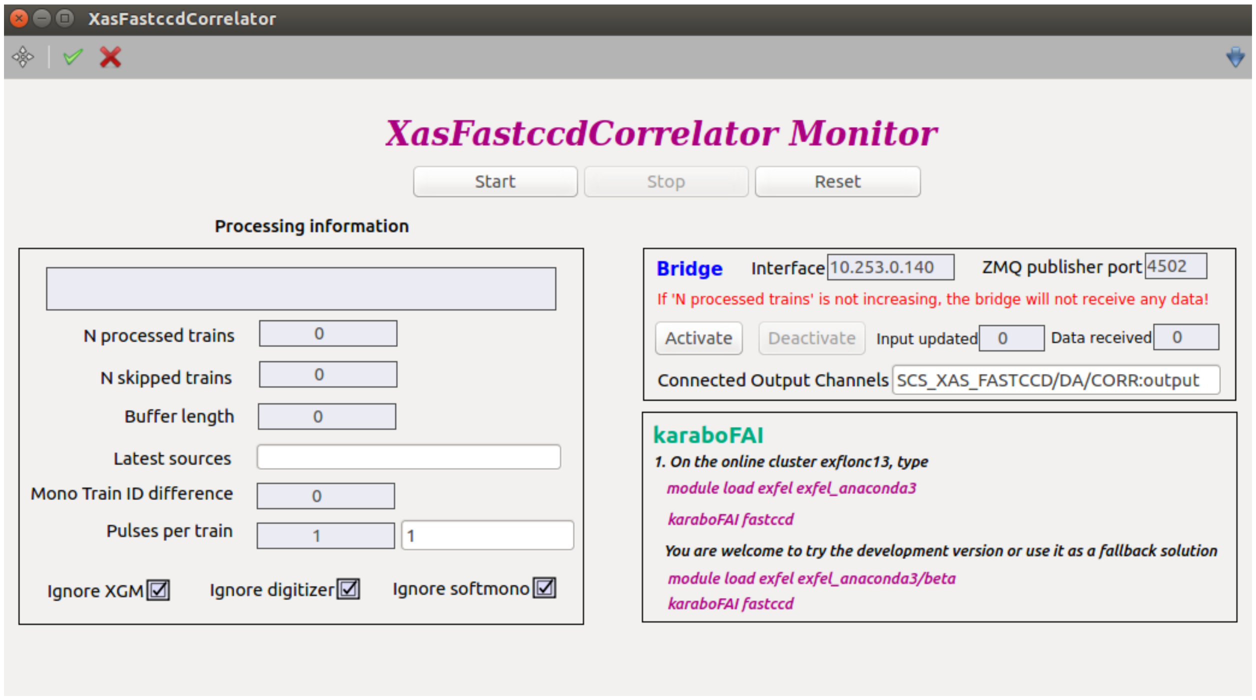
3. Open the following scene in Karabo:

Note: Please strictly follow the following steps since the data flows from upstream to downstream (processor -> bridge -> EXtra-foam). It also applies for troubleshooting!

4. Click the **Start** button in the scene. If *N processed trains* is increasing, it indicates that the processor is working. Otherwise, please check whether the other data source devices are running. To understand which data source is missing, one can check *Latest sources*;

Note: You may not need all the other data sources other than *FastCCD* in your experiment. Please feel free to "Ignore" the unwanted sources by ticking the checkbox in the scene.

5. Click the **Activate** button to activate the bridge. If *Input updated* and *Data received* are increasing, it indicates that the bridge is working. Otherwise, try re-instantiating the bridge device to re-establish the connection between the processor and the bridge;



N processed trains 0
N skipped trains 0
Buffer length 0
Latest sources
Mono Train ID difference 0
Pulses per train 1 1
Ignore XGM ☑ Ignore digitizer ☑ Ignore softmono ☑

Interface 10.253.0.140 ZMQ publisher port 4502
If 'N processed trains' is not increasing, the bridge will not receive any data!
Input updated 0 Data received 0
Connected Output Channels SCS_XAS_FASTCCD/DA/CORR:output

1. On the online cluster exflonc13, type
module load exfel exfel_anaconda3
karaboFAI fastccd
You are welcome to try the development version or use it as a fallback solution
module load exfel exfel_anaconda3/beta
karaboFAI fastccd

Note: If the bridge is still not working, please re-start the device server and repeat the step 2-5;

6. ssh to the online cluster **sa3-br-kc-comp-1** (**exflonc13**) and following the instructions about EXtra-foam in the Karabo scene.

7.1.2 XasTim

Toolchain for real-time data analysis and visualization of XAS (Xray absorption spectroscopy) experiments using TIM (transmission intensity monitor). In this tool chain, XasTimProcessor correlates and processes data from XGM, digitizer and softmono. The processed data is then sent out via the Karabo bridge to xas-tim-view, which visualizes the data in a web browser.

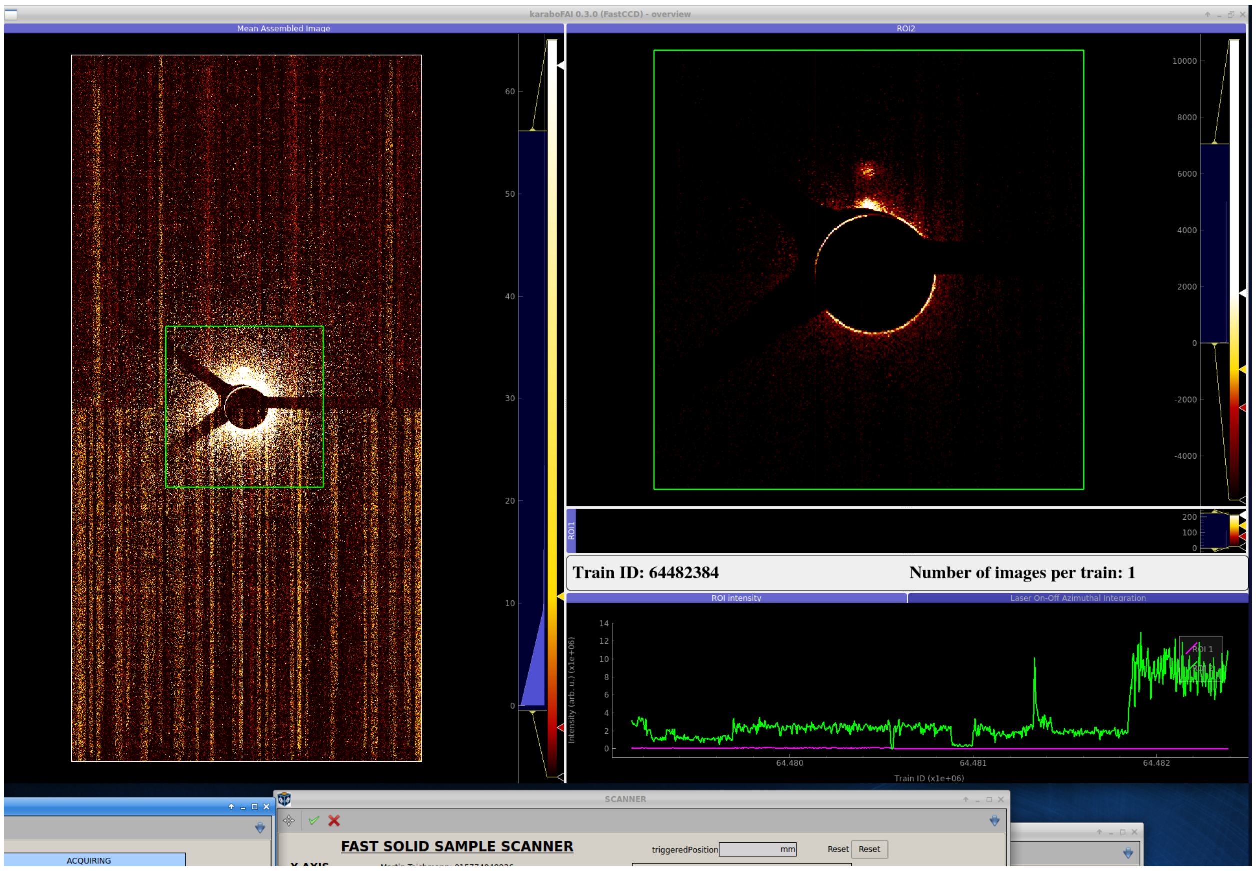
Usage

1. Open the project *ZMQ_BRIDGE* within the **SCS** topic;
2. Instantiate two devices: *SCS_XAS_TIM/DA/PROC* and *SCS_XAS_TIM/DA/BRIDGE*;
3. Open the following scene in Karabo;

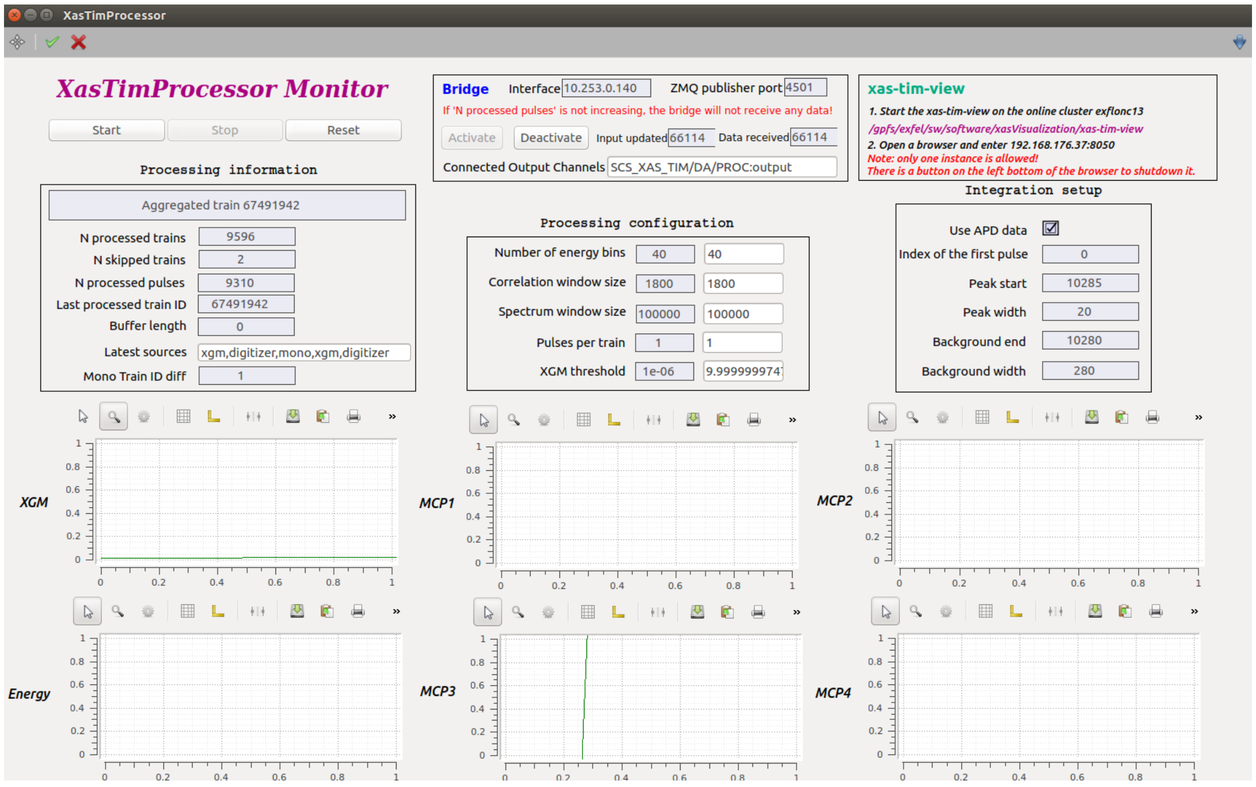
Note: Please strictly follow the following steps since the data flows from upstream to downstream (processor -> bridge -> xas-tim-view -> browser). It also applies for troubleshooting!

4. Click the **Start** button in the scene. If *N processed trains* is increasing, it indicates that the processor is working. Otherwise, please check whether the *XGM/digitizer/softmono* devices are running. To understand which data source is missing, one can check *Latest sources*;

5. Click the **Activate** button to activate the bridge. If *Input updated* and *Data received* are increasing, it indicates that the bridge is working. Otherwise, try re-instantiating the bridge device to re-establish the connection between the processor and the bridge;


karaboFAI 0.3.0 (FastCCD) - overview
Mean Assembled Image
ROI2
ROI1
ROI intensity
Laser On-Off Azimuthal Integration
Intensity (arb. u.) (x1e+06)
Train ID (x1e+06)
SCANNER
FAST SOLID SAMPLE SCANNER
triggeredPosition
Reset
ACQUIRING

Train ID: 64482384 Number of images per train: 1



XasTimProcessor Monitor

Processing information

Aggregated train 67491942

N processed trains	9596
N skipped trains	2
N processed pulses	9310
Last processed train ID	67491942
Buffer length	0
Latest sources	xgm,digitizer,mono,xgm,digitizer
Mono Train ID diff	1

Bridge Interface 10.253.0.140 ZMQ publisher port 4501

If 'N processed pulses' is not increasing, the bridge will not receive any data!

Input updated 66114 Data received 66114

Connected Output Channels SCS_XAS_TIM/DA/PROC:output

Processing configuration

Number of energy bins	40	40
Correlation window size	1800	1800
Spectrum window size	100000	100000
Pulses per train	1	1
XGM threshold	1e-06	9.999999974

xas-tim-view

1. Start the xas-tim-view on the online cluster exflonc13
/gpfs/exfel/sw/software/xasVisualization/xas-tim-view
2. Open a browser and enter 192.168.176.37:8050
Note: only one instance is allowed!
There is a button on the left bottom of the browser to shutdown it.

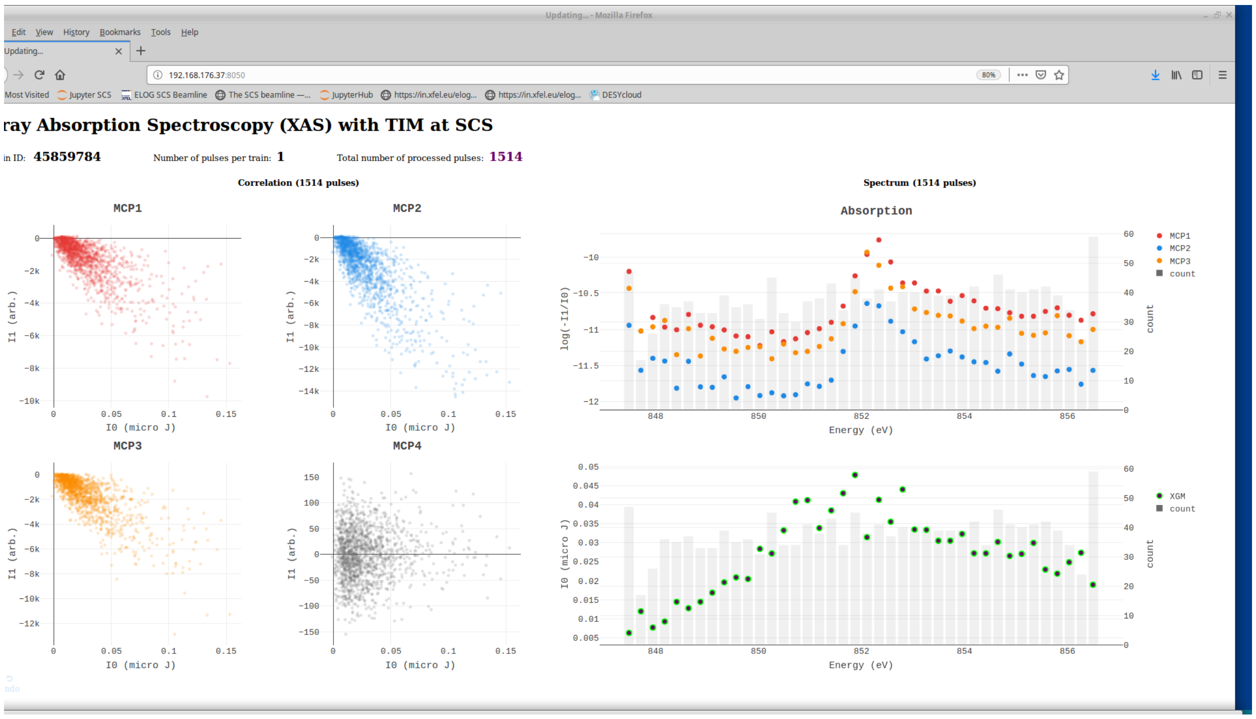
Integration setup

Use APD data	☑
Index of the first pulse	0
Peak start	10285
Peak width	20
Background end	10280
Background width	280

Note: If the bridge is still not working, please re-start the device server and repeat the step 2-5;

---

6. ssh to the online cluster **sa3-br-kc-comp-1** (**exflonc13**) and following the instructions about *xas-tim-view* in the Karabo scene.


Updating... - Mozilla Firefox
Edit View History Bookmarks Tools Help
192.168.176.37:8050
ray Absorption Spectroscopy (XAS) with TIM at SCS
in ID: 45859784
Number of pulses per train: 1
Total number of processed pulses: 1514
Correlation (1514 pulses)
Spectrum (1514 pulses)
MCP1
MCP2
MCP3
MCP4
I1 (arb.)
I0 (micro J)
Absorption
log(-I1/I0)
Energy (eV)
count
MCP1
MCP2
MCP3
count
I0 (micro J)
XGM
count

7.2 FXE toolbox

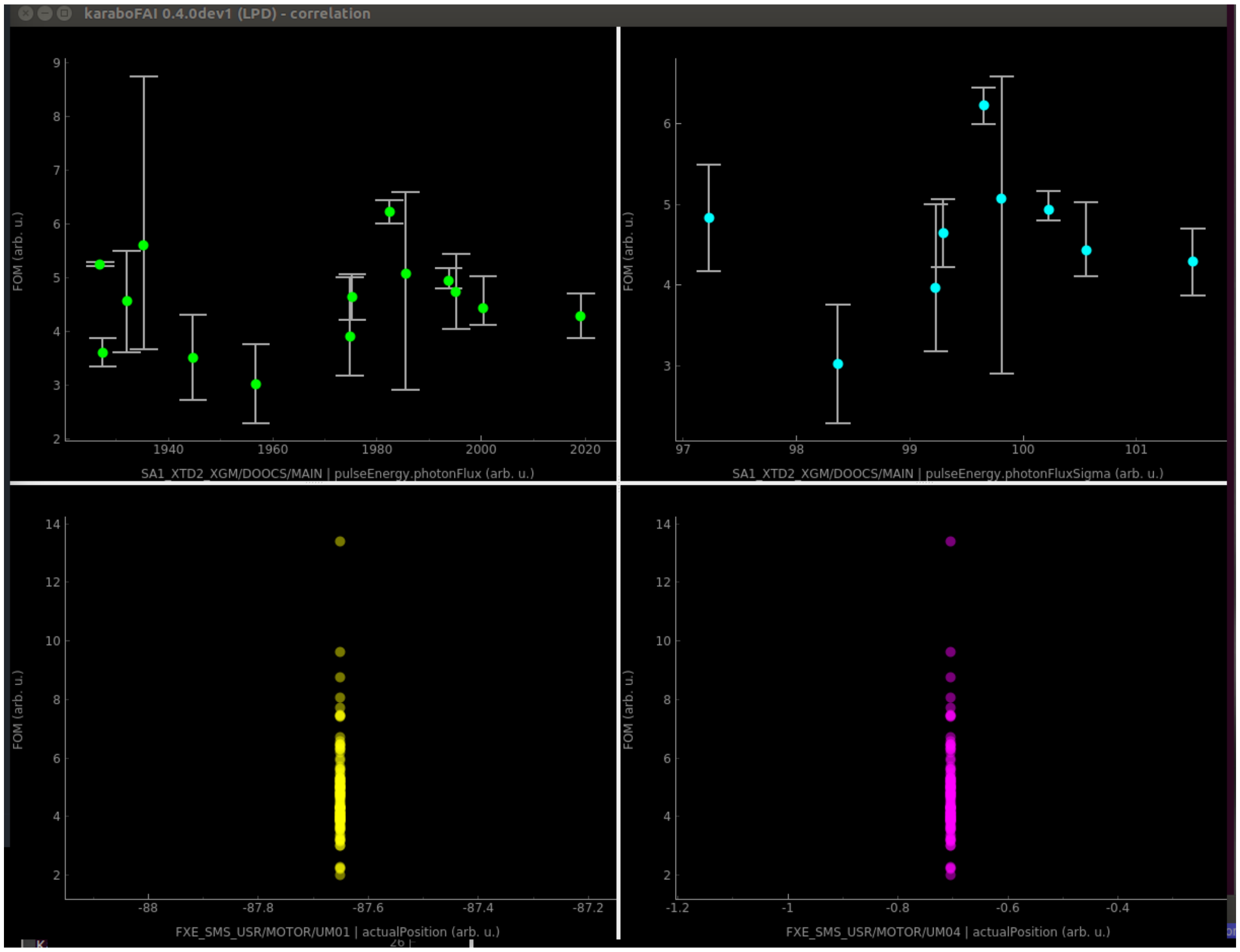
This section provides the workflow of data analysis specific tools at FXE instrument

7.2.1 Correlations in EXtra-foam

To optimize the experimental conditions, users need to observe the dependence of sample/jet position with respect to the Figure of Merit (FOM) evaluated from the diffraction signal in real time. EXtra-foam provides several ways of evaluating FOM from the images (fast data) obtained live during the experiment through **KaraboBridge**. As of now JUNGFRAU and LPD are the two detectors that are commonly used at FXE. **EXtra-foam** provides functionality to evaluate FOMs from data obtained from these two detectors.

To observe correlation between FOM and motor positions (any slow data) one needs to follow the workflow:

```
DataCorrelator (karabo device) -> KaraboBridge (karabo device) -> EXtra-foam (online␣
↪analysis tool)
```


karaboFAI 0.4.0dev1 (LPD) - correlation
FOM (arb. u.)
SA1_XTD2_XGM/DOOCS/MAIN | pulseEnergy.photonFlux (arb. u.)
SA1_XTD2_XGM/DOOCS/MAIN | pulseEnergy.photonFluxSigma (arb. u.)
FXE_SMS_USR/MOTOR/UM01 | actualPosition (arb. u.)
FXE_SMS_USR/MOTOR/UM04 | actualPosition (arb. u.)

7.3 HED toolbox

This section provides the workflow of data analysis specific tools at HED instrument.

7.3.1 Karabo bridge

Karabo bridge is available at the HED instrument and can be used to stream in realtime any data source available in Karabo during beamtime. Streamed data sources can be configured by instrument scientists upon request.

7.3.2 Dioptas

Dioptas <http://www.clemensprescher.com/programs/dioptas> is distributed in our software environment available on the *online cluster* and the *Maxwell cluster*:

```
module load exfel dioptas
```

CHAPTER EIGHT

OTHER RESOURCES

Access to a number of services is provided from https://in.xfel.eu, including the meta data catalog, and catering information.

8.1 UPEX portal

The User Portal to the European XFEL (UPEX) is the primary mechanism for communication with users, including submission of proposals, safety trainings and the provision of sample information ahead of experiments. In addition, a user's UPEX credentials (login name and password) provide access to the XFEL data analysis and computing resources as well as the users' experimental data.

It is accessible through the XFEL website: https://in.xfel.eu/upex

More information on how to use UPEX is available under:

https://www.xfel.eu/users/user_guide/upex_and_campus_accounts/index_eng.html

We advise future users to get knowledge about the *Scientific Data Policy of European XFEL*: https://www.xfel.eu/users/policies/index_eng.html

8.2 Desired support for experiment

Requirements for control of hardware and data analysis are initially recorded as part of the proposal application process. For accepted and scheduled proposals, subsequent planning will take place both in collaboration with the scientific instruments group and in collaboration with the control and analysis group to jointly identify and implement the most effective way of satisfying the users' requirements.

The first point of contact between the user group and the facility should always be the Local Contact (LC), a member of the scientific instrument group tasked with the coordination of an experiment from the facility side. An LC is assigned once a proposal is accepted and this information is provided to the users together with the letter of acceptance. The LC will direct questions and request from the users to the relevant groups at European XFEL. It is generally advisable to get in touch as soon as possible to advance the planning.

8.3 E-Log

Users can use the electronic notebook provided through https://in.xfel.eu/elog/. Some more details of are available at https://in.xfel.eu/elog/First+Help/

CHAPTER

NINE

PUBLICATIONS TO CITE

9.1 Data analysis support

For use of karabo pipelines, karabo bridge and extra-data/geom, and general data analysis support, please cite:

Short version:

- H. Fangohr et al., "Data Analysis Support in Karabo at European XFEL", in Proc. 16th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'17), Barcelona, Spain, Oct. 2017, paper TUCPA01, pp. 245-252, ISBN: 978-3-95450-193-9, doi:10.18429/JACoW-ICALEPCS2017-TUCPA01, 2018

Long version:

- H. Fangohr, M. Beg, V. Bondar, D. Boukhelef, S. Brockhauser, C. Danilevski, W. Ehsan, S.G. Esenov, G. Flucke, G. Giovanetti, D. Goeries, S. Hauf, B.C. Heisen, D.G. Hickin, D. Khakhulin, A. Klimovskaia, M. Kuster, P.M. Lang, L.G. Maia, L. Mekinda, T. Michelat, A. Parenti, G. Previtali, H. Santos, A. Silenzi, J. Sztuk-Dambietz, J. Szuba, M. Teichmann, K. Weger, J. Wiggins, K. Wrona, C. Xu (XFEL. EU, Schenefeld, Germany) and S. Aplin, A. Barty, M. Kuhn, V. Mariani (CFEL, Hamburg, Germany) and T. Kluyver (University of Southampton, Southampton, United Kingdom).

  "Data Analysis Support in Karabo at European XFEL", in Proc. 16th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'17), Barcelona, Spain, Oct. 2017, paper TUCPA01, pp. 245-252, ISBN: 978-3-95450-193-9, doi:10.18429/JACoW-ICALEPCS2017-TUCPA01, 2018

BibTeX to copy and paste:

```
@InProceedings{Fangohr:ICALEPCS2017-TUCPA01,
  author       = {H. Fangohr and others},
  title        = {{D}ata {A}nalysis {S}upport in {K}arabo at {E}uropean {XFEL}},
  booktitle    = {Proc. of International Conference on Accelerator and Large␣
↪Experimental Control Systems (ICALEPCS'17),
                  Barcelona, Spain, 8-13 October 2017},
  pages        = {245--252},
  paper        = {TUCPA01},
  language     = {english},
  keywords     = {ion, data-analysis, experiment, FEL, controls},
```

(continues on next page)

(continued from previous page)

```
  venue        = {Barcelona, Spain},
  series       = {International Conference on Accelerator and Large Experimental Control␣
↪Systems},
  number       = {16},
  publisher    = {JACoW},
  address      = {Geneva, Switzerland},
  month        = {Jan.},
  year         = {2018},
  isbn         = {978-3-95450-193-9},
  doi          = {https://doi.org/10.18429/JACoW-ICALEPCS2017-TUCPA01},
  url          = {http://jacow.org/icalepcs2017/papers/tucpa01.pdf},
  note         = {https://doi.org/10.18429/JACoW-ICALEPCS2017-TUCPA01},
}
```

Other formats available a plain LaTeX, Endnote and RIS.

The paper (pdf), slides (pdf), and different citations formats are available here.

9.2 Jupyter

For use of JupyterHub (for example through https://max-jhub.desy.de) and Jupyter Notebooks for data analysis at Eu-XFEL, please cite:

Short version:

- H. Fangohr et al., "Data Exploration and Analysis with Jupyter Notebooks", in Proc. ICALEPCS'19, New York, NY, USA, Oct. 2019, pp. 799-806. doi:10.18429/JACoW-ICALEPCS2019-TUCPR02, 2020.

Long version:

- H. Fangohr, M. Beg, M. Bergemann, V. Bondar, S. Brockhauser, C. Carinan, R. Costa, F. Dall'Antonia, C. Danilevski, J. C. E, W. Ehsan, S. G. Esenov, R. Fabbri, S. Fangohr, G. Flucke, C. Fortmann, D. Fulla Marsa, G. Giovanetti, D. Goeries, S. Hauf, D. G. Hickin, T. Jarosiewicz, E. Kamil, M. Karnevskiy, Y. Kirienko, A. Klimovskaia, T. A. Kluyver, M. Kuster, L. Le Guyader, A. Madsen, L. G. Maia, D. Mamchyk, L. Mercadier, T. Michelat, J. Möller, I. Mohacsi, A. Parenti, M. Reiser, R. Rosca, D. B. Rueck, T. Rüter, H. Santos, R. Schaffer, A. Scherz, M. Scholz, A. Silenzi, M. Spirzewski, J. Sztuk, J. Szuba, S. Trojanowski, K. Wrona, A. A. Yaroslavtsev, J. Zhu, J. Reppin, F. Schlünzen, M. Schuh, E. Fernandez-del-Castillo, G. Sipos, T. H. Rod, J. R. Selknaes, J. W. Taylor, A. Campbell, A. Götz, J. Kieffer, J. Hall, E. Pellegrini, J. F. Perrin.

  "Data exploration and analysis with Jupyter notebooks", in Proceedings of the 17th International Conference on Accelerator and Large Experimental Physics Control Systems ICALEPCS'19, New York, NY, USA, Oct. 2019, pp. 799-806. doi:10.18429/JACoW-ICALEPCS2019-TUCPR02, 2020.

BibTeX to copy and paste:

```
@InProceedings{fangohr:icalepcs2019-tucpr02,
  author       = {H. Fangohr and M. Beg and M. Bergemann and V. Bondar and S.␣
↪Brockhauser and A. Campbell and C. Carinan and R. Costa and F. Dall'Antonia and C.␣
↪Danilevski and J.C. E and W. Ehsan and S.G. Esenov and R. Fabbri and S. Fangohr and E.␣
↪Fernandez-del-Castillo and G. Flucke and C. Fortmann-Grote and D. Fulla Marsa and G.␣
↪Giovanetti and D. Goeries and A. Götz and J. Hall and S. Hauf and D.G. Hickin and T.␣
↪Holm Rod and T. Jarosiewicz and E. Kamil and M. Karnevskiy and J. Kieffer and Y.␣
↪Kirienko and A. Klimovskaia and T.A. Kluyver and M. Kuster and L. Le Guyader and A.␣
↪Madsen and L.G. Maia and D. Mamchyk and L. Mercadier and T. Michelat and I. Mohacsi␣
↪and J. Möller and A. Parenti and E. Pellegrini and J.F. Perrin and M. Reiser and J.␣
↪Reppin and R. Rosca and D.B. Rück and T. Rüter and H. Santos and R. Schaffer and A.␣
↪Scherz and F. Schlünzen and M. Scholz and M. Schuh and J.R. Selknaes and A. Silenzi␣
↪and G. Sipos and M. Spirzewski and J. Sztuk and J. Szuba and J.W. Taylor and S.␣
↪Trojanowski and K. Wrona and A.A. Yaroslavtsev and J. Zhu},
```

(continues on next page)

(continued from previous page)

```
% author       = {H. Fangohr and M. Beg and M. Bergemann and V. Bondar and S.␣
↪Brockhauser and A. Campbell and others},
% author       = {H. Fangohr and others},
  title        = {{Data Exploration and Analysis with Jupyter Notebooks}},
  booktitle    = {Proc. ICALEPCS'19},
  pages        = {799--806},
  paper        = {TUCPR02},
  language     = {english},
  keywords     = {FEL, data-analysis, experiment, software, detector},
  venue        = {New York, NY, USA},
  series       = {International Conference on Accelerator and Large Experimental Physics␣
↪Control Systems},
  number       = {17},
  publisher    = {JACoW Publishing, Geneva, Switzerland},
  month        = {08},
  year         = {2020},
  issn         = {2226-0358},
  isbn         = {978-3-95450-209-7},
  doi          = {10.18429/JACoW-ICALEPCS2019-TUCPR02},
  url          = {https://jacow.org/icalepcs2019/papers/tucpr02.pdf},
  note         = {https://doi.org/10.18429/JACoW-ICALEPCS2019-TUCPR02},
  abstract     = {Jupyter notebooks are executable documents that are displayed in a web␣
↪browser. The notebook elements consist of human-authored contextual elements and␣
↪computer code, and computer-generated output from executing the computer code. Such␣
↪outputs can include tables and plots. The notebook elements can be executed␣
↪interactively, and the whole notebook can be saved, re-loaded and re-executed, or␣
↪converted to read-only formats such as HTML, LaTeX and PDF. Exploiting these␣
↪characteristics, Jupyter notebooks can be used to improve the effectiveness of␣
↪computational and data exploration, documentation, communication, reproducibility and␣
↪re-usability of scientific research results. They also serve as building blocks of␣
↪remote data access and analysis as is required for facilities hosting large data sets␣
↪and initiatives such as the European Open Science Cloud (EOSC). In this contribution␣
↪we report from our experience of using Jupyter notebooks for data analysis at research␣
↪facilities, and outline opportunities and future plans.},
}
```

The paper (pdf) different citations formats are available here.

9.3 Detectors and calibration

For use of detectors and calibration pipeline, please cite

- M. Kuster, D. Boukhelef, M. Donato, J.-S. Dambietz, S. Hauf, L. Maia, N. Raab, J. Szuba, M. Turcato, K. Wrona & C. Youngman (2014) Detectors and Calibration Concept for the European XFEL, Synchrotron Radiation News, 27:4, 35-38, DOI: 10.1080/08940886.2014.930809

Online: https://www.tandfonline.com/doi/abs/10.1080/08940886.2014.930809.

As BibTeX:

```
@article{doi:10.1080/08940886.2014.930809,
  author = {M. Kuster and D. Boukhelef and M. Donato and J.-S. Dambietz and S. Hauf and␣
↪L. Maia and N. Raab and J. Szuba and M. Turcato and K. Wrona and C. Youngman},
```

(continues on next page)

(continued from previous page)

```
  title = {Detectors and Calibration Concept for the European XFEL},
  journal = {Synchrotron Radiation News},
  volume = {27},
  number = {4},
  pages = {35-38},
  year  = {2014},
  publisher = {Taylor & Francis},
  doi = {10.1080/08940886.2014.930809},
  URL = {https://doi.org/10.1080/08940886.2014.930809},
  eprint = {https://doi.org/10.1080/08940886.2014.930809}
}
```

CHAPTER

TEN

CHANGES TO THIS DOCUMENT

This documentation is under active development and may change anytime to include corrections and additions. Please get in touch with requests for clarifications/corrections. If you want to refer to a particular version, please use date and time as shown here in the next line.

Last compiled 2023-01-27, 10:06 (UTC).

- 1 Nov 2017: add comment on home directories
- 17 Nov 2017: add link to ELOG help
- 17 Nov 2017: same path on online and offline cluster for data files
- 17 Nov 2017: remove short version of file path (will disappear)
- 27 Nov 2017: mention `bastion.desy.de` for ssh access & clarifications
- 15 Jan 2018: adding FAQs, how to connect Jupyter Notebook to Maxwell cluster
- 17 July 2018: adding candidate frame list section
- 27 Aug 2018: Add numbering of sections
- ?
- 12 Jun 2019: update support email address (da-support)

BIBLIOGRAPHY

[Fangohr2017] Fangohr, Hans, et al. “Data Analysis Support in Karabo at European XFEL” ICALEPCS 2017. Available online: http://accelconf.web.cern.ch/AccelConf/icalepcs2017/doi/JACoW-ICALEPCS2017-TUCPA01.html

[Hauf2019] Hauf, Steffen, et al. “The Karabo distributed control system” J. Synchrotron Rad. 26 (2019): 1448-1461. Available online: https://doi.org/10.1107/S1600577519006696

[KusterCal2014] Kuster, Markus, et al. “Detectors and calibration concept for the European XFEL.” Synchrotron radiation news 27.4 (2014): 35-38. Available online: https://www.tandfonline.com/doi/abs/10.1080/08940886.2014.930809.

[Maltezopoulos2019] Maltezopoulos, Theophilos et al. “Operation of X-ray gas monitors at the European XFEL” J. Synchrotron Rad. 26 (2019): 1045-1051. Available online: https://doi.org/10.1107/S1600577519003795

PYTHON MODULE INDEX

INDEX