Detector Support Guidelines Documentation

Release 0.1

Detector Group

Mar 02, 2021

Contents

Contents:

CHAPTER 1

Logging onto exflgateway

Most detector control and online calibration will require you to work from a host within the control network. This procedure describes how to log into exflgateway. From there you can access control network hosts if you have the correct privileges.

Note: Your account has to be registered to access the control network. You can do this by filling out this form, and obtaining the relevant signatures

1. if not using a DESY VPN connection, log on to bastion.desy.de if you need X forwarding:

   ```
   ssh -X username@bastion.desy.de
   ```

   For using port forwarding with a Jupyter notebook

   ```
   ssh -L XXXX:localhost:XXXX username@bastion.desy.de
   ```

   where you replace *XXXX* with a port number, e.g. 7999.

Note: Use *ssh -Y* on MacOs computers.

2. log on to *exflgateway*; you can do this directly when using a VPN connection:

   ```
   ssh -X exflgateway
   ```

   or for port forwarding

   ```
   ssh -L XXXX:localhost:XXXX exflgateway
   ```

   be sure to use the same port as before.

CHAPTER 2

Initializing a Karabo Environment for an Instrument

To initialize a Karabo environment for a specific instrument please use the instructions for one of the following scenarios:

2.1 In the Hutch/Control Room

The control network client PCs should already point to the correct Karabo environment, so usually

```
karabo-gui
```

should suffice. If the *karabo-gui* command is not found, try

```
source ~/karabo/activate.
```

If this does not work either, please ask an instrument scientist or call CAS OCD.

2.2 On exflgateway - GUI only

Log onto exflgateway (*Logging onto exflgateway*). Then activate the most up-to-date Karabo environment and start the GUI:

```
source /opt/karabo/karabo/activate
karabo-gui
```

In the login dialog,

connect to the GUI server corresponding to your instrument:

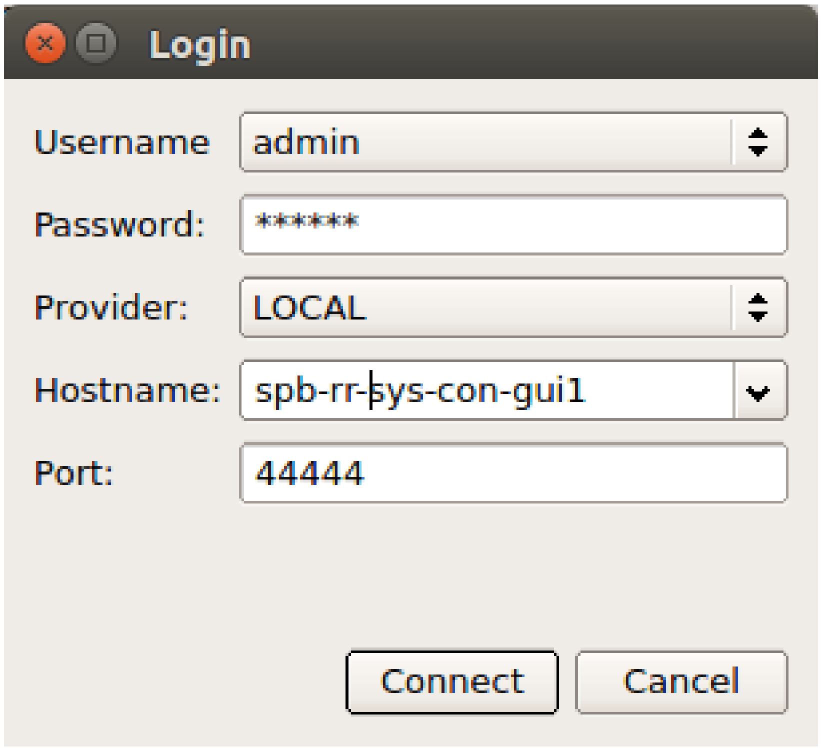

Username: admin
Password: ******
Provider: LOCAL
Hostname: spb-rr-sys-con-gui1
Port: 44444

Fig. 2.1: The Karabo logon dialog.

Instrument	Hostname
SPB	spb-rr-sys-con-gui1
FXE	fxe-rr-sys-con-gui1
SQS	sqs-rr-sys-con-gui1
SCS	scs-rr-sys-con-gui1
MID	mid-rr-sys-con-gui1
HED	hed-rr-sys-con-gui1

2.3 On exflgateway - iKarabo

Log onto exflgateway (*Logging onto exflgateway*). Then activate the most up-to-date Karabo environment and start the GUI:

```
source /opt/karabo/karabo/activate
```

You will now need to set-up the broker information for the respective instrument:

```
export KARABO_BROKER_TOPIC=<INSTRUMENT>
export export KARABO_BROKER=tcp://<SASE>-br-sys-broker-1:7777
```

where you should fill in <*INSTRUMENT*> and <*SASE*>:

Instrument	<INSTRUMENT>	<SASE>
SPB	SPB	sa1
FXE	FXE	sa1
MID	MID	sa2
HED	HED	sa2
SCS	SCS	sa3
SQS	SQS	sa3

Now you can run *iKarabo*:

```
ikarabo
```

Verify correct settings, but obtaining a list of devices for the instrument

```
getDevices()
```

CHAPTER 3

Restarting Online Calibration

3.1 Preface

Restarting the online calibration devices should happen as part of normal shift start-up and during trouble shooting. Please read the following sections first, in case you are not sure if a restart is actually needed:

General Detector DAQ Troubleshooting

3.2 Before Restart

Since the restart process involves many device which act on input in an event driven (push) fashion, it is best to assure that no new data is input into the pipeline during the restart process. Either:

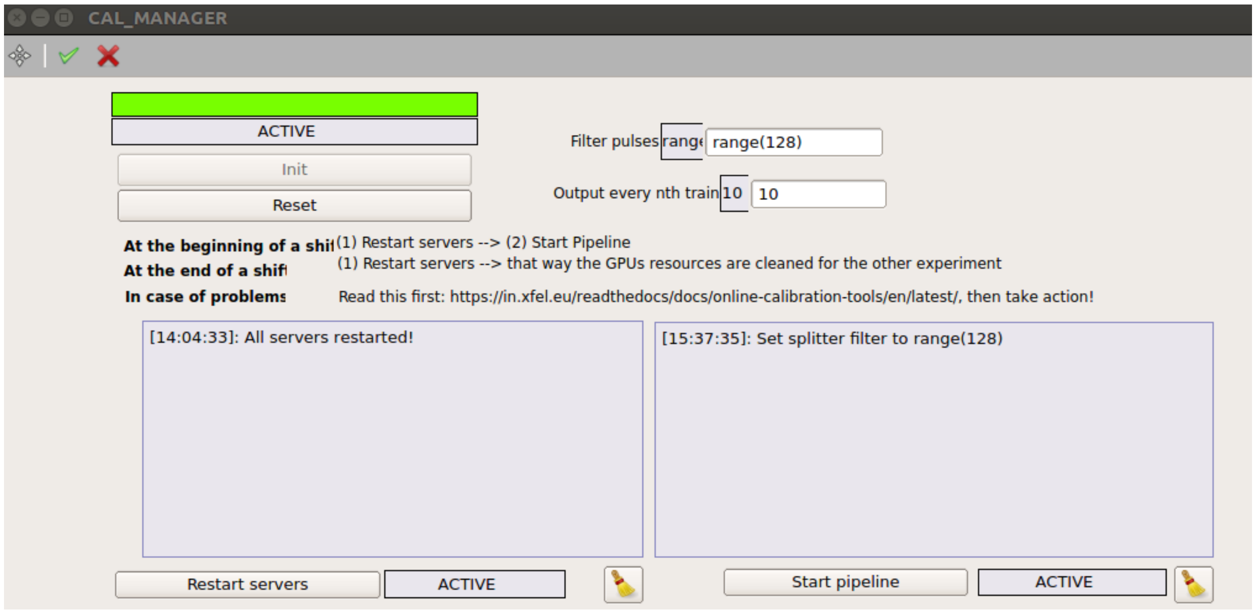
- stop the detector from sending data. DAQ can stay in monitoring mode
- if you do not want to stop the detector, switch the DAQ to ignore mode

3.3 Restart via GUI

Follow this procedure in case you are on-site or otherwise have stable and fast access to a Karabo-GUI of the respective instrument.

1. Follow the instructions in *Initializing a Karabo Environment for an Instrument* to open a Karabo-GUI, then load the master project of the respective detector. The will be a sub-project for calibration, which contains a *MANAGER* scene.
2. In this scene click on the *Restart servers* button, as shown in the screenshot. This will restart all servers involved in online calibration. It will take about 1 minute and will output the message “All servers restarted!” in the log window right above the button.

3. After restarting all servers, click *Start pipeline* to actually start the pipeline again. This might take a few minutes, especially until all constants have been loaded. The state color field will turn to green and the general state will turn to active when it is done, as shown in the screenshot.



Filter pulses range(128)

Output every nth train 10

At the beginning of a shift (1) Restart servers --> (2) Start Pipeline

At the end of a shift (1) Restart servers --> that way the GPUs resources are cleaned for the other experiment

In case of problems Read this first: https://in.xfel.eu/readthedocs/docs/online-calibration-tools/en/latest/, then take action!

```
[14:04:33]: All servers restarted!
```

```
[15:37:35]: Set splitter filter to range(128)
```

Fig. 3.1: Restarting servers and re-initializing calibration pipeline for MHz detectors.

3.4 Remote restart via command line

Follow this procedure in case you are giving remote support and do not have stable and fast access to a Karabo-GUI.

1. The restart needs to happen with the control network, please follow the instructions given at *Logging onto exflgateway*
2. You will also need to set-up a Karabo environment for the respective interface: *Initializing a Karabo Environment for an Instrument*
3. Open an *iKarabo* console:

```
ikarabo
```

4. Get the restart device and restart the device servers:

```
d = getDevice("<DEVICE_ID>")
d.restart()
waitUntil(lambda: d.state == State.ACTIVE)
```

where *<DEVICE_ID>* is the following for the different instruments:

Instrument	<DEVICE_ID>
SPB	SPB_DET_AGIPD1M-1/CAL/RESTART
MID	MID_DET_AGIPD1M-1/CAL/RESTART
FXE	FXE_DET_LPD1M-1/CAL/RESTART
SCS	SCS_DET_DSSC1M-1/CAL/RESTART

Note: The SQS and HED instrument currently do not use MHz detectors and thus all calibration devices are started from the Karabo project.

The last command will only return after the pipeline is initialized again.

5. Now re-init the calibration pipeline

```
d = getDevice("<DEVICE_ID>")
d.start_pipeline()
waitUntil(lambda: d.state == State.ACTIVE)
```

where *<DEVICE_ID>* is the following for the different instruments:

Instrument	<DEVICE_ID>
SPB	SPB_DET_AGIPD1M-1/CAL/INIT
MID	MID_DET_AGIPD1M-1/CAL/INIT
FXE	FXE_DET_LPD1M-1/CAL/INIT
SCS	SCS_DET_DSSC1M-1/CAL/INIT

The last command will only return after the pipeline is initialized again.

Note: If for any reasons one or the other of the above needed devices are not running, ask the instrument to instantiate them via the Karabo-GUI. They are in the online calibration subproject. You can also call CAS OCD.

3.5 After restart - push initial data

Regardless of how you restarted the pipeline, after the restart you should first leave, or instruct the instrument operator to leave the filter pulse expression and the output every nth trains at the default value. They should then have the detector push data.

The pipeline is fully initialized if all components show green state colors on the *MANAGER* scene. This may take up to about 30 seconds - the detector can continuously produce data during this time.

After all indicators are green, the filter expressions may be set to lower values.

3.6 Restarting only parts of the pipeline

If only individual components of the pipeline have crashed or are otherwise disfunctional, you can attempt to restart only these, instead of restarting everything. This might be considerably faster than a full restart.

0. Stop the detector from producing data
1. Shutdown those devices which need a restart in case they are still alive
2. Make sure the pipeline is in an idle state: either click *Reset* in the *MANAGER* scene, or on the command line, find the calibration manager device and then use its reset method:

```
d = getDevice("<DEVICE_ID>")
d.resedt()
waitUntil(lambda: d.state == State.PASSIVE)
```

where *<DEVICE_ID>* is the following for the different instruments:

Instrument	<DEVICE_ID>
SPB	SPB_DET_AGIPD1M-1/CAL/CAL_MANAGER
MID	MID_DET_AGIPD1M-1/CAL/CAL_MANAGER
FXE	FXE_DET_LPD1M-1/CAL/CAL_MANAGER
SCS	SCS_DET_DSSC1M-1/CAL/CAL_MANAGER

3. Now start the pipeline using either the *Start pipeline* button on the *MANAGER* scene, or via the command line

```
d = getDevice("<DEVICE_ID>")
d.start_pipeline()
waitUntil(lambda: d.state == State.ACTIVE)
```

As with a full restart the overall state should turn to *ACTIVE* once the restart is done.

CHAPTER 4

Restart Offline Calibration

In the following the procedures for verifying operation and restarting of offline calibration are described

4.1 Preface

The offline calibration runs on *max-exfl016* under the *xcal* user account. You need to be able to *kinit* yourself into that account in case you want to perform any of the below procedures. Please ask ITDM for access.

You can login direcly from within the DESY network.

```
kinit <username>
ssh xcal@max-exfl016
```

Remotely, when not on a VPN you need to go via Bastion:

```
ssh <username>@bastion.desy.de
ssh xcal@max-exfl016
```

Warning: You are working on the production offline calibration service. Please proceed with care

4.2 Checking Status

The following status checks should be performed to verify if the service is working correctly:

4.2.1 Is the Service Running

The service is run as a Python webservice. To check if it is running type:

```
ps aux | grep webservice
```

This should result in output similar to

```
xcal     15440  0.0  0.0 401188 36464 ?        Sl   Jun10   0:20 /home/xcal/
↪calibration_webservice_venv/bin/python /home/xcal/calibration_webservice/webservice/
↪webservice.py --mode prod --logging INFO --config-file /home/xcal/calibration_
↪webservice/webservice/webservice.yaml
xcal     17569  0.0  0.0 112812  1020 pts/1    S+   13:49   0:00 grep --color=auto␣
↪webservice
xcal     21500  0.0  0.0 222752 87788 ?        S    Jun02   2:23 /home/xcal/
↪calibration_webservice_venv/bin/python /home/xcal/calibration_webservice/webservice/
↪serve_overview.py --config /home/xcal/calibration_webservice/webservice/serve_
↪overview.yaml
```

It is important that the first line is present.

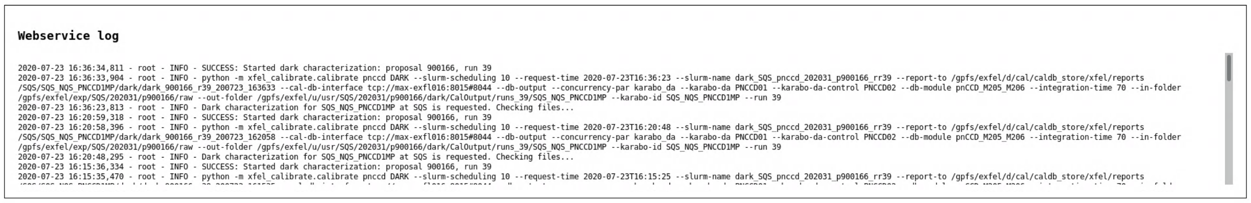
As the offline calibration webservice is running using a virtual environment, you will need to activate this environment before you go further.

The path for the bin folder for the python environment should be shown as the above output. to activate this python environment write:

source /home/xcal/calibration_webservice_venv/bin/activate

4.2.2 Checking the Logs

The logs are located the *max-exfl016:8008* web page:



Webservice log

```
2020-07-23 16:36:34,811 - root - INFO - SUCCESS: Started dark characterization: proposal 900166, run 39
2020-07-23 16:36:33,904 - root - INFO - python -m xfel_calibrate.calibrate pnccd DARK --slurm-scheduling 10 --request-time 2020-07-23T16:36:23 --slurm-name dark_SQS_pnccd_202031_p900166_rr39 --report-to /gpfs/exfel/d/cal/caldb_store/xfel/reports
/SQS/SQS_NQS_PNCCD1MP/dark/dark_900166_r39_200723_163633 --cal-db-interface tcp://max-exfl016:8015#8044 --db-output --concurrency-par karabo_da --karabo-da PNCCD01 --karabo-da-control PNCCD02 --db-module pnCCD_M205_M206 --integration-time 70 --in-folder
/gpfs/exfel/exp/SQS/202031/p900166/raw --out-folder /gpfs/exfel/u/usr/SQS/202031/p900166/dark/CalOutput/runs_39/SQS_NQS_PNCCD1MP --karabo-id SQS_NQS_PNCCD1MP --run 39
2020-07-23 16:36:23,813 - root - INFO - Dark characterization for SQS_NQS_PNCCD1MP at SQS is requested. Checking files...
2020-07-23 16:20:59,318 - root - INFO - SUCCESS: Started dark characterization: proposal 900166, run 39
2020-07-23 16:20:58,396 - root - INFO - python -m xfel_calibrate.calibrate pnccd DARK --slurm-scheduling 10 --request-time 2020-07-23T16:20:48 --slurm-name dark_SQS_pnccd_202031_p900166_rr39 --report-to /gpfs/exfel/d/cal/caldb_store/xfel/reports
/SQS/SQS_NQS_PNCCD1MP/dark/dark_900166_r39_200723_162058 --cal-db-interface tcp://max-exfl016:8015#8044 --db-output --concurrency-par karabo_da --karabo-da PNCCD01 --karabo-da-control PNCCD02 --db-module pnCCD_M205_M206 --integration-time 70 --in-folder
/gpfs/exfel/exp/SQS/202031/p900166/raw --out-folder /gpfs/exfel/u/usr/SQS/202031/p900166/dark/CalOutput/runs_39/SQS_NQS_PNCCD1MP --karabo-id SQS_NQS_PNCCD1MP --run 39
2020-07-23 16:20:48,295 - root - INFO - Dark characterization for SQS_NQS_PNCCD1MP at SQS is requested. Checking files...
2020-07-23 16:15:36,334 - root - INFO - SUCCESS: Started dark characterization: proposal 900166, run 39
2020-07-23 16:15:35,470 - root - INFO - python -m xfel_calibrate.calibrate pnccd DARK --slurm-scheduling 10 --request-time 2020-07-23T16:15:25 --slurm-name dark_SQS_pnccd_202031_p900166_rr39 --report-to /gpfs/exfel/d/cal/caldb_store/xfel/reports
```

or they can be accessed in the webservice run subdirectory:

```
tail -500f calibration_webservice_run/web.log
```

You should see entries similar to:

```
2019-07-04 12:20:48,988 - root - INFO - Copying /gpfs/exfel/exp/SQS...
2019-07-04 12:20:48,996 - root - INFO - Copying /gpfs/exfel/exp/SQS...
2019-07-04 12:20:49,004 - root - INFO - Copying /gpfs/exfel/exp/SQS...
2019-07-04 12:20:49,013 - root - INFO - Copying /gpfs/exfel/exp/SQS...
2019-07-09 16:11:29,966 - root - INFO - python -m xfel_calibrate.calibrate JUNGFRAU␣
↪CORRECT ...
2019-07-09 16:11:36,987 - root - INFO - SUCCESS: Started correction: proposal 900063,␣
↪run 1051
```

These will tell you the last processes that were started. Especially useful are the lines which start with *python -m xfel_calibrate.calibrate* ... as these tell you which calibration jobs were launched and with which parameters.

4.2.3 Checking for running Jobs

The following command will display a list of currently running jobs on Maxwell:

```
squeue  -u xcal -o "%.18i %60j %.3t %.10M %R"
```

You should check that jobs are not piling up, e.g. have been queued for a long time. If this is the case, the instrument should be contacted.

4.3 Restarting the Service

Restarting the calibration webservice should rarely be necessary, and if so, should only be done with great care as it can affect the entire facility. If you are unsure, first consult with other colleagues.

1. Check if the service process is still running and kill it:

   ```
   ps aux | grep webservice.py
   ```

   Note down the PID and kill that process if it is running:

   ```
   kill -9 <PID>
   ```

2. Restart the service:

   ```
   cd calibration_webservice_run
   nohup /home/xcal/calibration_webservice_venv/bin/python /home/xcal/calibration_
   ↪webservice/webservice/webservice.py --mode prod --logging INFO --config-file /
   ↪home/xcal/calibration_webservice/webservice/webservice.yaml > nohup_webservice.
   ↪out&
   ```

3. Verify it is running by checking for the process:

   ```
   ps aux | grep webservice.py
   ```

   and the logs:

   ```
   tail -500f /home/xcal/calibration_webservice_run/web.log
   ```

4. Notify the instruments that the service was down:

   ```
   send-to: <all instrument emails>
   subject: Restart of Calibration Webservice

   The calibration webservice required a restart due to....

   The restart was performed in the time from XXX to YYY. During this time some jobs␣
   ↪submitted through the
   MDC or calibrate_dark script may not have been handled. If you submitted jobs␣
   ↪during this time, please consider
   resubmitting them. If you have questions, please contact det-support@xfel.eu.␣
   ↪Please also forward this information
   to your users.
   ```

4.4 Restarting the overview webpage

Previous first step for restarting the calibration webservice should be performed then

2. Restart the overview page:

```
cd calibration_webservice_run
nohup /home/xcal/calibration_webservice_venv/bin/python /home/xcal/calibration_
↪webservice/webservice/serve_overview.py --config /home/xcal/calibration_
↪webservice/webservice/serve_overview.yaml > nohup_serve_overview.out &
```

3. Verify it is running by checking for the process:

```
ps aux | grep webservice.py
```

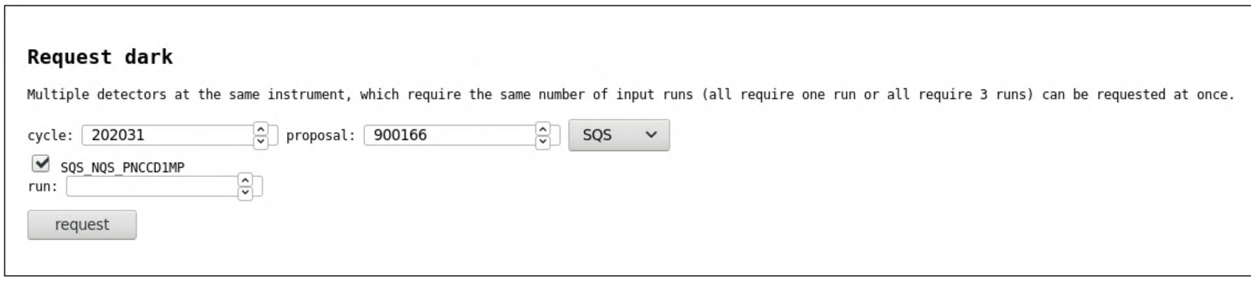
and the webpage through max-exfl16:8008 from a browser accessing the maxwell network.

CHAPTER 5

Requesting Dark Characterization

Dark characterization is currently done via *max-exfl016:8008*:

1. Record the dark images as appropriate for the detector in use. This will result in either one or three relevant runs.
2. Migrate these runs to Maxwell using the metadata catalogue.
3. Open a firefox browser from xfel network and access *max-exfl016:8008*
4. Go to *Request dark* section and enter the *CYCLE* and *PROPOSAL* numbers, then select the *INSTRUMENT*. Select the needed detector from the choices appearing and enter the dark runs numbers.



Request dark

Multiple detectors at the same instrument, which require the same number of input runs (all require one run or all require 3 runs) can be requested at once.

cycle: 202031 proposal: 900166 SQS

✓ SQS_NQS_PNCCD1MP

run:

CHAPTER 6

Connecting a Single Module's Raw Data to the Bridge

Follow the procedure below to connect a ZMQ bridge directly to a single modules data stream in extraordinary case and in the **only supported** way.

0. Talk to the users and instrument. Question if they *really* needs this option. Then question again if they *really really* need it and if they know how to handle raw data from the detector. Tell them that for any calibration they will be on their own, and that it impacts online views of the instrument.
1. Stop the detector from producing data
2. Remove the module for the mandatory modules list, found at the bottom left corners of the *CORRECTED* and *RAW* overview scenes, in case it is configured there.
3. Find the corresponding *SPLITTER* device by searching the topology for the module location, e.g. *FEM_SPLITTER_Q1M1*.
4. On this device, set *Output to GPU* to false and select set *Output to* to *B*. This will make sure no data is forwarded to the online pipeline anymore.
5. On the same device you can now reconfigure *Filter pulses* to individual ranges other than the global setting in the manager scene.
6. In a differnt Karabo-GUI, open the *ZMQ Bridge* project corresponding to the instrument and detector. Navigate to the *ZEROMQ/RAW* device and shut it down. Now reconfigure the bridge device to …*output_1* instead of …*output_0*. Make sure it also connects to the correct splitter device id. Then reinstantiate the device.
7. Finally, you can now decrease the train module number on the corresponding data aggregator. You will find them in the corresponding subproject. They are named by channel, where for AGIPD e.g. Q1M1 is channel 0 and Q4M4 is channel 15. You will find the field as *DataDispatcher.trainStride*. Note that *1*, and thus every train, will likely fail. A train stride of *2* has worked in the past.

To revert, step through the sequence in reversed order.

CHAPTER 7

Disabling Calibration Constants

Sometimes it turns out that faulty calibration parameters have been injected. In this case disabling the relevant constant versions will assure that future correction tasks (both online and offline) will not use these parameters anymore.

Warning: This procedure can have severe impact on detector calibration. Only proceed if you know what you are doing!

7.1 Screencast of Procedure

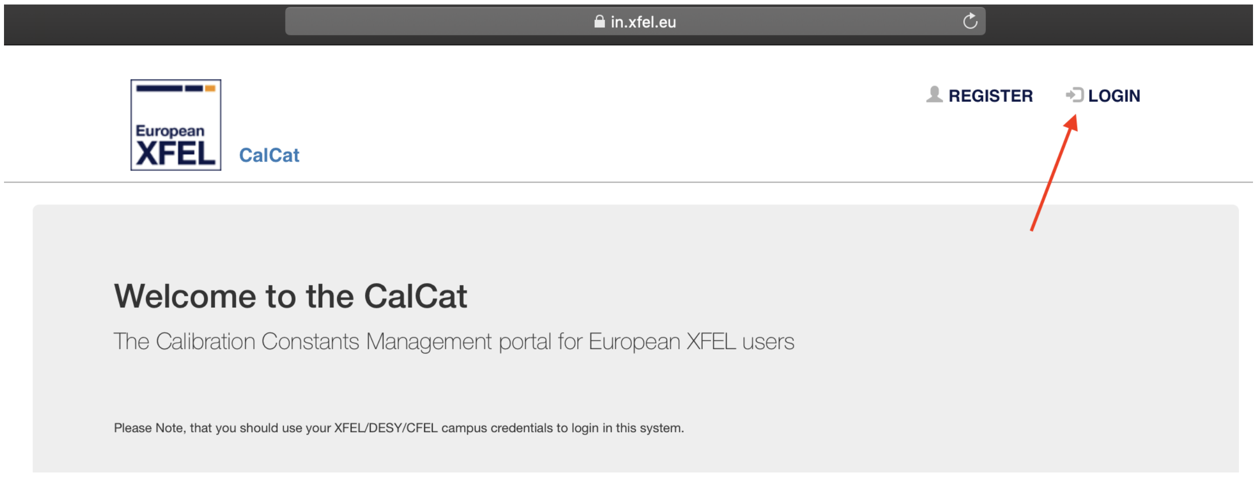
Fig. 7.1: How to disable calibration constants in the database backend.

7.2 Step by Step Instructions

Disabling is done via the calibration database backend.

1. Navigate to the database backend and log in using your normal LDAP credentials. For the following steps you will need elevated access rights, which can be requested at *det-support@xfel.eu*.
2. After logging in, navigate to the *Admin* area, and here to *Calibration constant versions*. This will list you the latest calibration parameters injected into the database.
3. You should now apply filters to find the faulty constants. It is usually most appropriate to filter by *Physical device* and *Start date*. The *Physical device* filter is fuzzy; hence entering e.g. AGIPD will list all AGIPD modules.

   The *Start date* filter will usually match when the data used for characterization was taken.

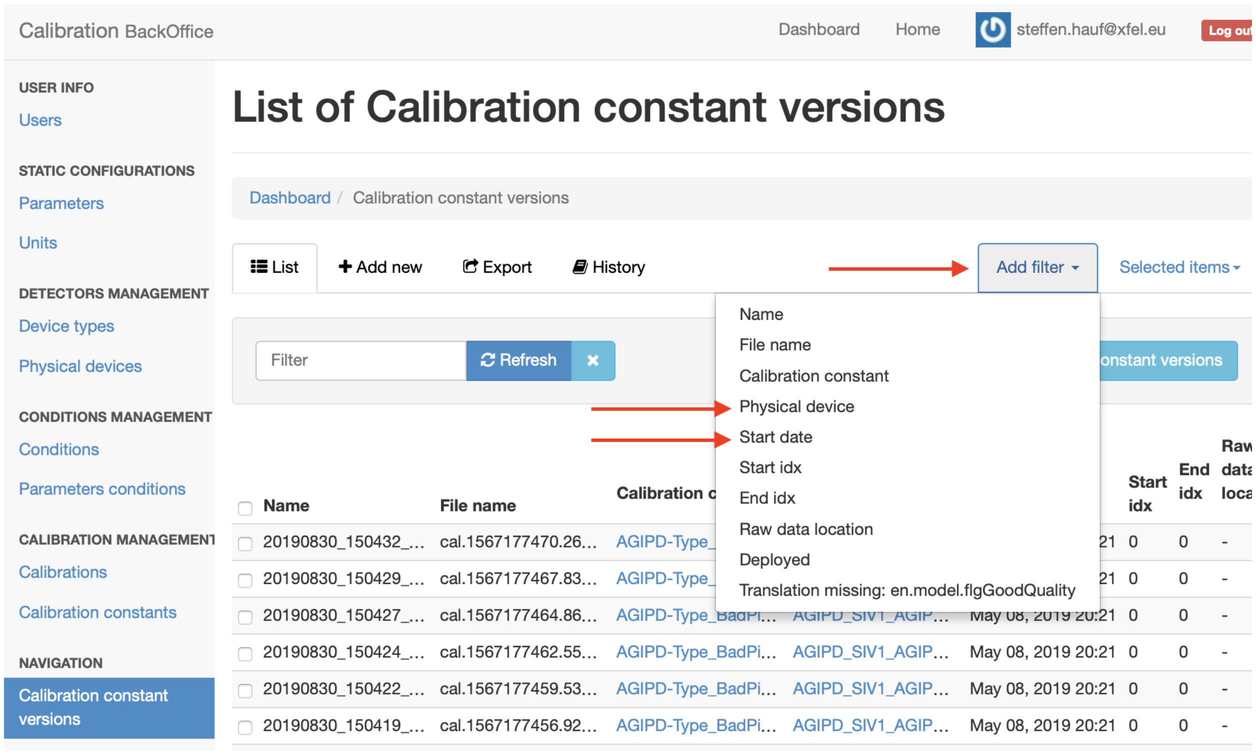


Welcome to the CalCat

The Calibration Constants Management portal for European XFEL users

Please Note, that you should use your XFEL/DESY/CFEL campus credentials to login in this system.

Fig. 7.2: Logging into the calibration database backend

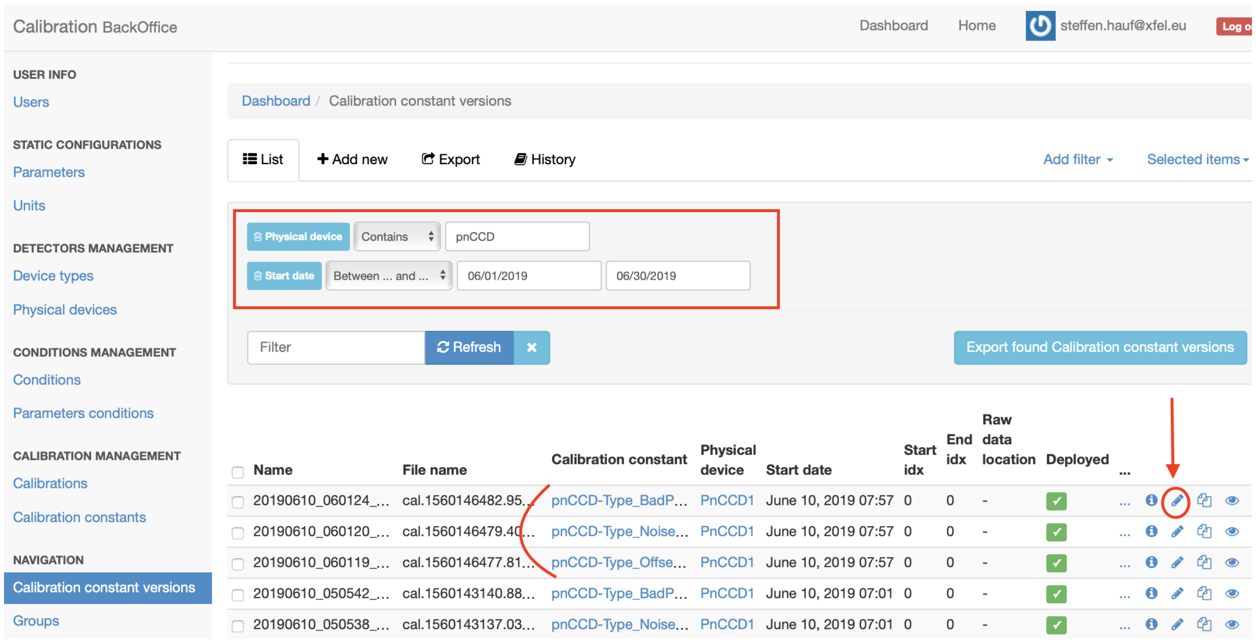


List of Calibration constant versions

Dashboard / Calibration constant versions

Name	File name	Calibration c			Start idx	End idx	Raw data loca
20190830_150432_...	cal.1567177470.26...	AGIPD-Type_		21	0	0	-
20190830_150429_...	cal.1567177467.83...	AGIPD-Type_		21	0	0	-
20190830_150427_...	cal.1567177464.86...	AGIPD-Type_BadPi...	AGIPD_SIV1_AGIP...	May 08, 2019 20:21	0	0	-
20190830_150424_...	cal.1567177462.55...	AGIPD-Type_BadPi...	AGIPD_SIV1_AGIP...	May 08, 2019 20:21	0	0	-
20190830_150422_...	cal.1567177459.53...	AGIPD-Type_BadPi...	AGIPD_SIV1_AGIP...	May 08, 2019 20:21	0	0	-
20190830_150419_...	cal.1567177456.92...	AGIPD-Type_BadPi...	AGIPD_SIV1_AGIP...	May 08, 2019 20:21	0	0	-

Fig. 7.3: The calibration database version overview. Setting a filter is highlighted.


Calibration BackOffice

Dashboard Home steffen.hauf@xfel.eu Log o

USER INFO
Users

STATIC CONFIGURATIONS
Parameters
Units

DETECTORS MANAGEMENT
Device types
Physical devices

CONDITIONS MANAGEMENT
Conditions
Parameters conditions

CALIBRATION MANAGEMENT
Calibrations
Calibration constants

NAVIGATION
Calibration constant versions
Groups

Dashboard / Calibration constant versions

List | Add new | Export | History | Add filter | Selected items

Physical device: Contains pnCCD

Start date: Between ... and ... 06/01/2019 06/30/2019

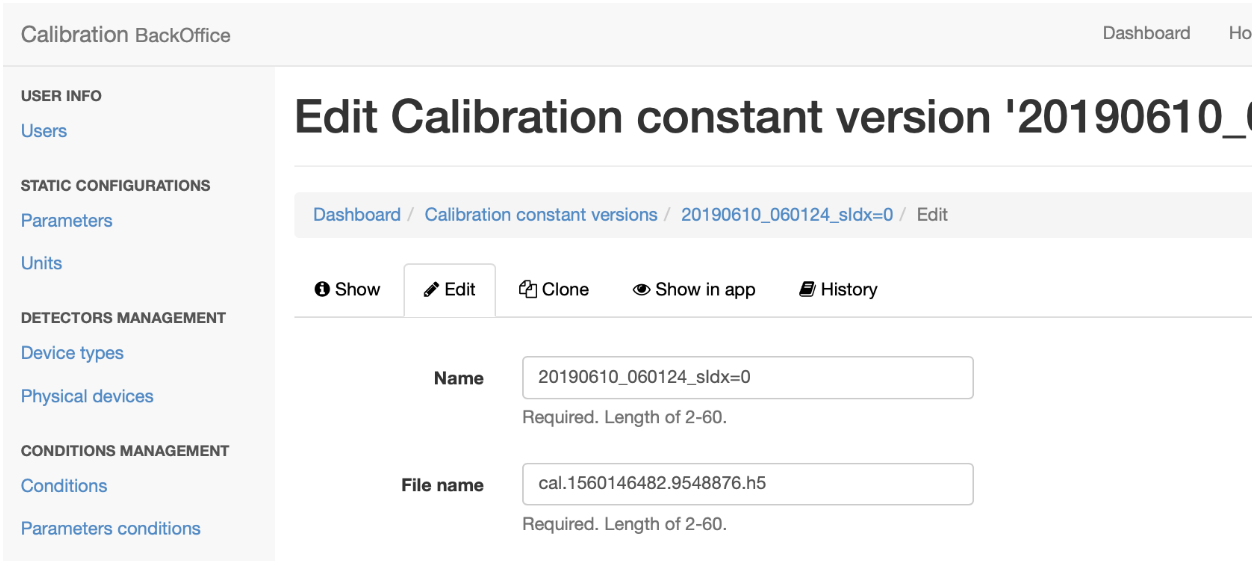
Filter | Refresh | Export found Calibration constant versions

Name	File name	Calibration constant	Physical device	Start date	Start idx	End idx	Raw data location	Deployed	...
20190610_060124_...	cal.1560146482.95...	pnCCD-Type_BadP...	PnCCD1	June 10, 2019 07:57	0	0	-	✓	...
20190610_060120_...	cal.1560146479.40...	pnCCD-Type_Noise...	PnCCD1	June 10, 2019 07:57	0	0	-	✓	...
20190610_060119_...	cal.1560146477.81...	pnCCD-Type_Offse...	PnCCD1	June 10, 2019 07:57	0	0	-	✓	...
20190610_050542_...	cal.1560143140.88...	pnCCD-Type_BadP...	PnCCD1	June 10, 2019 07:01	0	0	-	✓	...
20190610_050538_...	cal.1560143137.03...	pnCCD-Type_Noise...	PnCCD1	June 10, 2019 07:01	0	0	-	✓	...

Fig. 7.4: Setting filter options on the calibration constant version page and selecting versions to disable

4. After having located the faulty parameters you need to disable them. For Dark image derived constants usually come in triplets: Noise, Offset, and Bad Pixels, or for AGIPD additionally with Thresholds. You should disable each one, by clicking on the pen symbol at the far right and editing the constant version in a new tab.
5. On the edit page, scroll to the bottom and uncheck the two boxes labelled *deployed* and good quality'. Then click save.

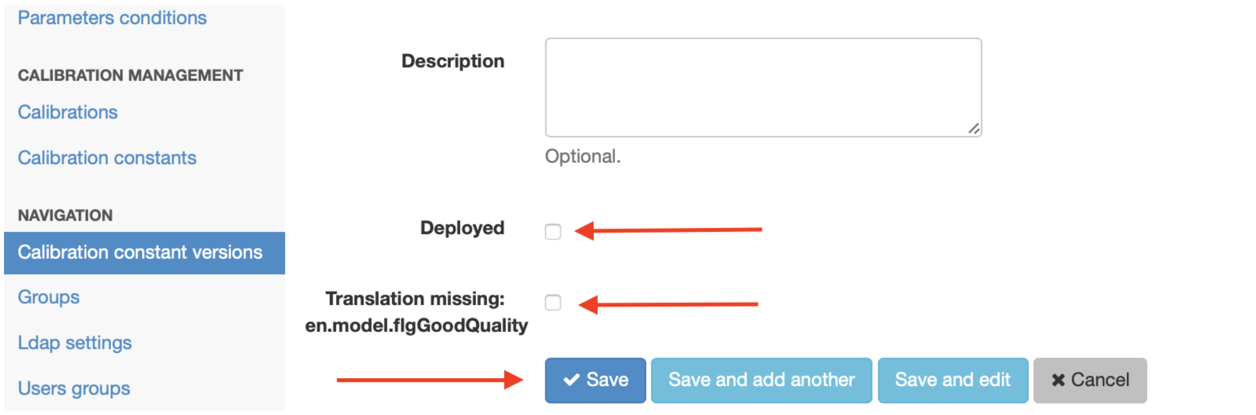
6. Refreshing the filtered calibration constant version view should now list these calibration parameters as disabled.



Edit Calibration constant version '20190610_(

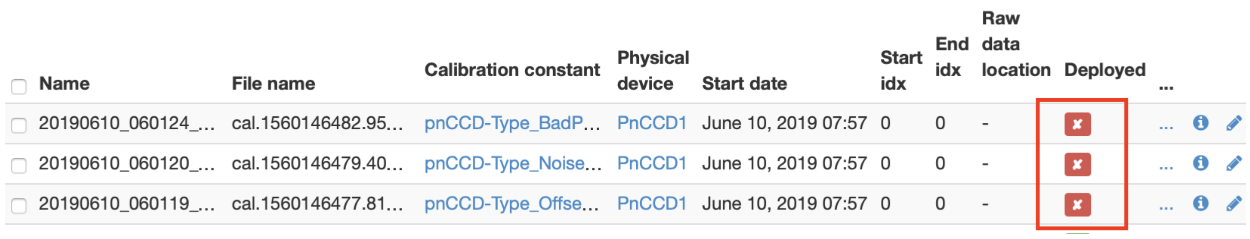
Name	20190610_060124_sIdx=0
	Required. Length of 2-60.
File name	cal.1560146482.9548876.h5
	Required. Length of 2-60.

Fig. 7.5: The edit calibration constant version dialog.



Description	
	Optional.
Deployed	☐
Translation missing: en.model.flgGoodQuality	☐

Fig. 7.6: Disabling deployed and good quality markers for a constant version.



☐	Name	File name	Calibration constant	Physical device	Start date	Start idx	End idx	Raw data location	Deployed	...		
☐	20190610_060124_...	cal.1560146482.95...	pnCCD-Type_BadP...	PnCCD1	June 10, 2019 07:57	0	0	-	✗	...	ℹ	✎
☐	20190610_060120_...	cal.1560146479.40...	pnCCD-Type_Noise...	PnCCD1	June 10, 2019 07:57	0	0	-	✗	...	ℹ	✎
☐	20190610_060119_...	cal.1560146477.81...	pnCCD-Type_Offse...	PnCCD1	June 10, 2019 07:57	0	0	-	✗	...	ℹ	✎

Fig. 7.7: Disabled versions in the overview.

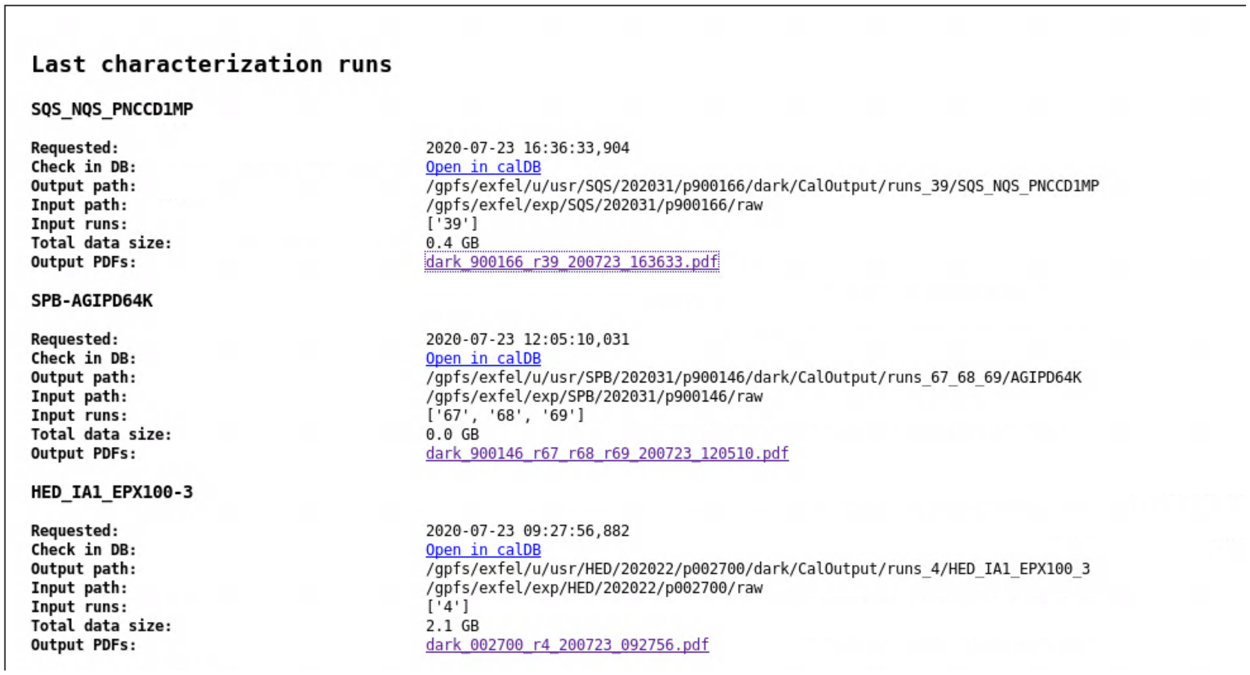
CHAPTER 8

Offline Characterization Reports

All offline characterization tasks run via *xfel_calibrate* should produce a report in the report folder given upon task execution. This report is generated from the Jupyter notebooks which are run as part of the processing job.

For dark characterization tasks launched using the *Requesting Dark Characterization* in the max-exfl016:8008 web page, the output reports can be access through:

A) Accessing last report characterization reports from *max-exfl016:8008* as shown



```
Last characterization runs

SQS_NQS_PNCCD1MP

Requested:              2020-07-23 16:36:33,904
Check in DB:            Open in calDB
Output path:            /gpfs/exfel/u/usr/SQS/202031/p900166/dark/CalOutput/runs_39/SQS_NQS_PNCCD1MP
Input path:             /gpfs/exfel/exp/SQS/202031/p900166/raw
Input runs:             ['39']
Total data size:        0.4 GB
Output PDFs:            dark_900166_r39_200723_163633.pdf

SPB-AGIPD64K

Requested:              2020-07-23 12:05:10,031
Check in DB:            Open in calDB
Output path:            /gpfs/exfel/u/usr/SPB/202031/p900146/dark/CalOutput/runs_67_68_69/AGIPD64K
Input path:             /gpfs/exfel/exp/SPB/202031/p900146/raw
Input runs:             ['67', '68', '69']
Total data size:        0.0 GB
Output PDFs:            dark_900146_r67_r68_r69_200723_120510.pdf

HED_IA1_EPX100-3

Requested:              2020-07-23 09:27:56,882
Check in DB:            Open in calDB
Output path:            /gpfs/exfel/u/usr/HED/202022/p002700/dark/CalOutput/runs_4/HED_IA1_EPX100_3
Input path:             /gpfs/exfel/exp/HED/202022/p002700/raw
Input runs:             ['4']
Total data size:        2.1 GB
Output PDFs:            dark_002700_r4_200723_092756.pdf
```

B) Accessing all dark reports requested for all Instruments from the link at the end of *Last characterization runs*.

C) In a standardized location, identified as:

Full list of dark characterizations

```
/gpfs/exfel/d/cal/caldb_store/xfel/reports/{instrument}/{det_instance}/dark/dark_
↪{proposal}_{runs}_{time_stamp}
```

For all other task the location will vary according to the input.

Reports are in PDF format and may be opened using any standard PDF viewer.

8.1 No Report is Generated

If you have launched a characterization job and after a reasonable time (~ 30 minutes) no report has appeared in the anticipated location, the following points should be checked:

- has the job been executed and finished? Check on Maxwell if it is very busy and if jobs on the *xcal* account are still queued:

```
ssh user@max-exfl
squeue  -u xcal -o "%.18i %60j %.3t %.10M %R"
```

- has the job been launched at all? Check the logs of the calibration webservice, if your job has launched without an error:

  A) Check written logs in *Webservice log* section in *max-exfl016:8008* web page.

  B) With permission for xcal, check log file directly:

  ```
  ssh xcal@max-exfl016
  grep "{DETECTOR} {TYPE}" calibration_webservice_run/webservice/web.log
  ```

  Here {DETECTOR}‘ should be e.g. AGIPD and *{TYPE}* frequently will be *DARK*.

- were there failures in the report rendering? Checking this is a bit more tedious and a solution will require intervention by an expert as the notebook generating the code will likely have to be updated.

  First check if there is a an entry for your characterizaton task in the *temp* directory of the webservice. It should not have been deleted after job completion if rendering errors were present:

```
ssh xcal@max-exfl016
ll -tr calibration_webservice_run/webservice/temp
```

  will list you a time sorted list of entries in the directory, with each folder named *slurm_out_<detector>_<task>_t<time_stamp>*. Investigate those closest to the time when you launched the characterization job. In the slurm sub-folders you will find the notebooks executed and an *inputParameters.rst*. Inspecting this file should allow you to confirm the corrected given input parameters.

  Now check the output of the *slurmXXX.out* file with the latest file modification date. This is the output from the slurm job responsible for running the PDF rendering process and finally copying the final report to the output directory. Hence it will contain any errors that prevented correct execution and led to the missing report. Inform *det-support@xfel.eu* about your observations.

8.2 Report is Generated with many errors.

- has the constants been created and injected to the database? If constants were injected to the database, the notebook should have printed messages similar to:

```
Inject Offset constants from 2020-07-23 09:06:39+02:00
Inject Noise constants from 2020-07-23 09:06:39+02:00
```

  A) If they are injected to the database, then the errors might be related to the plotting in the report. This should not prevent the online calibration from retrieving the constants as long as it was injected with the right parameter conditions.

  B) If they were not injected to the database, then check if all Input parameters in the first report pages are correct and inform *det-support@xfel.eu* about your observations to correct the input configurations from the offline calibration webservice.

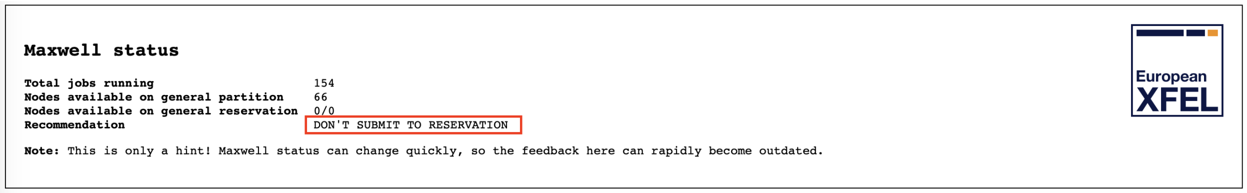
CHAPTER 9

Webservice Overview

At *max-exfl016:8008* an overview of the current calibration webservice status is available.

Note: This is currently still considered an experimental service, so there might be unexpected down times.

9.1 Maxwell Status

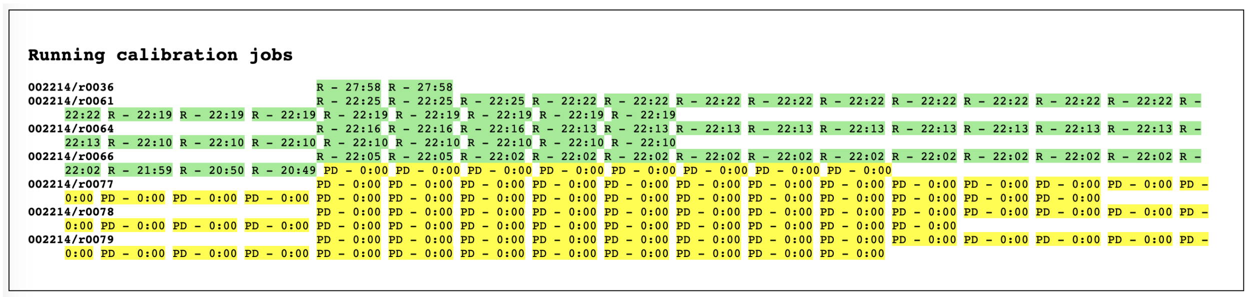


```
Maxwell status

Total jobs running                      154
Nodes available on general partition    66
Nodes available on general reservation  0/0
Recommendation                          DON'T SUBMIT TO RESERVATION

Note: This is only a hint! Maxwell status can change quickly, so the feedback here can rapidly become outdated.
```

Fig. 9.1: The Maxwell overview status panel. It indicates the current number of busy and free nodes, as well as nodes available on the *xcal* reservation.

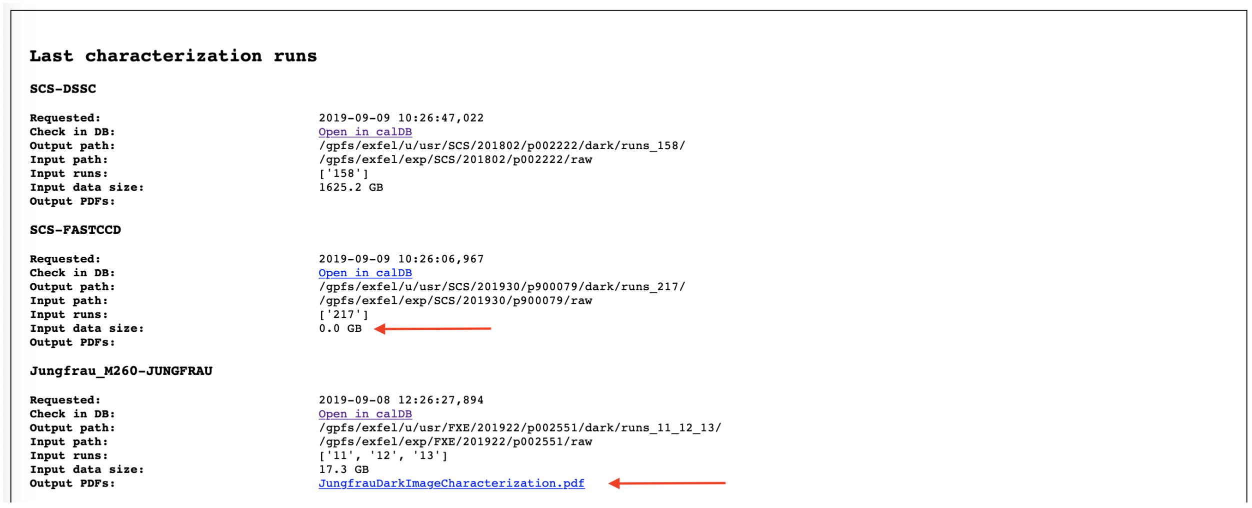
It will also give you a recommendation on whether it makes sense to submit to a user proposal reservation, if available, or if there are enough free nodes on the general cluster partition.

Note: This is only a momentary snap shot and may change any second.



```
Running calibration jobs

002214/r0036                          R - 27:58 R - 27:58
002214/r0061                          R - 22:25 R - 22:25 R - 22:25 R - 22:22 R - 22:22 R - 22:22 R - 22:22 R - 22:22 R - 22:22 R - 22:22 R - 22:22 R - 22:22 R -
     22:22 R - 22:19 R - 22:19 R - 22:19 R - 22:19 R - 22:19 R - 22:19 R - 22:19 R - 22:19
002214/r0064                          R - 22:16 R - 22:16 R - 22:16 R - 22:13 R - 22:13 R - 22:13 R - 22:13 R - 22:13 R - 22:13 R - 22:13 R - 22:13 R - 22:13 R -
     22:13 R - 22:10 R - 22:10 R - 22:10 R - 22:10 R - 22:10 R - 22:10 R - 22:10 R - 22:10
002214/r0066                          R - 22:05 R - 22:05 R - 22:02 R - 22:02 R - 22:02 R - 22:02 R - 22:02 R - 22:02 R - 22:02 R - 22:02 R - 22:02 R - 22:02 R -
     22:02 R - 21:59 R - 20:50 R - 20:49 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00
002214/r0077                          PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD -
     0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00
002214/r0078                          PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD -
     0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00
002214/r0079                          PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD -
     0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00 PD - 0:00
```

Fig. 9.2: An overview of the jobs currently being run through the webservice. Jobs currently being executed are shown in green, pending jobs are indicated yellow.



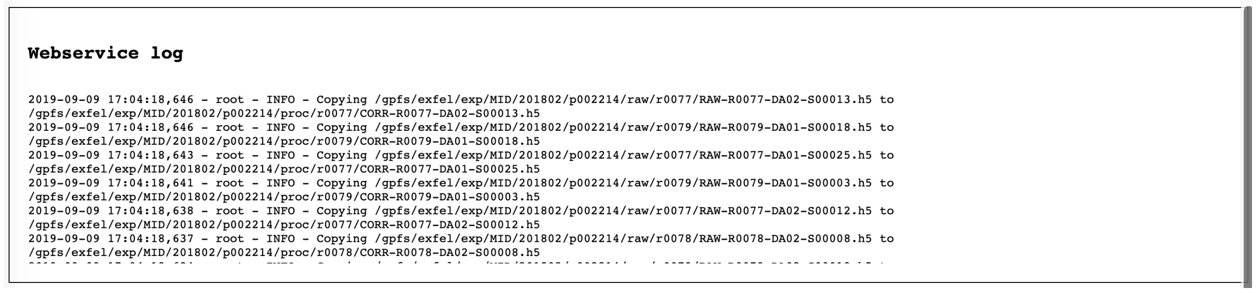
```
Last characterization runs

SCS-DSSC

Requested:                 2019-09-09 10:26:47,022
Check in DB:               Open in calDB
Output path:               /gpfs/exfel/u/usr/SCS/201802/p002222/dark/runs_158/
Input path:                /gpfs/exfel/exp/SCS/201802/p002222/raw
Input runs:                ['158']
Input data size:           1625.2 GB
Output PDFs:

SCS-FASTCCD

Requested:                 2019-09-09 10:26:06,967
Check in DB:               Open in calDB
Output path:               /gpfs/exfel/u/usr/SCS/201930/p900079/dark/runs_217/
Input path:                /gpfs/exfel/exp/SCS/201930/p900079/raw
Input runs:                ['217']
Input data size:           0.0 GB
Output PDFs:

Jungfrau_M260-JUNGFRAU

Requested:                 2019-09-08 12:26:27,894
Check in DB:               Open in calDB
Output path:               /gpfs/exfel/u/usr/FXE/201922/p002551/dark/runs_11_12_13/
Input path:                /gpfs/exfel/exp/FXE/201922/p002551/raw
Input runs:                ['11', '12', '13']
Input data size:           17.3 GB
Output PDFs:               JungfrauDarkImageCharacterization.pdf
```

Fig. 9.3: Last characterizations for each instrument and detector. This will indicate the total file size of files in the run(s), link to the calibration database entries for this detector, and provide a download link for the PDF report, if one was generated.

9.2 Running Calibration Jobs

9.3 Last Characterization Runs

9.4 Webservice Log



Webservice log

```
2019-09-09 17:04:18,646 - root - INFO - Copying /gpfs/exfel/exp/MID/201802/p002214/raw/r0077/RAW-R0077-DA02-S00013.h5 to
/gpfs/exfel/exp/MID/201802/p002214/proc/r0077/CORR-R0077-DA02-S00013.h5
2019-09-09 17:04:18,646 - root - INFO - Copying /gpfs/exfel/exp/MID/201802/p002214/raw/r0079/RAW-R0079-DA01-S00018.h5 to
/gpfs/exfel/exp/MID/201802/p002214/proc/r0079/CORR-R0079-DA01-S00018.h5
2019-09-09 17:04:18,643 - root - INFO - Copying /gpfs/exfel/exp/MID/201802/p002214/raw/r0077/RAW-R0077-DA01-S00025.h5 to
/gpfs/exfel/exp/MID/201802/p002214/proc/r0077/CORR-R0077-DA01-S00025.h5
2019-09-09 17:04:18,641 - root - INFO - Copying /gpfs/exfel/exp/MID/201802/p002214/raw/r0079/RAW-R0079-DA01-S00003.h5 to
/gpfs/exfel/exp/MID/201802/p002214/proc/r0079/CORR-R0079-DA01-S00003.h5
2019-09-09 17:04:18,638 - root - INFO - Copying /gpfs/exfel/exp/MID/201802/p002214/raw/r0077/RAW-R0077-DA02-S00012.h5 to
/gpfs/exfel/exp/MID/201802/p002214/proc/r0077/CORR-R0077-DA02-S00012.h5
2019-09-09 17:04:18,637 - root - INFO - Copying /gpfs/exfel/exp/MID/201802/p002214/raw/r0078/RAW-R0078-DA02-S00008.h5 to
/gpfs/exfel/exp/MID/201802/p002214/proc/r0078/CORR-R0078-DA02-S00008.h5
```

Fig. 9.4: This outputs the last lines of the log files of the webservice.

CHAPTER 10

General Detector DAQ Troubleshooting

The following general points should be verified in case a detector is reported to have DAQ problems:

10.1 No data written to disk

- Verify that indeed no data is written to disk. You need to be on the control-network to see, *Logging onto exflgateway* in case you are not in the hutch. Go to one of the online cluster servers:

SASE	suggested server
1	exflonc10
2	exflonc17
3	exflonc14

  Then check file sizes for the last run:

  ```
  ssh <username>@<server>
  ll -alh /gpfs/exfel/exp/<instrument>/<cycle>/<proposal>/raw/<run>
  ```

  For the MHz detector you expect file sizes in the gigabyte range.

- Is the DAQ in the *RECORD* state? If not put the DAQ into the *RECORD* state.
- Is the detector sending data? Check that the detector is powered, configured and sending data. You can further verify this via the RunDeck tool, which is linked in the browser bookmarks of each instrument.
- Are the data aggregators in an error state? The DAQ projects are subprojects of the MHz detector top-level project; check the 16 XTDF data aggregators therein in the project panel. If they are in error state, call ITDM OCD.

10.2 No data on Online Views

- Is the DAQ in the *MONITOR* or *RECORD* state? If not put the DAQ into into the *MONITOR*.
- Is the detector sending data? Check that the detector is powered, configured and sending data. You can further verify this via the RunDeck tool, which is linked in the browser bookmarks of each instrument.
- Is the online calbration pipeline started, and are constants loaded? You can check this on the *MANAGER* scene, all devices should be active, and indicator lights should be green for any device connected to a data output. Date and times should be shown for all constants.
- Is the pipeline output data to the GUI? This can be checked in the *MANAGER* scene. The *APPENDER* devices show time of flight and update rates. Are these updating? If not, check if the minimum number of modules configured matches the number of modules in the detector actually producing data. If updates are happening, the GUI is not displaying them: call CAS OCD.

CHAPTER 11

General Online Calibration Troubleshooting

11.1 Restart Servers Script does not run through

Restarting the calibration pipeline servers is the initial step of re-initializing the larger calibration pipelines. For AGIPD, DSSC and LPD, this is done using the restart server script (see *Restarting Online Calibration*). In case of misconfiguration or addition of new servers this script has stalled in the past. You will see this if updates in the restart server status field stop before "Restarted all servers" is displayed. Failures have previously occurred for the following reasons:

- a new device server had been added on one of the hosts the restart works on:

Instrument	Hosts	Account
SPB	spb-br-sys-cal-[0-7]	spbonc
FXE	sa1-br-sys-cal-[0-7]	fxeonc
SCS	scs-br-sys-cal-[0-7]	scsonc
SQS	sqs-br-sys-cal-1	sqsonc
MID	mid-br-sys-cal-[0-7]	midonc
HED	hed-br-sys-cal-1	hedonc

  Such new device servers, which should usually not be restarted, need to be added to the *Omit from restart* field of the *RESTART* device which an expression that matches (parts of) the server naming (see Fig. 11.1).

- a webserver on one of the hosts is not running. The restart works by calling karabo webservers. Hence, if they are not running, it will fail. SSH onto the respective hosts (see table above) and check that the servers are running:

```
source /scratch/[account]/karabo/activate
karabo-check
```

  Should include a line

```
webserver: up (pid 1249) 2475094 seconds, normally down, running
```

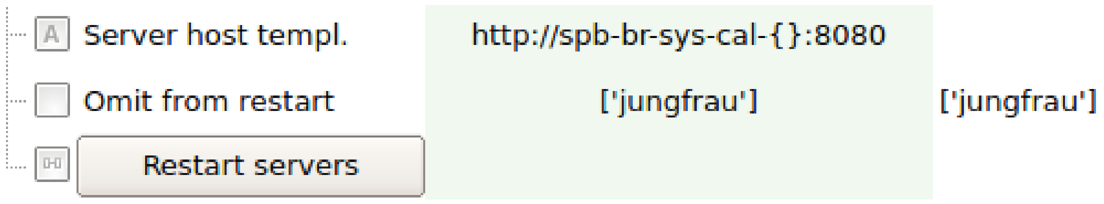
If not one needs to investigate why it is not running (CAS OCD will help).

- a webserver used to restart calibration is misconfigured in its filter settings:

```
[spbonc@exflong01 ~]$ cat /scratch/spbonc/karabo/var/service/webserver/run
#!/bin/bash
# this file has been generated by ansible
exec 2>&1
exec envdir ../../environment env \
karabo-webserver serverId=webserver \
--port 8080 --filter cppspb_spb_cal_1 pyspb_spb_cal_1 mlspb_spb_cal_
```

  Note how it will only act on the servers specified. If a server runs on that host that does not match this criterion and is not exempt by the omit from restart settings, the restart will stall.

- the template used to address the webservers is misconfigured in the *RESTART* device.



Server host templ.	http://spb-br-sys-cal-{}:8080	
Omit from restart	['jungfrau']	['jungfrau']

Fig. 11.1: The *RESTART* device configuration

11.2 Start Pipeline Times Out on Initializing Devices

If the restart ran successfully and reinitializing devices has failed when clicking on *Start Pipeline*. This has happened for the following reasons in the past:

- the Detector is actively pushing data during a restart. This might result in a device not correctly initializing. In general: the DAQ can stay in Monitoring mode during restart, but the detector must not send data! If this happened restart the servers, and then try again.
- a Karabo installation on one of the hosts was not started at all. Check on each of an instruments hosts

Instrument	Hosts	Account
SPB	spb-br-sys-cal-[0-7]	spbonc
FXE	sa1-br-sys-cal-[0-7]	fxeonc
SCS	scs-br-sys-cal-[0-7]	scsonc
SQS	sqs-br-sys-cal-1	sqsonc
MID	mid-br-sys-cal-[0-7]	midonc
HED	hed-br-sys-cal-1	hedonc

  that Karabo is running:

```
source /scratch/[account]/karabo/activate
karabo-check
```

  If not start it:

```
karabo-start
```

and try again.

- the execution environment of a GPU enabled server is not configured properly:

```
cat /scratch/spbonc/karabo/var/service/pyspb_spb_cal_1/run
#!/bin/bash
# this file has been generated by ansible
exec 2>&1
exec envdir ../../environment env \
PATH=/usr/local/cuda-10.1/bin/:$PATH \
karabo-pythonserver serverId=pySPB/spb_cal_1 \
Logger.priority=INFO
```

  Note how *CUDA* is added to the path. In general the code should not fail if no GPU is present, but rather it will lead to much slower processing (see . . .) for details)

  Also, cross check that the used *CUDA* version is installed and available.

```
ll /usr/local/cuda-10.1
```

- an older installation has been started and device ids are already in use. This has happened a few times. Before we had GPUs available the online calibration was installed on other (CPU) only hosts. From time to time, something or someone seems to start these again, conflicting with the current installation. No calibration devices or servers should be running on exflonc hosts for the large detectors. You can check e.g. via iKarabo.

11.3 Calibration Parameters are not loaded

After initializing devices the calibration manager device instructs all processing devices to load their respective calibration parameters. This will usually show itself by having the calibration devices stuck in *CHANGING* state and the calibration parameter dates empty. This has in the past failed for the following reasons:

- the detector is producing data when constants are loaded. This might lead to failures. Stop the detector, restart the pipeline and try again.
- the calibration DB remote devices are not running. These devices are under ITDM responsiblity, but may sometimes have note been initialized yet after a reboot or power failure. In general they run on the same hosts as the calibration pipeline under the xcal account:

Instrument	Hosts	Account
SPB	spb-br-sys-cal-[0-8]	xcal
FXE	fxe-br-sys-cal-[0-8]	xcal
SCS	scs-br-sys-cal-[0-8]	xcal
SQS	scs-br-sys-cal-[0-8]	xcal
MID	mid-br-sys-cal-[0-8]	xcal
HED	mid-br-sys-cal-[0-8]	xcal

  Note how the instruments without MHz detectors also use these devices.

  You can identify this problem by grepping the calibration device logs for *Resource temporarly not available*:

```
grep "Resource temporarily" /scratch/scsonc/karabo/var/log/pythonserver_scs_cal_1/
↪current
WARN  SCS_CDIDET_DSSC/CAL/OFFSET_CORR_Q1M2  : Could not find constant Offset:␣
↪Resource temporarily unavailable
WARN  SCS_CDIDET_DSSC/CAL/OFFSET_CORR_Q3M2  : Could not find constant Offset:␣
↪Resource temporarily unavailable
```

In such a case check the respective devices responsible for the SASE:

```
ssh xcal@spb-br-sys-cal-0
[xcal@exflong08 ~]$ source /scratch/xcal/karabo/activate
[xcal@exflong08 ~]$ karabo-check
pythonserver_sa1_daq_rcal_0: up (pid 32532) 59542 seconds, normally down, running
```

If the server is not running, restart it or resolve the problem. The devices are configured to auto-start. Then restart the pipeline.

- detector operating settings do not match anything in the database. To investigate this, first check the the common operating conditions set on the *CAL_MANAGER* device if they are what you expect (if you know what to expect that is). An example is shown in Fig. 11.2. Note that beam_energy should usually be empty. Also you do not want any fields to display “None”. If this is the case, delete “None” such that the field is empty, or set it to a value.

  If you do not know what to expect, chances are that the last offsets were injected with reasonable settings to the calibration database. To check to the following:

  Navigate to the database backend and log in. Click *Admin->Calibration constant versions* and filter for the detector you are investigating by adding a physical device filter:

  Note that for AGIPD you will need to differentiate MID and SPB instances. If you know approximately when the last dark run was taken you can usually do this via the *Start date* column. If not, you will find the current module mapping here. The SPB AGIPD is the AGIPD1M1 instance, the MID the AGIPD1M2 instance.

  Click on a constant which contains the work *Offset* in its name. The click on the link under *Condition*. Here you will now find a list of *Parameter conditions* which you can inspect to see if their values match the expectations in the common operating conditions field.

  Note that the field will likely contain additional parameters not applicable to calibration parameters produced from dark images. These change much less frequently though, so setting the appropriate parameters for dark image derived parameters will suffice in most cases.

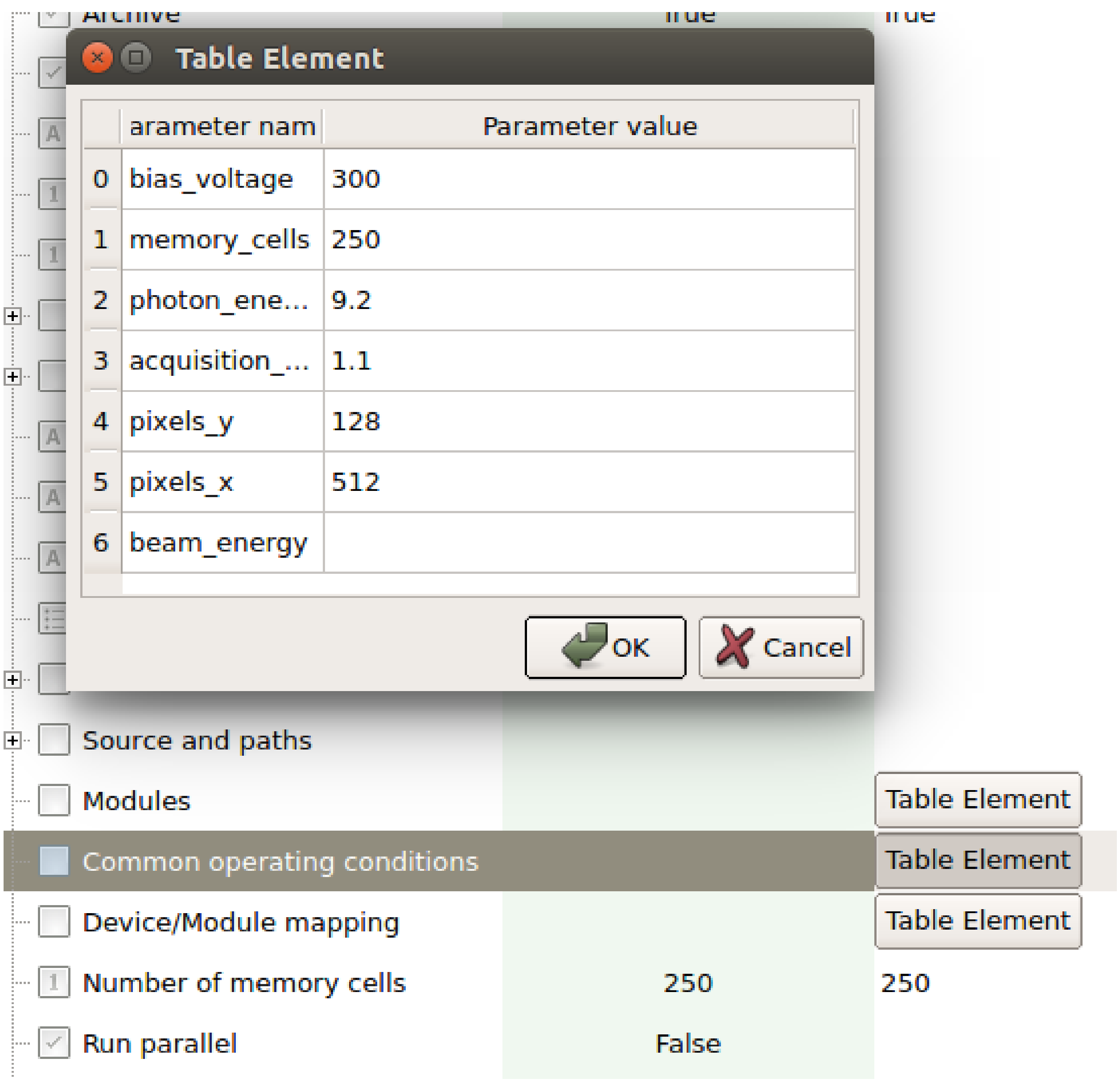
  Once you are done, click apply all on the calibration manager device, then click reset and finally init.

Note: The AGIPD pipeline will run if no flat field constants (FF) are found.

11.4 “Wrong” Calibration parameters are loaded

Sometime other parameters than expected seem to be loaded. Most frequently, offsets are of older date than the users expect. Verify the following:

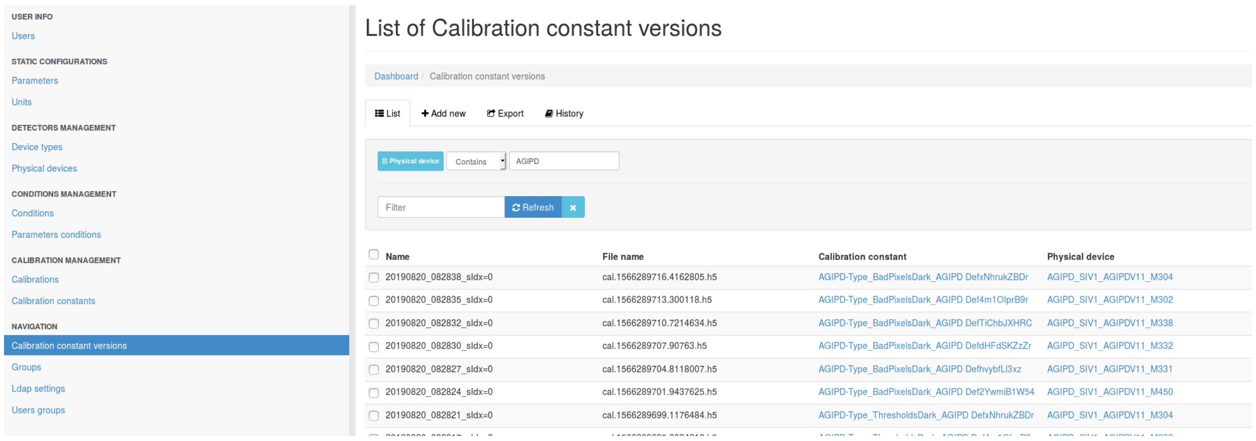
- have new offsets been characterized?, e.g. has the procedure for dark characterization been followed (see *Requesting Dark Characterization*). Has data been been migrated, before running the script? Is Maxwell particularly busy, such that jobs are queued? In this case it is advisable to wait till the jobs are finished and try to click *reset*, then click *init* in the *CAL MANAGER* to retrieve and use the correct constants. If you are unsure, you can check the database backend for the latest injected constants and verify these are the ones you expect.
- do the detector operating conditions match what is set in the *CAL MANAGER* device. This can quite frequently occur if operators have changed AGIPD operation scenarios from 1.1 MHz to e.g. 4.5 MHz or change the number of memory cells. click *verify config* on the karabo scene if available, which will check the table with the detector middle layer and generate the verification result on the *CAL MANAGER* status. If this button is not presence then ask if detector operations has recently been changed, and then adjust the settings as described in *Calibration Parameters are not loaded*.



	arameter nam	Parameter value
0	bias_voltage	300
1	memory_cells	250
2	photon_ene...	9.2
3	acquisition_...	1.1
4	pixels_y	128
5	pixels_x	512
6	beam_energy	

Source and paths		
Modules		Table Element
Common operating conditions		Table Element
Device/Module mapping		Table Element
Number of memory cells	250	250
Run parallel	False	

Fig. 11.2: Example of common operating conditions for the SPB AGIPD.



List of Calibration constant versions

Physical device: Contains AGIPD

Filter

Name	File name	Calibration constant	Physical device
20190820_082838_sIdx=0	cal.1566289716.4162805.h5	AGIPD-Type_BadPixelsDark_AGIPD DefxNhrukZBDr	AGIPD_SIV1_AGIPDV11_M304
20190820_082835_sIdx=0	cal.1566289713.300118.h5	AGIPD-Type_BadPixelsDark_AGIPD Def4m1OlprB9r	AGIPD_SIV1_AGIPDV11_M302
20190820_082832_sIdx=0	cal.1566289710.7214634.h5	AGIPD-Type_BadPixelsDark_AGIPD DefTiChbJXHRC	AGIPD_SIV1_AGIPDV11_M338
20190820_082830_sIdx=0	cal.1566289707.90763.h5	AGIPD-Type_BadPixelsDark_AGIPD DefdHFdSKZzZr	AGIPD_SIV1_AGIPDV11_M332
20190820_082827_sIdx=0	cal.1566289704.8118007.h5	AGIPD-Type_BadPixelsDark_AGIPD DefhvybfLl3xz	AGIPD_SIV1_AGIPDV11_M331
20190820_082824_sIdx=0	cal.1566289701.9437625.h5	AGIPD-Type_BadPixelsDark_AGIPD Def2YwmiB1W54	AGIPD_SIV1_AGIPDV11_M450
20190820_082821_sIdx=0	cal.1566289699.1176484.h5	AGIPD-Type_ThresholdsDark_AGIPD DefxNhrukZBDr	AGIPD_SIV1_AGIPDV11_M304

Fig. 11.3: Example of constants injected into the calibration database.

11.5 Pipeline Initialized but no Image Updates on the GUI

If all (expected) constants loaded, but pushing data through the pipeline does not result in corrected data being displayed, check the following:

- on the *CAL MANAGER* scene (see Fig. 11.5) is the RAW appender updating, are all splitter input rates updating according to the *Output every nth train setting*? If so, data is entering the pipeline and something went wrong on initialization. Perform the check mentioned above. If there are no updates, check that the DAQ is in monitoring/acquiring state and that the proper data sources are selected, try again. Follow general DAQ trouble shooting instruction (*General Detector DAQ Troubleshooting*).
- If you see updates on the splitters but no updated images, in the RAW appender, go to the *RAW OVERVIEW* scene and reduce the minimum number of modules required for forwarding data.

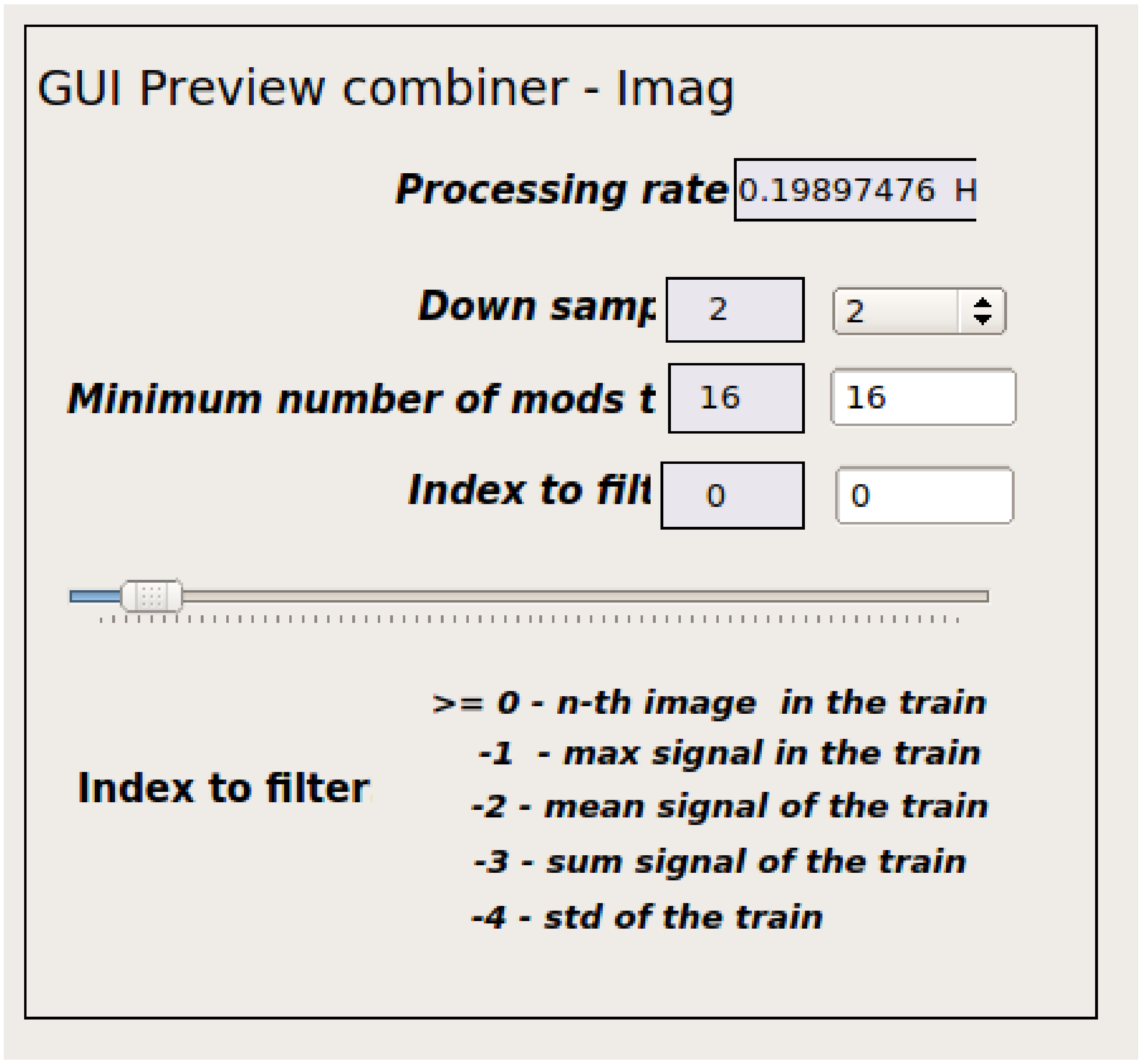
  If some modules are updating, but others are not pushing data into the pipeline. Check if the respective DAQ aggregators are in an error state and call ITDM OCD in case this is the issue. If the DAQ looks fine, use the run-deck tool to verify these modules are sending data. Troubleshoot the detector if not.

  In case you want to continue with a reduced number of modules, make sure the CORRECTED appender is also set to reflect this through setting parameter "min-modules=available number of modules".
- the RAW appender is updating but the CORRECTED appender is not updating for an AGIPD detector, right after a pipeline restart. The *MEDIUM GAIN ADJUST* are not in the *ACTIVE* (green) state.

  This is normal. These devices need to first calculate some parameters, which can take up to a minute after first pushing data. It is important that all devices left of them have switched to *ACTIVE*. If this is the case, simply wait. If not, troubleshoot as above.
- the APPENDER rates are updating, but the images are not. This is likely then a GUI connection problem. Try a different GUI server and call CAS OCD.

11.6 Pipeline is slow

In general the pipeline has been shown to handle up to 256 images/s in various settings with about 2000ms latency for corrected data. A conservative always safe setting are train rate and pulse filter numbers which multiply to 128 images/s. This should give about 1500ms latency for corrected data. If this is not achieved check the following:



GUI Preview combiner - Imag

Processing rate 0.19897476 H

Down samp 2 2

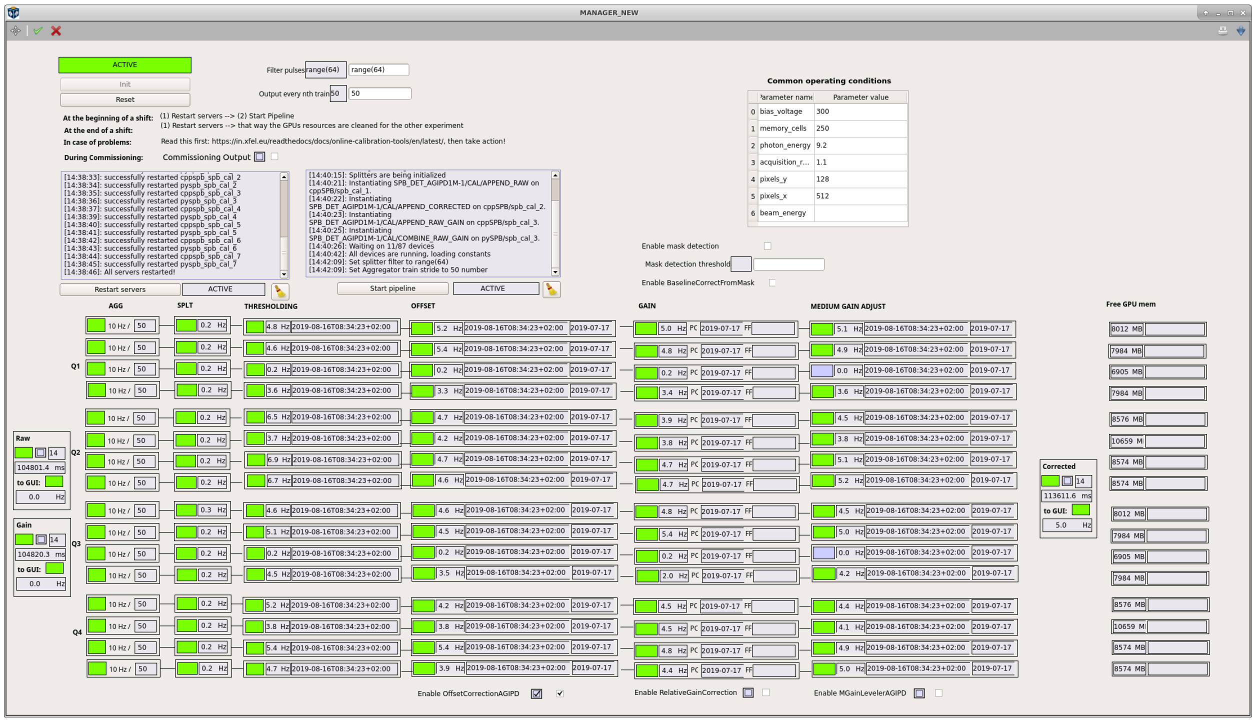
Minimum number of mods t 16 16

Index to filt 0 0

Index to filter

>= 0 - n-th image in the train
-1 - max signal in the train
-2 - mean signal of the train
-3 - sum signal of the train
-4 - std of the train

Fig. 11.4: APPENDER setting for online previews.



Filter pulses range(64) range(64)

Output every nth train 50 50

At the beginning of a shift: (1) Restart servers --> (2) Start Pipeline

At the end of a shift: (1) Restart servers --> that way the GPUs resources are cleaned for the other experiment

In case of problems: Read this first: https://in.xfel.eu/readthedocs/docs/online-calibration-tools/en/latest/, then take action!

During Commissioning: Commissioning Output

```
[14:38:33]: successfully restarted cppspb_spb_cal_2
[14:38:34]: successfully restarted pyspb_spb_cal_2
[14:38:35]: successfully restarted cppspb_spb_cal_3
[14:38:36]: successfully restarted pyspb_spb_cal_3
[14:38:37]: successfully restarted cppspb_spb_cal_4
[14:38:39]: successfully restarted pyspb_spb_cal_4
[14:38:40]: successfully restarted cppspb_spb_cal_5
[14:38:41]: successfully restarted pyspb_spb_cal_5
[14:38:42]: successfully restarted cppspb_spb_cal_6
[14:38:43]: successfully restarted pyspb_spb_cal_6
[14:38:44]: successfully restarted cppspb_spb_cal_7
[14:38:45]: successfully restarted pyspb_spb_cal_7
[14:38:46]: All servers restarted!
```

```
[14:40:15]: Splitters are being initialized
[14:40:21]: Instantiating SPB_DET_AGIPD1M-1/CAL/APPEND_RAW on
cppSPB/spb_cal_1.
[14:40:22]: Instantiating
SPB_DET_AGIPD1M-1/CAL/APPEND_CORRECTED on cppSPB/spb_cal_2.
[14:40:23]: Instantiating
SPB_DET_AGIPD1M-1/CAL/APPEND_RAW_GAIN on cppSPB/spb_cal_3.
[14:40:25]: Instantiating
SPB_DET_AGIPD1M-1/CAL/COMBINE_RAW_GAIN on pySPB/spb_cal_3.
[14:40:26]: Waiting on 11/87 devices
[14:40:42]: All devices are running, loading constants
[14:42:09]: Set splitter filter to range(64)
[14:42:09]: Set Aggregator train stride to 50 number
```

Common operating conditions

	Parameter name	Parameter value
0	bias_voltage	300
1	memory_cells	250
2	photon_energy	9.2
3	acquisition_r...	1.1
4	pixels_y	128
5	pixels_x	512
6	beam_energy	

Enable mask detection ☐

Mask detection threshold

Enable BaselineCorrectFromMask ☐

	AGG	SPLT	THRESHOLDING	OFFSET	GAIN	MEDIUM GAIN ADJUST	Free GPU mem
Q1	10 Hz / 50	0.2 Hz	4.8 Hz 2019-08-16T08:34:23+02:00	5.2 Hz 2019-08-16T08:34:23+02:00 2019-07-17	5.0 Hz PC 2019-07-17 FF	5.1 Hz 2019-08-16T08:34:23+02:00 2019-07-17	8012 MB
	10 Hz / 50	0.2 Hz	4.6 Hz 2019-08-16T08:34:23+02:00	5.4 Hz 2019-08-16T08:34:23+02:00 2019-07-17	4.8 Hz PC 2019-07-17 FF	4.9 Hz 2019-08-16T08:34:23+02:00 2019-07-17	7984 MB
	10 Hz / 50	0.2 Hz	0.2 Hz 2019-08-16T08:34:23+02:00	0.2 Hz 2019-08-16T08:34:23+02:00 2019-07-17	0.2 Hz PC 2019-07-17 FF	0.0 Hz 2019-08-16T08:34:23+02:00 2019-07-17	6905 MB
	10 Hz / 50	0.2 Hz	3.6 Hz 2019-08-16T08:34:23+02:00	3.3 Hz 2019-08-16T08:34:23+02:00 2019-07-17	3.4 Hz PC 2019-07-17 FF	3.6 Hz 2019-08-16T08:34:23+02:00 2019-07-17	7984 MB
Q2	10 Hz / 50	0.2 Hz	6.5 Hz 2019-08-16T08:34:23+02:00	4.7 Hz 2019-08-16T08:34:23+02:00 2019-07-17	3.9 Hz PC 2019-07-17 FF	4.5 Hz 2019-08-16T08:34:23+02:00 2019-07-17	8576 MB
	10 Hz / 50	0.2 Hz	3.7 Hz 2019-08-16T08:34:23+02:00	4.2 Hz 2019-08-16T08:34:23+02:00 2019-07-17	3.8 Hz PC 2019-07-17 FF	3.8 Hz 2019-08-16T08:34:23+02:00 2019-07-17	10659 M
	10 Hz / 50	0.2 Hz	6.9 Hz 2019-08-16T08:34:23+02:00	4.7 Hz 2019-08-16T08:34:23+02:00 2019-07-17	4.7 Hz PC 2019-07-17 FF	5.1 Hz 2019-08-16T08:34:23+02:00 2019-07-17	8574 MB
	10 Hz / 50	0.2 Hz	6.7 Hz 2019-08-16T08:34:23+02:00	4.6 Hz 2019-08-16T08:34:23+02:00 2019-07-17	4.7 Hz PC 2019-07-17 FF	5.2 Hz 2019-08-16T08:34:23+02:00 2019-07-17	8574 MB
Q3	10 Hz / 50	0.3 Hz	4.6 Hz 2019-08-16T08:34:23+02:00	4.6 Hz 2019-08-16T08:34:23+02:00 2019-07-17	4.8 Hz PC 2019-07-17 FF	4.5 Hz 2019-08-16T08:34:23+02:00 2019-07-17	8012 MB
	10 Hz / 50	0.2 Hz	5.1 Hz 2019-08-16T08:34:23+02:00	4.5 Hz 2019-08-16T08:34:23+02:00 2019-07-17	5.4 Hz PC 2019-07-17 FF	5.0 Hz 2019-08-16T08:34:23+02:00 2019-07-17	7984 MB
	10 Hz / 50	0.2 Hz	0.2 Hz 2019-08-16T08:34:23+02:00	0.2 Hz 2019-08-16T08:34:23+02:00 2019-07-17	0.2 Hz PC 2019-07-17 FF	0.0 Hz 2019-08-16T08:34:23+02:00 2019-07-17	6905 MB
	10 Hz / 50	0.2 Hz	4.5 Hz 2019-08-16T08:34:23+02:00	3.5 Hz 2019-08-16T08:34:23+02:00 2019-07-17	2.0 Hz PC 2019-07-17 FF	4.2 Hz 2019-08-16T08:34:23+02:00 2019-07-17	7984 MB
Q4	10 Hz / 50	0.2 Hz	5.2 Hz 2019-08-16T08:34:23+02:00	4.2 Hz 2019-08-16T08:34:23+02:00 2019-07-17	4.5 Hz PC 2019-07-17 FF	4.4 Hz 2019-08-16T08:34:23+02:00 2019-07-17	8576 MB
	10 Hz / 50	0.2 Hz	3.8 Hz 2019-08-16T08:34:23+02:00	3.8 Hz 2019-08-16T08:34:23+02:00 2019-07-17	4.5 Hz PC 2019-07-17 FF	4.1 Hz 2019-08-16T08:34:23+02:00 2019-07-17	10659 M
	10 Hz / 50	0.2 Hz	5.4 Hz 2019-08-16T08:34:23+02:00	5.4 Hz 2019-08-16T08:34:23+02:00 2019-07-17	4.8 Hz PC 2019-07-17 FF	4.9 Hz 2019-08-16T08:34:23+02:00 2019-07-17	8574 MB
	10 Hz / 50	0.2 Hz	4.7 Hz 2019-08-16T08:34:23+02:00	3.9 Hz 2019-08-16T08:34:23+02:00 2019-07-17	4.4 Hz PC 2019-07-17 FF	5.0 Hz 2019-08-16T08:34:23+02:00 2019-07-17	8574 MB

Raw 14, 104801.4 ms, **to GUI:** 0.0 Hz

Gain 14, 104820.3 ms, **to GUI:** 0.0 Hz

Corrected 14, 113611.6 ms, **to GUI:** 5.0 Hz

Enable OffsetCorrectionAGIPD ✓

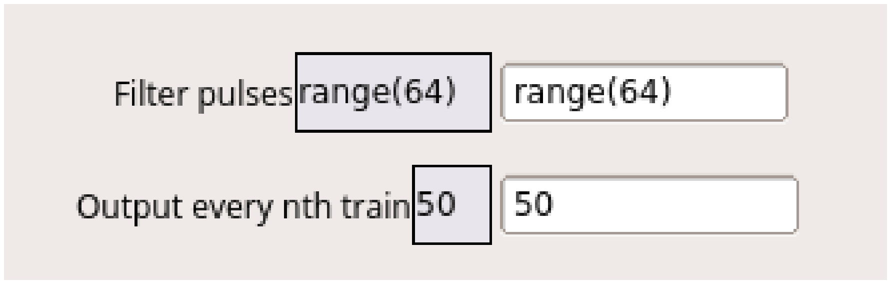
Enable RelativeGainCorrection ☐

Enable MGainLevelerAGIPD ☐

Fig. 11.5: Calibration pipeline overview scene of AGIPD. Other detectors look similar.

- the the output "every nth train" setting correct. It defaults to 50 on a restart and once the pipeline is initialized should be set to 10.
- Is the pulse filter setting correct. It accepts range expressions, but also lists. To increase useful data, e.g. in an alternating pulse pattern, add those data indices you would like forwarded to it, e.g. "0,2,4,5,6", or use an range with stride, e.g. "range(0,250,2)".
- If too aggressive settings have been chose, the pipeline updates will decouple and train matching rarely occurs at the final APPENDER. This can be identified by the following:
  - the latency of the APPENDER devices starts piling up or varying greatly
  - the update rates of pipeline components do not match the rate at the SPLITTER anymore, or have large variations.

  In either case, the pipeline should recover after a few seconds of streaming data with known safe settings, e.g. outputting every 10th train, and a filter of range(128).



Filter pulses range(64) range(64)

Output every nth train 50 50

Fig. 11.6: Performance relevant settings on the manager scene.

- If a user wants single module data with low latency, you can connect to a singular modules data stream directly.

There is a procedure here: *Connecting a Single Module's Raw Data to the Bridge*.

11.7 GPU related

- The GPU memory fields shows 0 MB free and are red. This is fine after a restart. They will update once the detector pushes data.
- The GPU memory still is not at zero but very low. This usually indicates that parts of the pipeline are not fully functional. Check the logs on the respective servers for any errors and check that it is connected to the correct DAQ output.

CHAPTER 12

Debugging the CalibrationDbRemote Devices

The so-called *calibrationDbRemote* devices are the access points to the calibration database, both for the online and the offline services. Retrieving and injecting constants through them are performed using *ZMQ* requests. The devices themselves run in a Karabo environment.

For online calibration one should check the procedures at *General Online Calibration Troubleshooting* for how to identify and debug problems with these devices.

The offline devices run under the *xcal* account on *max-exfl016* in a Karabo environment installed in the */scratch/xcal/karabo* folder. This installation is maintained by ITDM. Note the the folder is only accessible from *max-exfl016*; it is not shared across Maxwell hosts.

12.1 Inspecting Logs

To inspect logs, log onto *xcal@max-exfl016* and *tail* the log file of the device server:

```
ssh xcal@max-exfl016
tail -500f /scratch/xcal/karabo/var/log/pythonserver_max_web_rcal_0/current
```

This should result in an output similar to:

```
INFO  MAX_WEB_DATA/DM/CAL_REMOTE_5  : Waiting on data
INFO  MAX_WEB_DATA/DM/CAL_REMOTE_5  : Saved file cal.1567178413.0473201.h5 at xfel/
↪cal/agipd-type/agipd_siv1_agipdv11_m400/
INFO  MAX_WEB_DATA/DM/CAL_REMOTE_5  : Start writing to Calibration Catalogue...
INFO  MAX_WEB_DATA/DM/CAL_REMOTE_5  : Finnish writing to Calibration Catalogue...
INFO  MAX_WEB_DATA/DM/CAL_REMOTE_5  : Registered cal.1567178413.0473201.h5␣
↪successfully
```

for when parameters are being written to the database, or:

```
INFO  MAX_WEB_DATA/DM/CAL_REMOTE_0  : Waiting on data
INFO  MAX_WEB_DATA/DM/CAL_REMOTE_0  : Start searching in Calibration Catalogue...
```

(continues on next page)

(continued from previous page)

```
INFO  MAX_WEB_DATA/DM/CAL_REMOTE_0  : Got calibration: Offset (AGIPD1M1), meta_only:␣
↪True
INFO  MAX_WEB_DATA/DM/CAL_REMOTE_0  : Waiting on data
INFO  MAX_WEB_DATA/DM/CAL_REMOTE_0  : Start searching in Calibration Catalogue...
INFO  MAX_WEB_DATA/DM/CAL_REMOTE_0  : Got calibration: BadPixelsDarkCCD (FastCCD1),␣
↪meta_only: False
```

when retrieving parameters from the database.

There are a total of 30 devices acting as access points, and most calibration routines will randomly pick one of these and retry contacting another one if it is not available.

12.2 Error Scenarios

The following error scenarios have been known to occur:

- **Calibration database devices do not respond to queries**. This can be due to a temporary overload of the server, such that the processes running the access points were not able to allocate sufficient memory. This should not happen anymore if meta_data only retrieval mode is used for larger parameters, like those of AGIPD and LPD. It can be diagnosed by the log outputs not updating even though querying jobs have been launched.

  It can be solved by restarting the calibrationDbRemote server:

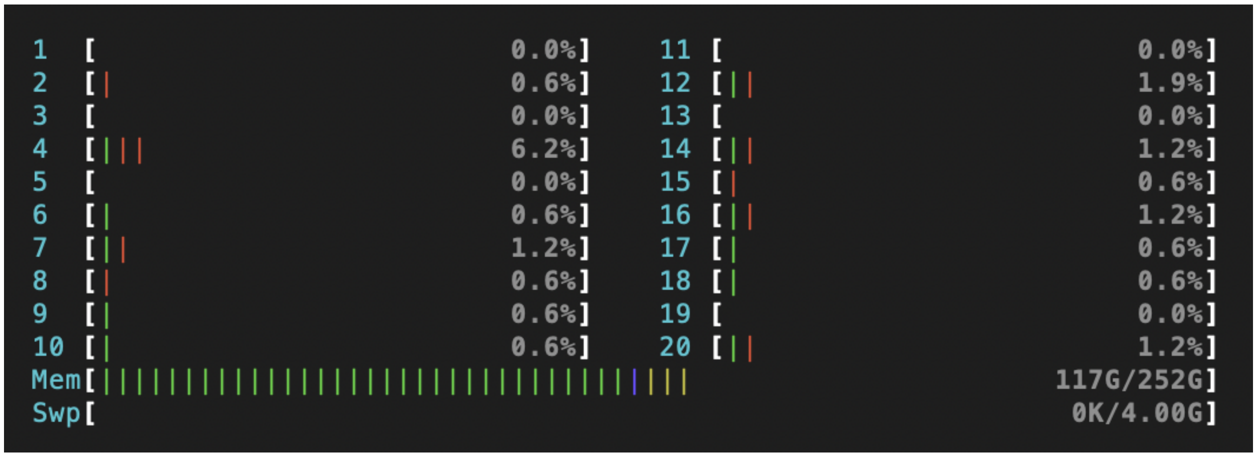
  ```
  source /scratch/xcal/karabo/activate
  karabo-kill -k pythonserver_max_web_rcal_0
  ```

  Note that this might lead to errors on calibration jobs currently in flight, when the restart occurs. All devices will autostart after the server restart. The procedure takes about 30 seconds until the system will be running again.

  To diagnose if the problems occured through requesting large constants, while not using meta_only mode, check memory consumption via *htop*:

      htop

  and assure that no swapping had occurred.



```
1  [                          0.0%]   11 [                          0.0%]
2  [|                         0.6%]   12 [||                        1.9%]
3  [                          0.0%]   13 [                          0.0%]
4  [|||                       6.2%]   14 [||                        1.2%]
5  [                          0.0%]   15 [|                         0.6%]
6  [|                         0.6%]   16 [||                        1.2%]
7  [||                        1.2%]   17 [|                         0.6%]
8  [|                         0.6%]   18 [|                         0.6%]
9  [|                         0.6%]   19 [                          0.0%]
10 [|                         0.6%]   20 [||                        1.2%]
Mem[||||||||||||||||||||||||||||||||||||                   117G/252G]
Swp[                                                          0K/4.00G]
```

Fig. 12.1: Example output of *htop* command. The swap bar should be near zero.

- **Unusually high number of request, of the same parameters** can occur if ZMQ requests with improper timeout error handling are being made. This will lead to recurring log entries requesting the same parameters. Current production code has long enough timeouts set, but legacy code is known to run into this scenario. To fix it, first check if you can find processes that might be responsible for the requests:

```
squeue -u xcal
```

and stop these, e.g. by using *scancel*. The restart the calibration database devices as mentioned above. Make sure to increase timeouts on calling code before initiating new requests.

- **Reports show ‘Version already taken‘ error**. This is not really a failure scenario, but rather expected behaviour. A parameter with exactly the same start date and operating conditions has previously been injected into the calibration database. The database is now refusing to overwrite it. Check if you are characterizing the correct data with the correct conditions.

CHAPTER 13

Indices and tables

- genindex
- modindex
- search