



# Hardware Triggered Scanning: GDA and DAWN

**Philip Taylor, Emma Arandjelovic**  
**Observatory Sciences Limited**



# Software Stack: Reminder



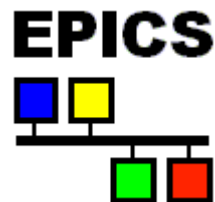
**Data Analysis WorkbeNch**  
- Analysis and visualization



**Generic Data Acquisition**  
- Experiment setup and supervision



**Malcolm**  
- Scan configuration

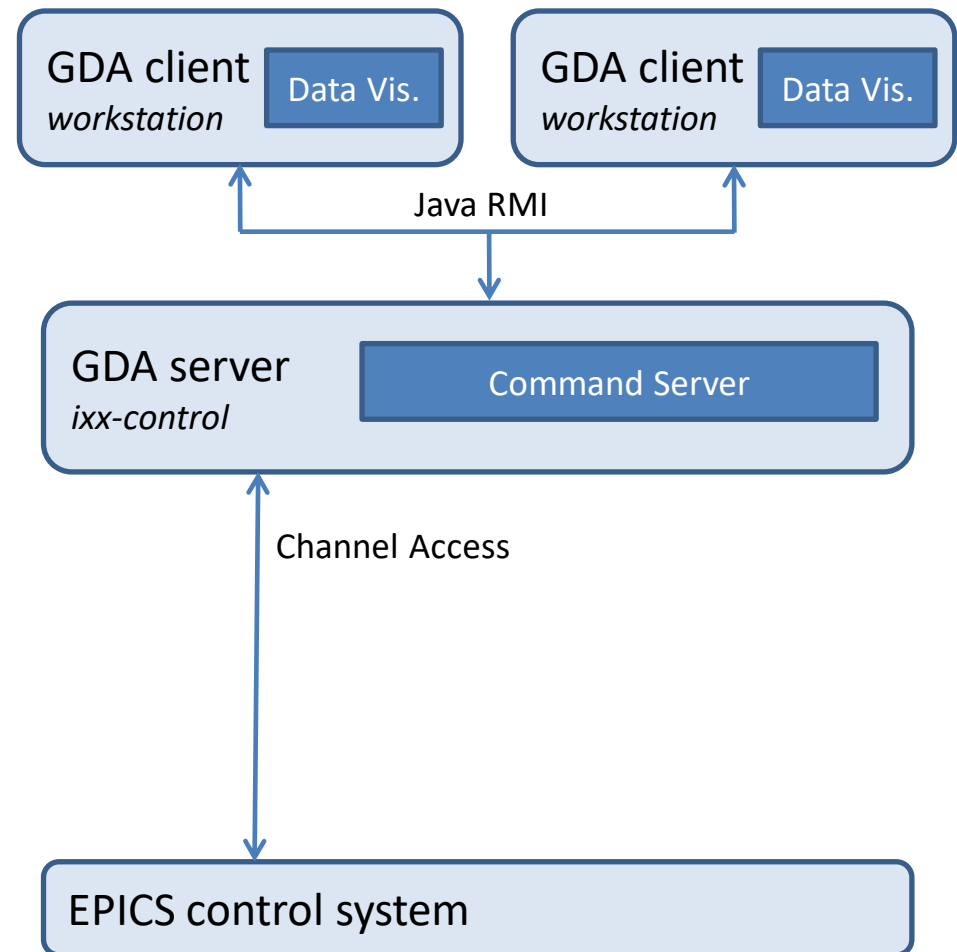


**Experimental Physics &  
Industrial Control System**  
- Low level control of hardware



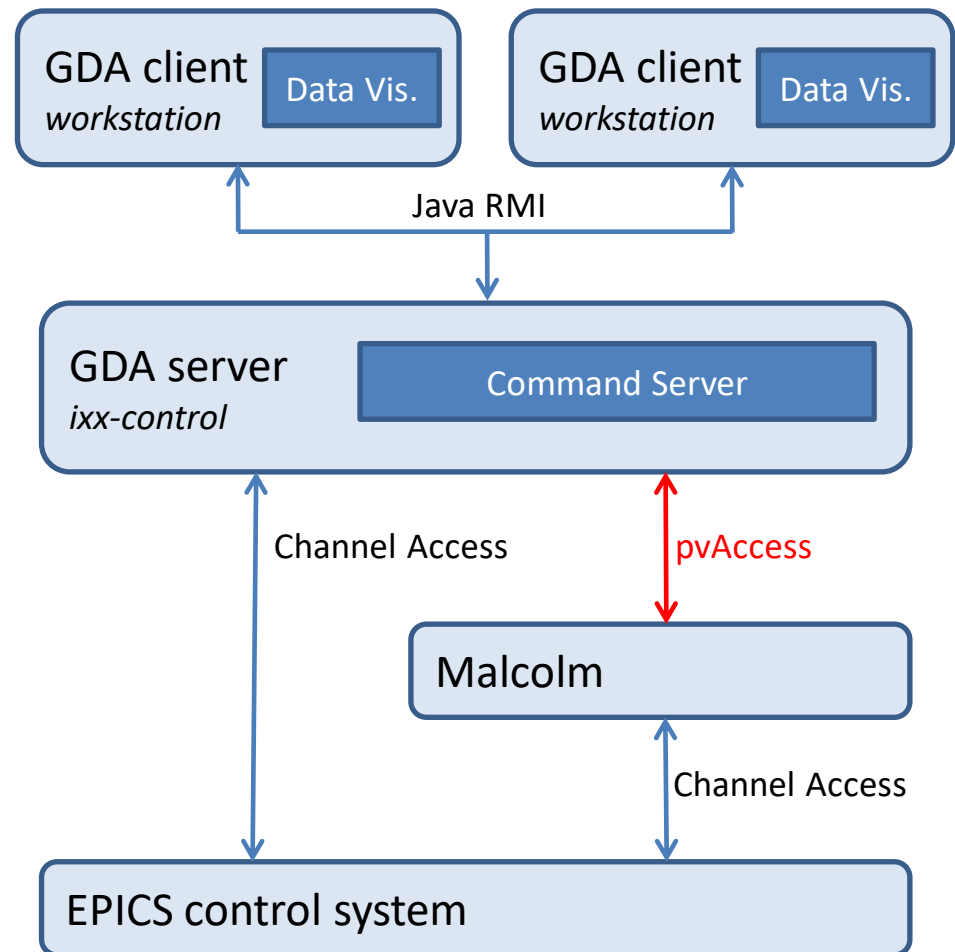
# Overview

- Main beamline user interface
- Presents a science based view of experiments
- Client-server design
- *Command Server* provides a Jython interpreter
- Client provides live data visualization



# Overview

- Main beamline user interface
- Presents a science based view of experiments
- Client-server design
- *Command Server* provides a Jython interpreter
- Client provides live data visualization
- GDA<->Malcolm interface developed to integrate scanning framework



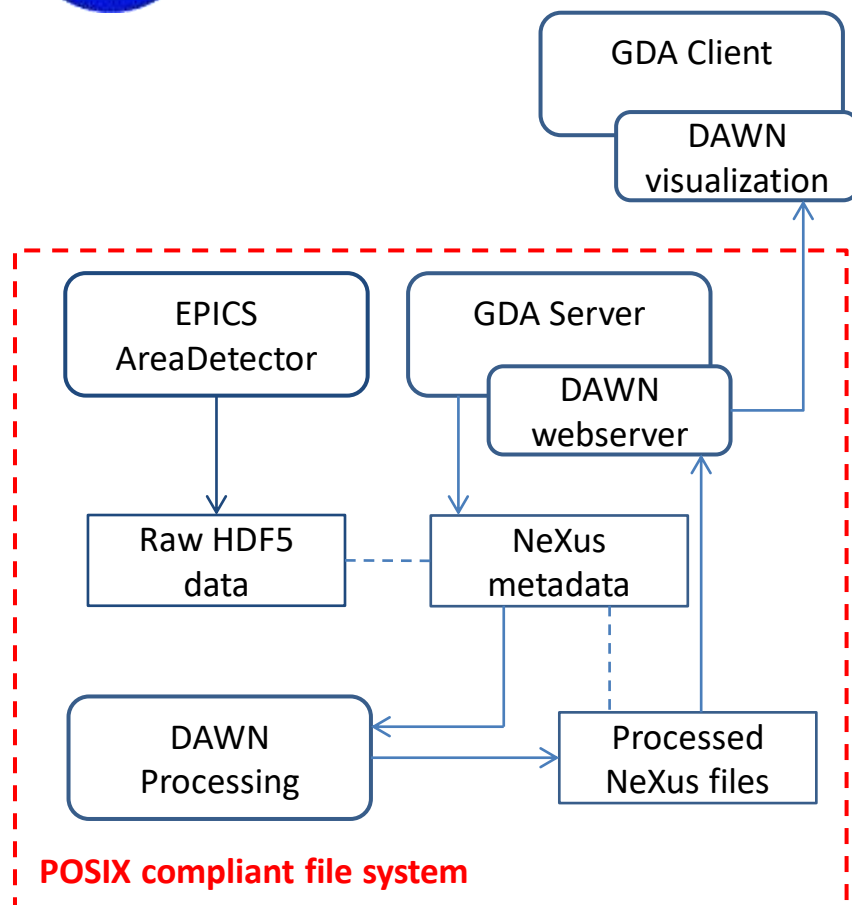


# Overview

- Data Analysis Workbench
- Sister project to GDA (Eclipse based)
- Provides visualization components to GDA client
- Includes generic data processing tools
- Data processing pipelines can be customized
- Supports multiple file formats, including NeXus
- See <https://dawnsci.org/>



# Data Flow

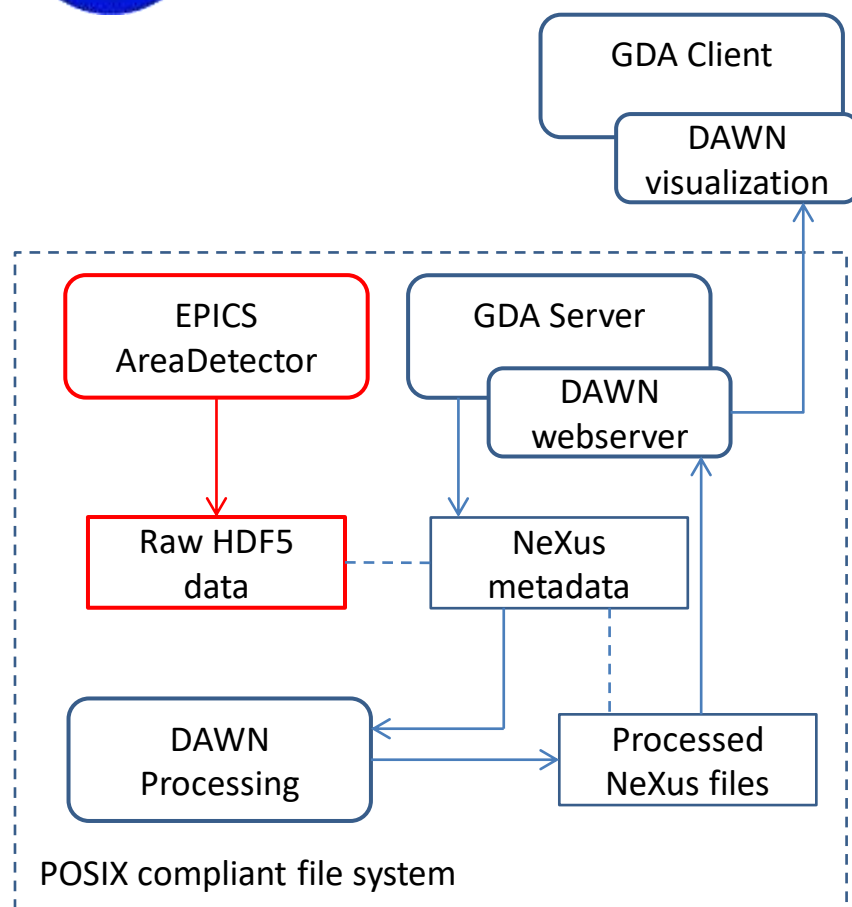


Data written to a POSIX file system

- Enables SWMR mode



# Data Flow



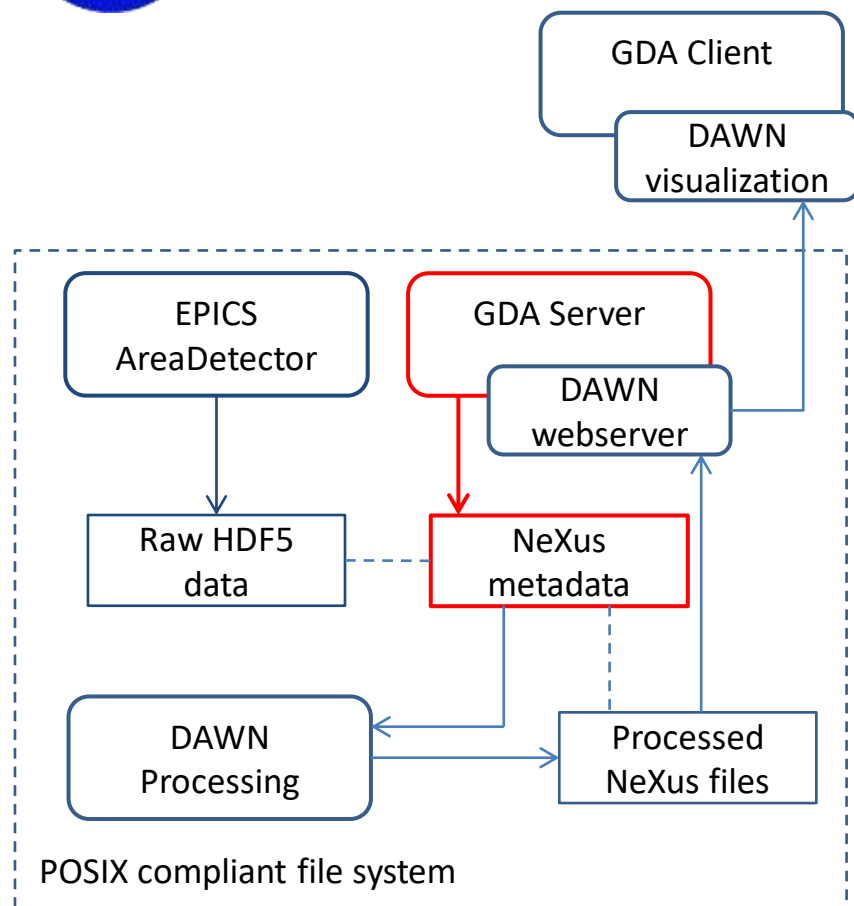
Data written to a POSIX file system

➤ Enables SWMR mode

1. AreaDetector writes raw HDF data



# Data Flow



Data written to a POSIX file system

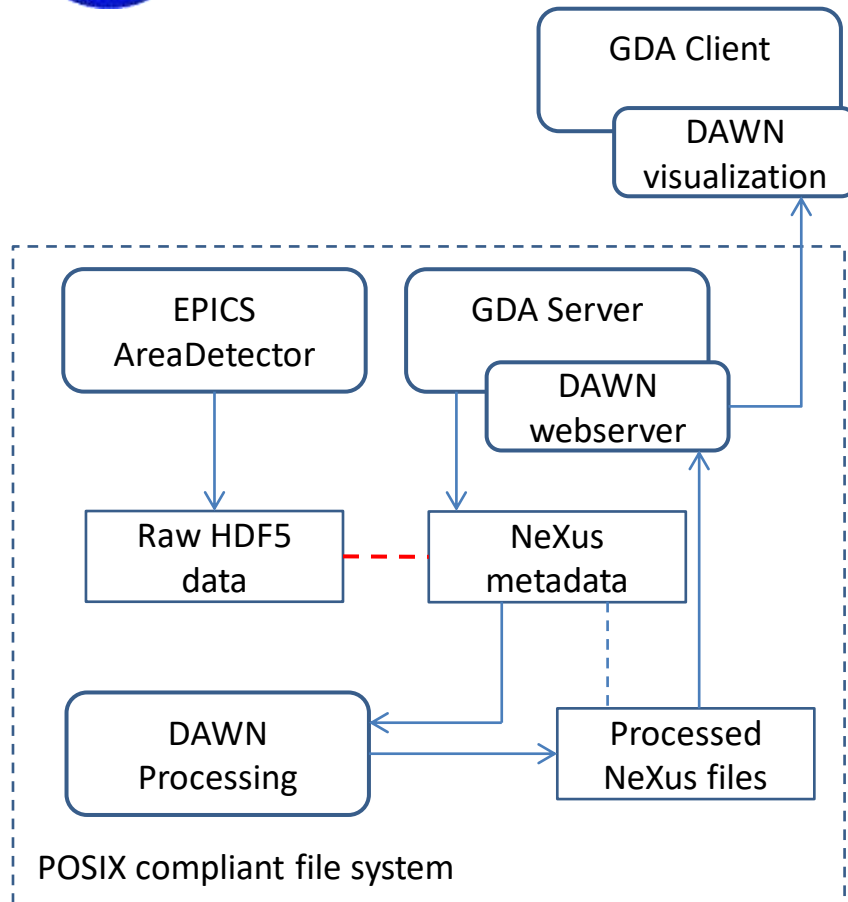
➤ Enables SWMR mode

1. AreaDetector writes raw HDF data
2. GDA writes metadata to another HDF file





# Data Flow



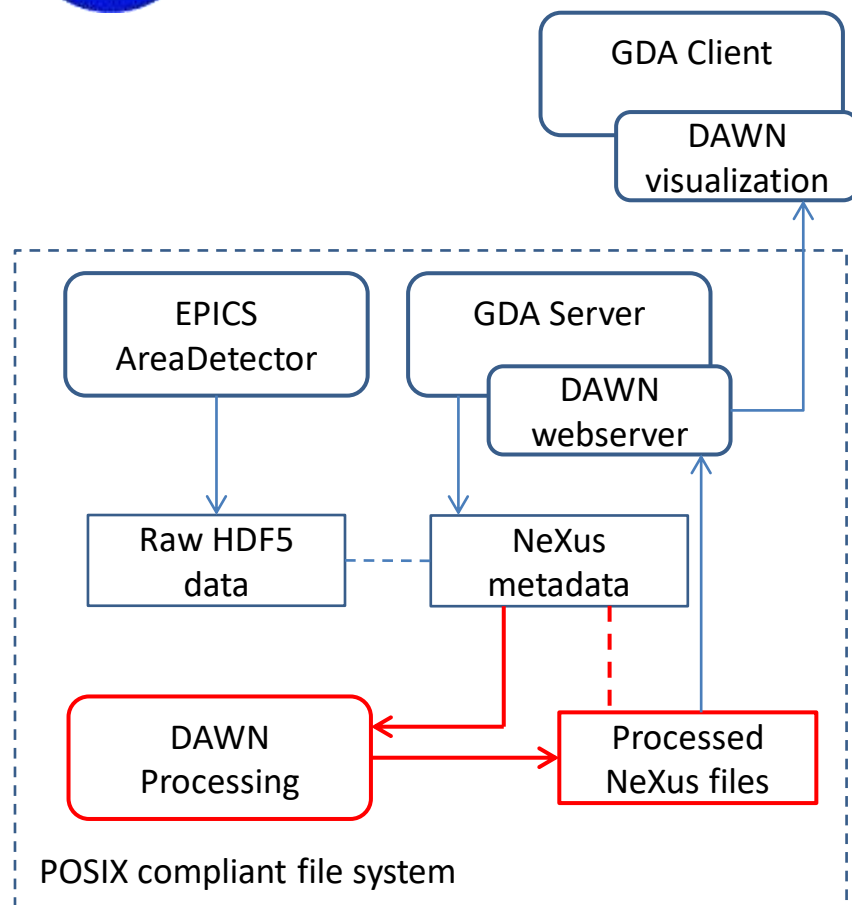
Data written to a POSIX file system

➤ Enables SWMR mode

1. AreaDetector writes raw HDF data
2. GDA writes metadata to another HDF file
3. The two HDF files are linked



# Data Flow



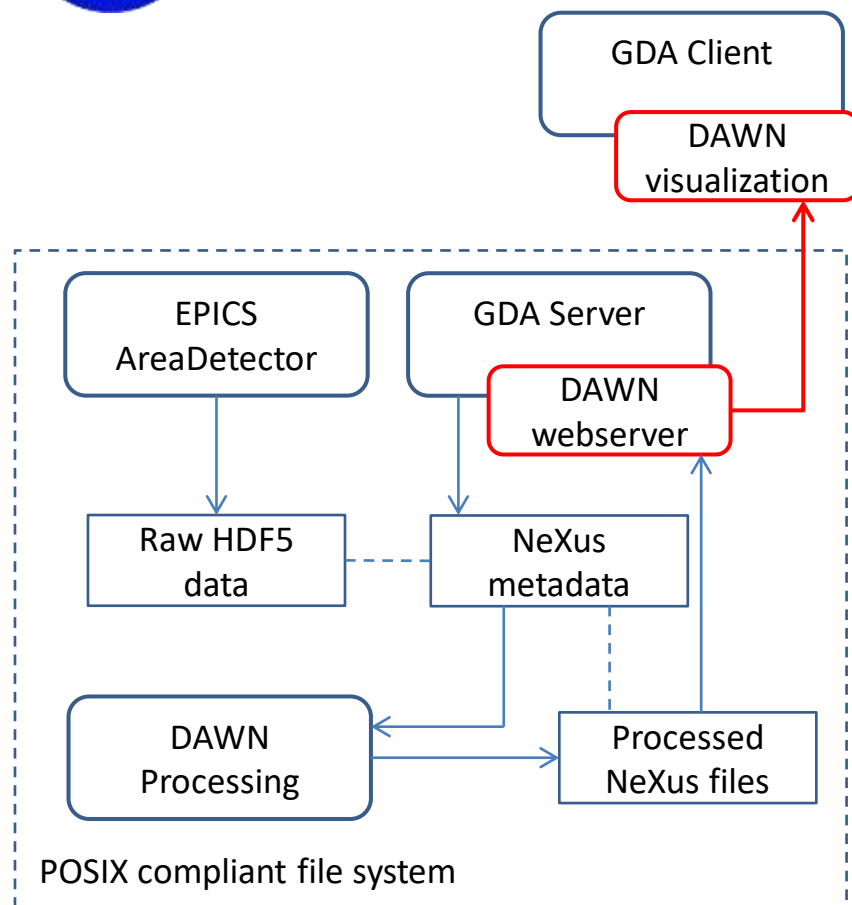
Data written to a POSIX file system

➤ Enables SWMR mode

1. AreaDetector writes raw HDF data
2. GDA writes metadata to another HDF file
3. The two HDF files are linked
4. DAWN processing reads both files



# Data Flow



Data written to a POSIX file system

➤ Enables SWMR mode

1. AreaDetector writes raw HDF data
2. GDA writes metadata to another HDF file
3. The two HDF files are linked
4. DAWN processing reads both file
5. DAWN webserver makes the raw and processed files available to the GDA client outside the POSIX file system



# Interface to Malcolm

- Malcolm<->GDA interface uses [pvAccess](#) (EPICS V4)
- This supports structured data
- Each Malcolm 'device' in GDA maps onto a block:

`$(gda_config)/servers/main/live/malcolm_real.xml`

```
<bean id="realMalcolmModel" class="org.eclipse.scanning.api.device.models.MalcolmModel">  
  <property name="name" value="{PV_PREFIX}-ML-SCAN-01"/>  
  <property name="exposureTime" value="0.01"/>  
</bean>
```

- This defines the MRI (Malcolm Resource Identifier) for the top level scan block



# Interface Customization

`$(gda_config)/clients/main/_common/mapping.xml`

Axis name labels:

```
<!-- Stage axis names defined as String beans for convenience and to avoid repetition of the same literal value -->
<bean id="x_axis_name" class="java.lang.String" factory-method="valueOf"><constructor-arg value="stagea"/></bean>
<bean id="y_axis_name" class="java.lang.String" factory-method="valueOf"><constructor-arg value="stagex"/></bean>
```

Outer scan axes:

```
</property>
<property name="scanDefinition">
  <bean class="uk.ac.diamond.daq.mapping.impl.MappingScanDefinition">
    <property name="outerScannables">
      <list>
        <bean class="uk.ac.diamond.daq.mapping.impl.ScanPathModelWrapper">
          <constructor-arg name="name" value="stagez" />
          <constructor-arg name="model"><null /></constructor-arg>
          <constructor-arg name="includeInScan" value="false" />
        </bean>
      </list>
    </property>
  </bean>
</property>
```



# GDA perspectives

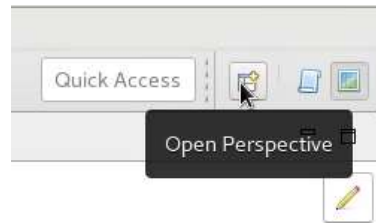
GDA has three main perspectives you'll use with Malcolm:

- Mapping
- DataVis
  - Used both of these in Part 1
- Processing

*A perspective is a collection of related views*

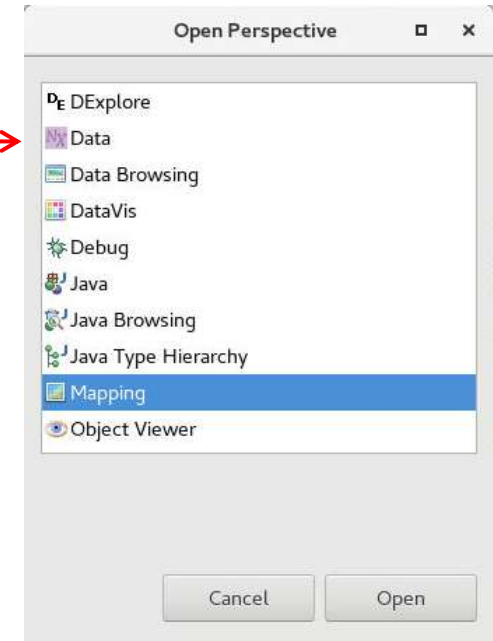


# Mapping Perspective



Mapping perspective used for:

- Setting up scan 'map'
- Configuring sample metadata
- Start/stop/pause controls
- Monitoring progress
- Viewing results





# Mapping Experiment Setup

1. Configure any outer scan axes

Mapping Experiment Setup

Other Scan Axes

☐ stage\_z

Detectors

☐ GigE Camera 0.004

☒ Malcolm real motors 0.004

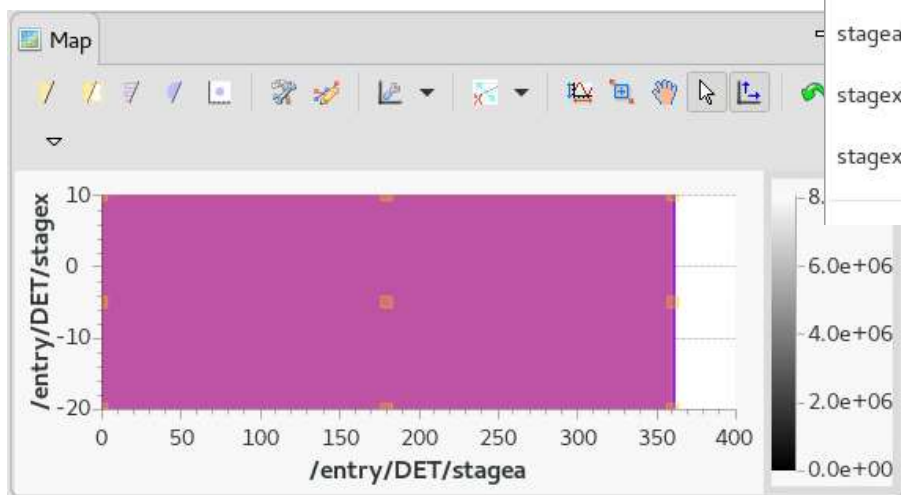
Region shape: Rectangle Scan path: Grid

stagea Start 0 mm stagea Points 180

stagea Stop 360 mm stagex Points 180

stagex Start -20 mm Snake ☒

stagex Stop 10 mm Continuous ☒







# Mapping Experiment Setup

1. Configure any outer scan axes
2. Select the Malcolm device and enter desired exposure time

A screenshot of the "Mapping Experiment Setup" window. The window has a title bar with a close button. It contains several sections: "Other Scan Axes" with a checkbox for "stage\_z" and a text field; "Detectors" with checkboxes for "GigE Camera" and "Malcolm real motors", each followed by a text field for exposure time. The "Malcolm real motors" section is highlighted with a red rectangle. Below this, there are dropdown menus for "Region shape" (set to "Rectangle") and "Scan path" (set to "Grid"). At the bottom, there are input fields for "stagea Start", "stagea Stop", "stagea Points", "stagex Start", "stagex Stop", "stagex Points", and checkboxes for "Snake" and "Continuous".

Mapping Experiment Setup

Other Scan Axes

☐ stage\_z

Detectors

☐ GigE Camera 0.004

☒ Malcolm real motors 0.004

Region shape: Rectangle

Scan path: Grid

stagea Start 0 mm

stagea Stop 360 mm

stagea Points 180

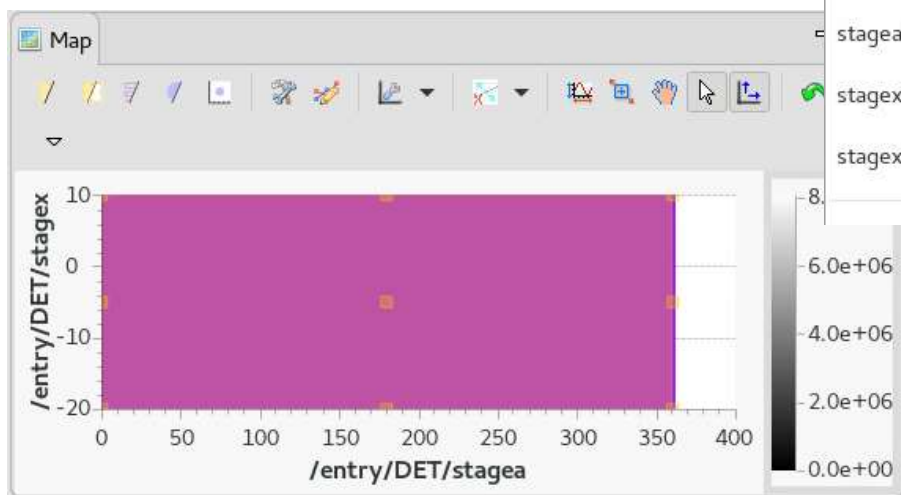
stagex Start -20 mm

stagex Stop 10 mm

stagex Points 180

Snake ☒

Continuous ☒





# Mapping Experiment Setup

1. Configure any outer scan axes
2. Select the Malcolm device and enter desired exposure time
3. Choose desired region shape and scan path

Mapping Experiment Setup

Other Scan Axes

☐ stage\_z

Detectors

☐ GigE Camera 0.004

☒ Malcolm real motors 0.004

Region shape: Rectangle

Scan path: Grid

stagea Start 0 mm

stagea Stop 360 mm

stagex Start -20 mm

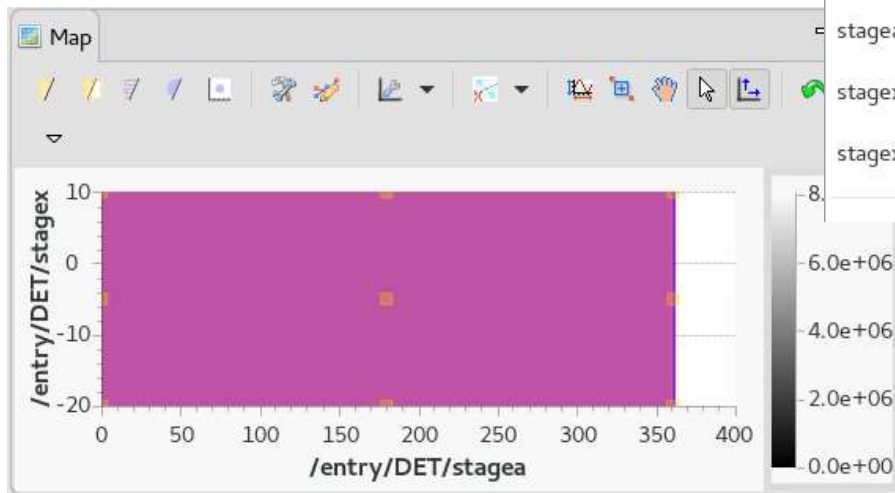
stagex Stop 10 mm

stagea Points 180

stagex Points 180

Snake ☒

Continuous ☒





# Mapping Experiment Setup

1. Configure any outer scan axes
2. Select the Malcolm device and enter desired exposure time
3. Choose desired region shape and scan path

Mapping Experiment Setup

Other Scan Axes

☐ stage\_z

Detectors

☐ GigE Camera 0.004

☒ Malcolm real motors 0.004

Region shape: Rectangle Scan path: Grid

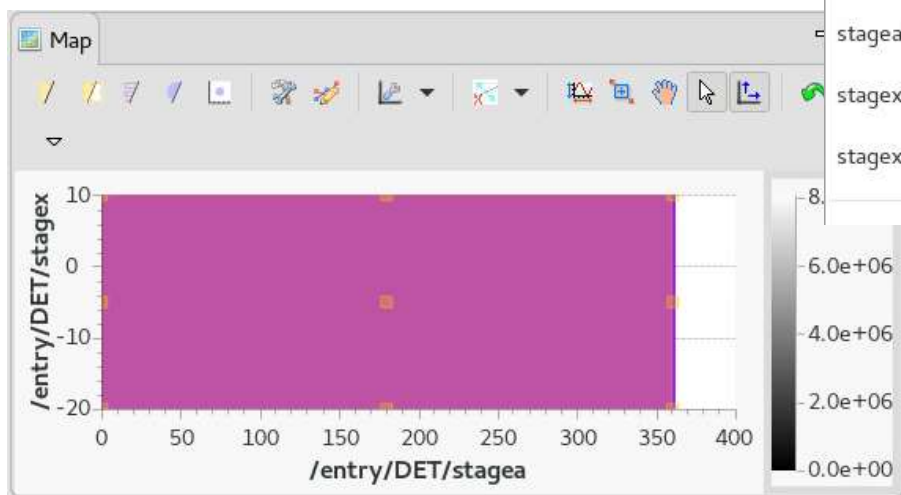
stagea Start 0 mm stagea Points 180

stagea Stop 360 mm

stagex Start -20 mm stagex Points 180

stagex Stop 10 mm

Snake ☒ Continuous ☒



4. Configure the map region
  - a) Manually



# Mapping Experiment Setup

1. Configure any outer scan axes
2. Select the Malcolm device and enter desired exposure time
3. Choose desired region shape and scan path

Mapping Experiment Setup


Other Scan Axes

☐ stage\_z

Detectors

☐ GigE Camera 0.004

☒ Malcolm real motors 0.004

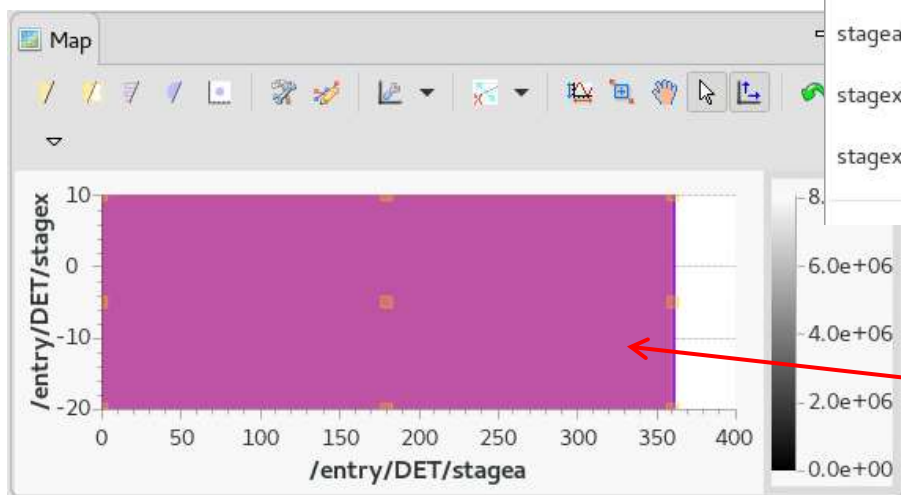
Region shape: Rectangle  Scan path: Grid

stagea Start 0 mm stagea Points 180

stagea Stop 360 mm stagea Points 180

stagex Start -20 mm Snake ☒

stagex Stop 10 mm Continuous ☒



4. Configure the map region
  - a) Manually, or
  - b) By drawing desired shape



# Mapping Experiment Setup

## 5. Choose a processing template

The screenshot shows a software interface for mapping experiment setup. At the top, under the "Processing" section, there is a checkbox labeled "mean\_integrate\_working.nxs" which is checked. To the right of this checkbox are three buttons: "Add Processing...", "Configure...", and "Delete". A red arrow points from the text "5. Choose a processing template" to the "Add Processing..." button. Below the processing section, there is a status bar displaying "Map points: 32400", "Exposure time: 00:02:10", and "Smallest steps: X = 2.000; Y = 0.1667; Absolute = 0.1667". Below this status bar is a "Queue Scan" button. A red arrow points from the text "6. Add scan to queue" to this button. To the right of the "Queue Scan" button is a row of three icons: a document icon, a document with a green plus sign, and a floppy disk icon. These three icons are circled in red. A red arrow points from the text "Map points calculated automatically" to the status bar. Another red arrow points from the text "Save/load/copy command to clipboard" to the floppy disk icon.

Processing

☒ mean\_integrate\_working.nxs

Add Processing...

Configure... Delete

Map points: 32400 Exposure time: 00:02:10  
Smallest steps: X = 2.000; Y = 0.1667; Absolute = 0.1667

Queue Scan

6. Add scan  
to queue

Map points calculated  
automatically

Save/load/copy  
command to clipboard



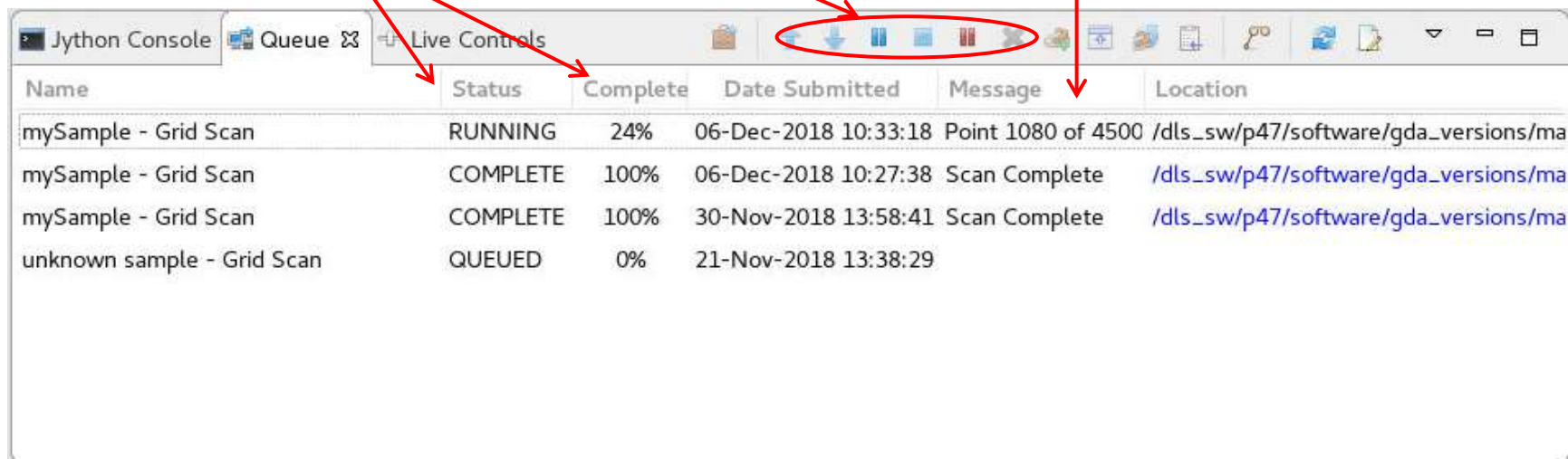
# Mapping Queue

Scan 'jobs' are put into a queue and started automatically

View status and progress

Re-order / pause / abort jobs

Error messages displayed here

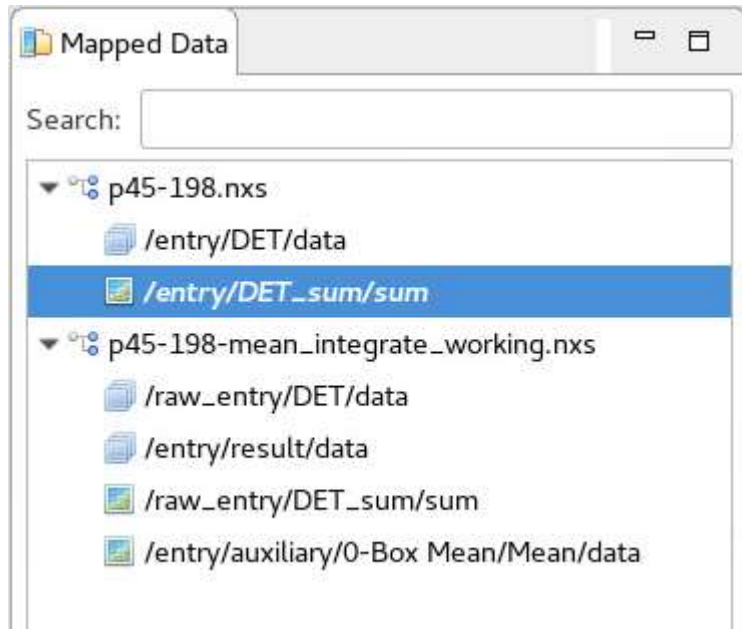


The screenshot shows a software interface with a toolbar and a table. The toolbar contains icons for queue management: a list, a refresh, a pause, a stop, and a restart. The table displays the status of various scan jobs.

Name	Status	Complete	Date Submitted	Message	Location
mySample - Grid Scan	RUNNING	24%	06-Dec-2018 10:33:18	Point 1080 of 4500	/dls_sw/p47/software/gda_versions/ma
mySample - Grid Scan	COMPLETE	100%	06-Dec-2018 10:27:38	Scan Complete	/dls_sw/p47/software/gda_versions/ma
mySample - Grid Scan	COMPLETE	100%	30-Nov-2018 13:58:41	Scan Complete	/dls_sw/p47/software/gda_versions/ma
unknown sample - Grid Scan	QUEUED	0%	21-Nov-2018 13:38:29		

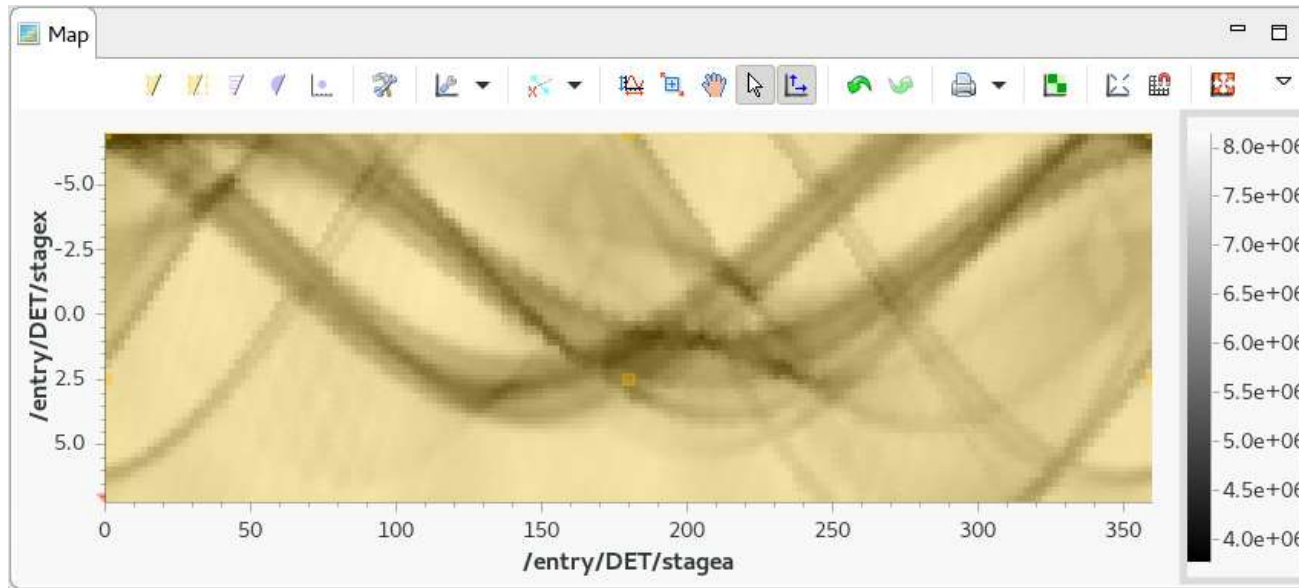


# Mapped Data View



- Lists the generated NeXus (.nxs) files
- The *sum* dataset is created by summing the pixel values over the whole image
- If processing was selected, a processed file is also generated

# Map View



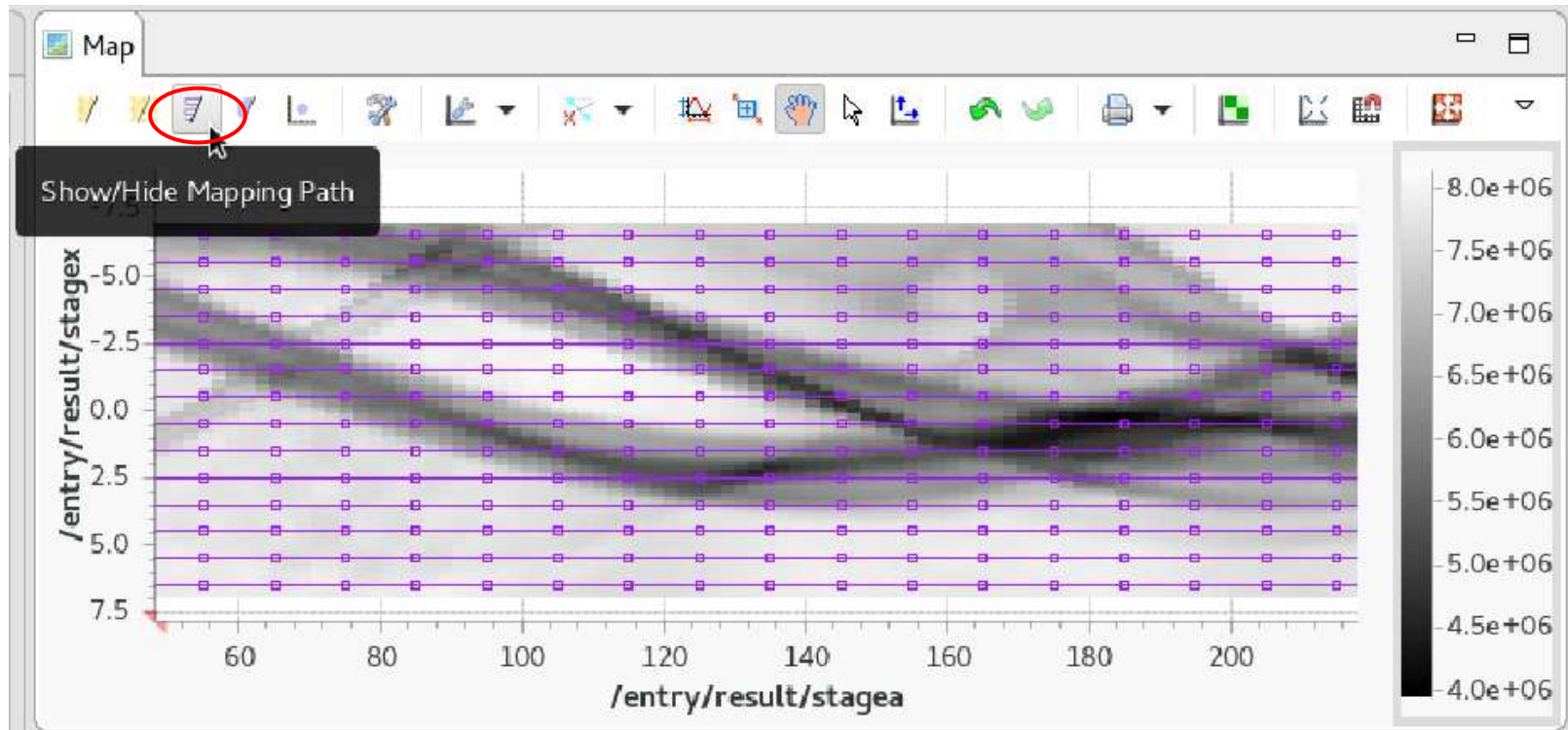
Example: representation of the *sum* (total intensity) data set

- Displays single pixel per scan point
- As the sample is rotated, the camera captures a different angle
- This results in an approximate sinewave for each sample 'stick'



# Map View

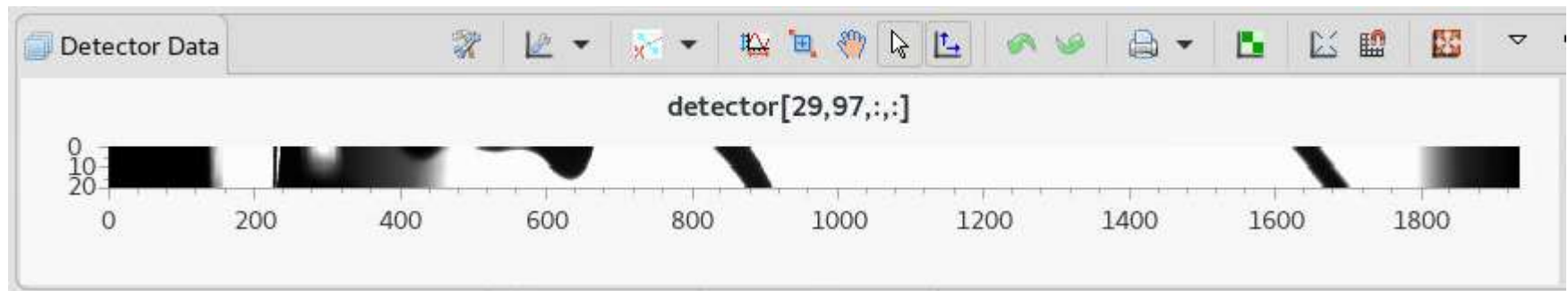
- Scans can be performed over the top of each other
- Use case: scan a small area in higher resolution





# Detector Data View

- Click a point in the map view
- Detector data view updates to show corresponding detector frame





# Live Mapping

- The mapping perspective updates raw and processed data as the scan progresses
- This is done via a *DataServer* on ixx-control (localhost on the test rig)
- Can check this using http requests in a browser:
  - <http://localhost:8690/tree/?path=<path2nxsfile>>
    - Displays the NeXus tree as XML
  - <http://localhost:8690/info/?path=<path2nxsfile>&dataset=<dataset>>
    - Displays information about the specified dataset e.g. /entry/DET/data
  - [http://localhost:8690/slice/?path=<path2nxsfile>&dataset=<dataset>&slice=\[0,0,,::\]](http://localhost:8690/slice/?path=<path2nxsfile>&dataset=<dataset>&slice=[0,0,,::])
    - Downloads data in specified slice as zip file
- When scan finished, file loaded directly from disk

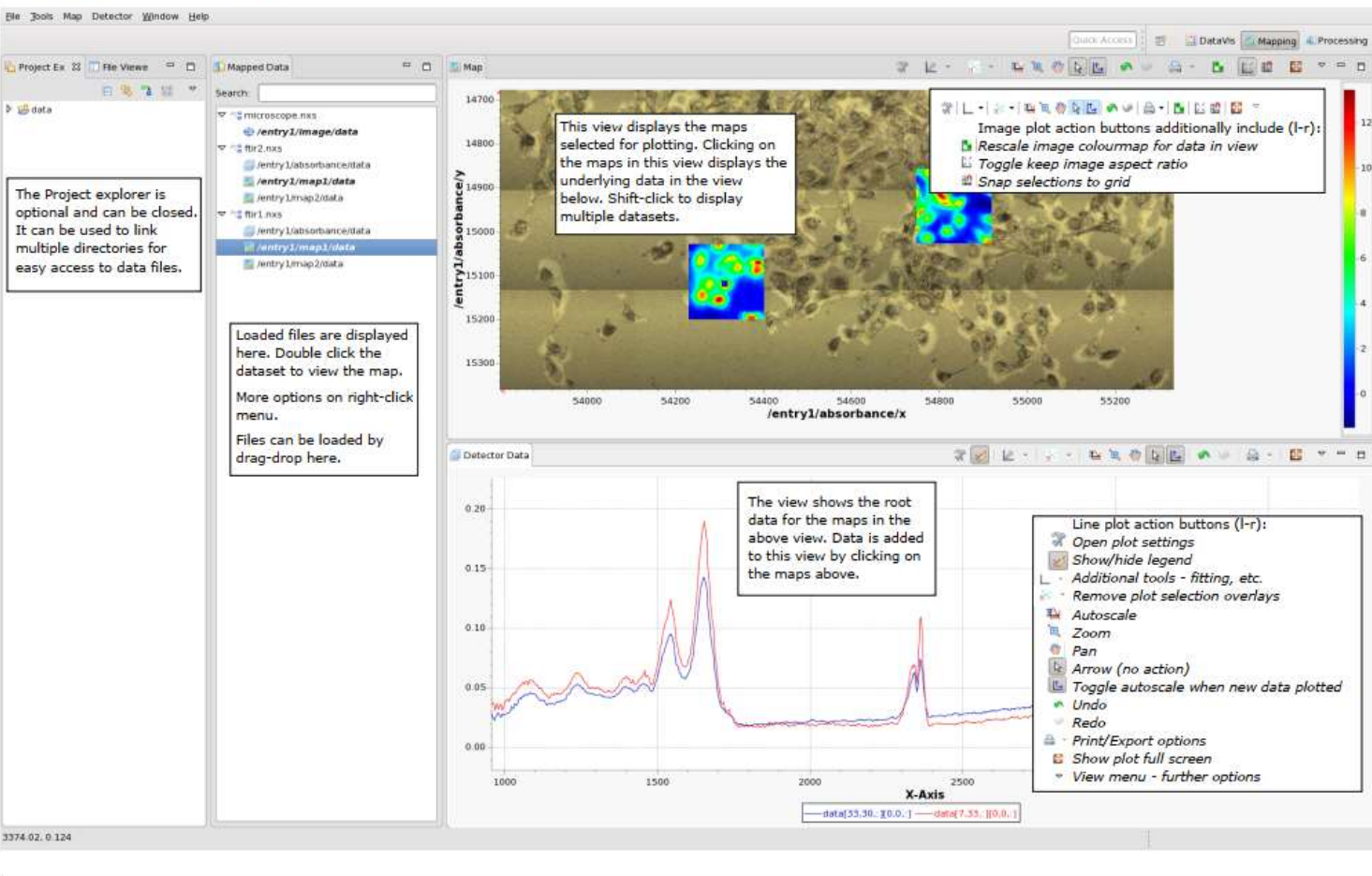
## Mapping Cheat Sheet - DAWN 2.13

Visualisation of Mapping scan data. NeXus Files Only.

Use *File/Open File...* to start

File Tools Map Detector

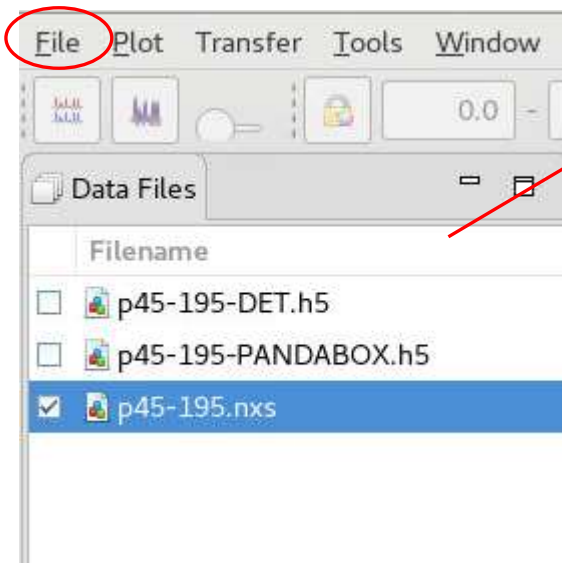
More features (Setting map transparency...) can be found in the top level menu items



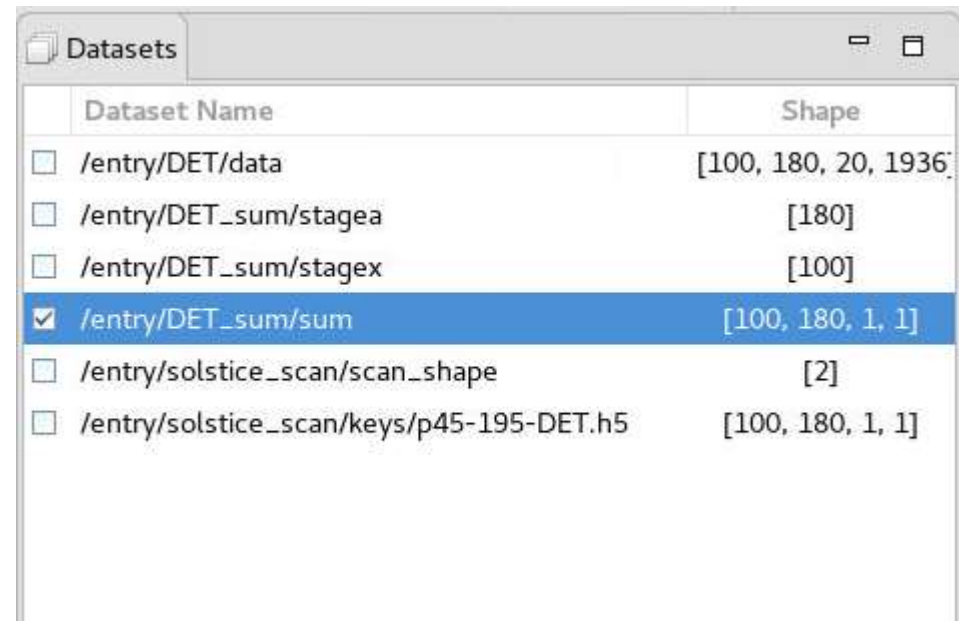


# DataVis Perspective

- Standalone perspective provided by DAWN
- Can be used to open any HDF file
- Can also be used at runtime to display additional datasets e.g. UniqueKeys



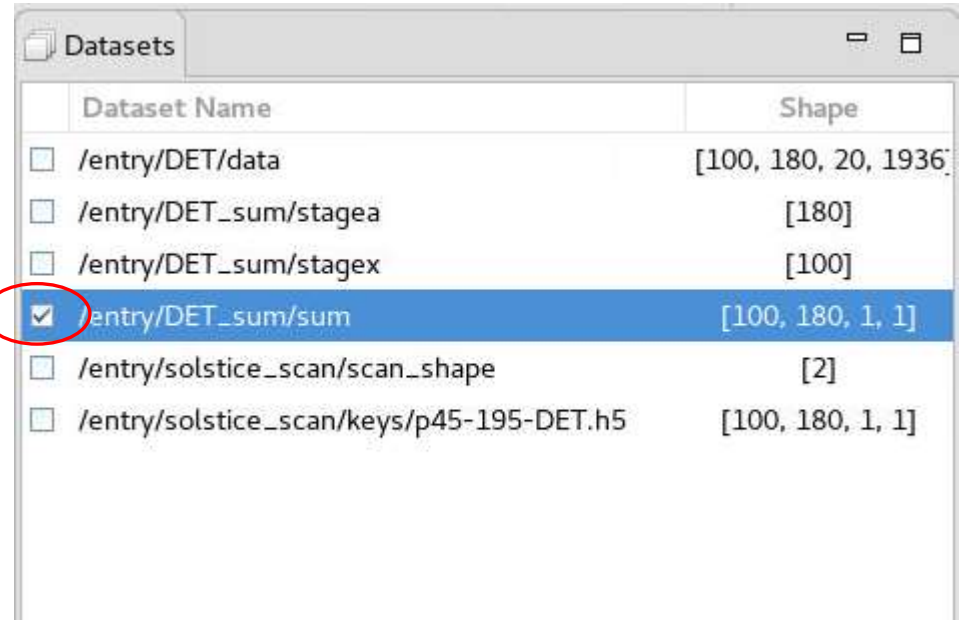
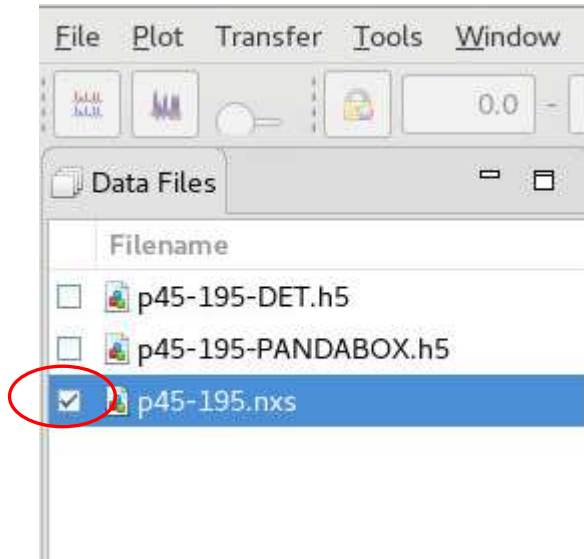
Use the *File* menu to load data files and select for viewing



*Datasets* view shows the available datasets in the selected file and their dimensions



# DataVis Perspective



Check the box on both the file AND dataset to activate it for display





# DataVis Perspective

Choose how to display the selected dataset from the dropdown

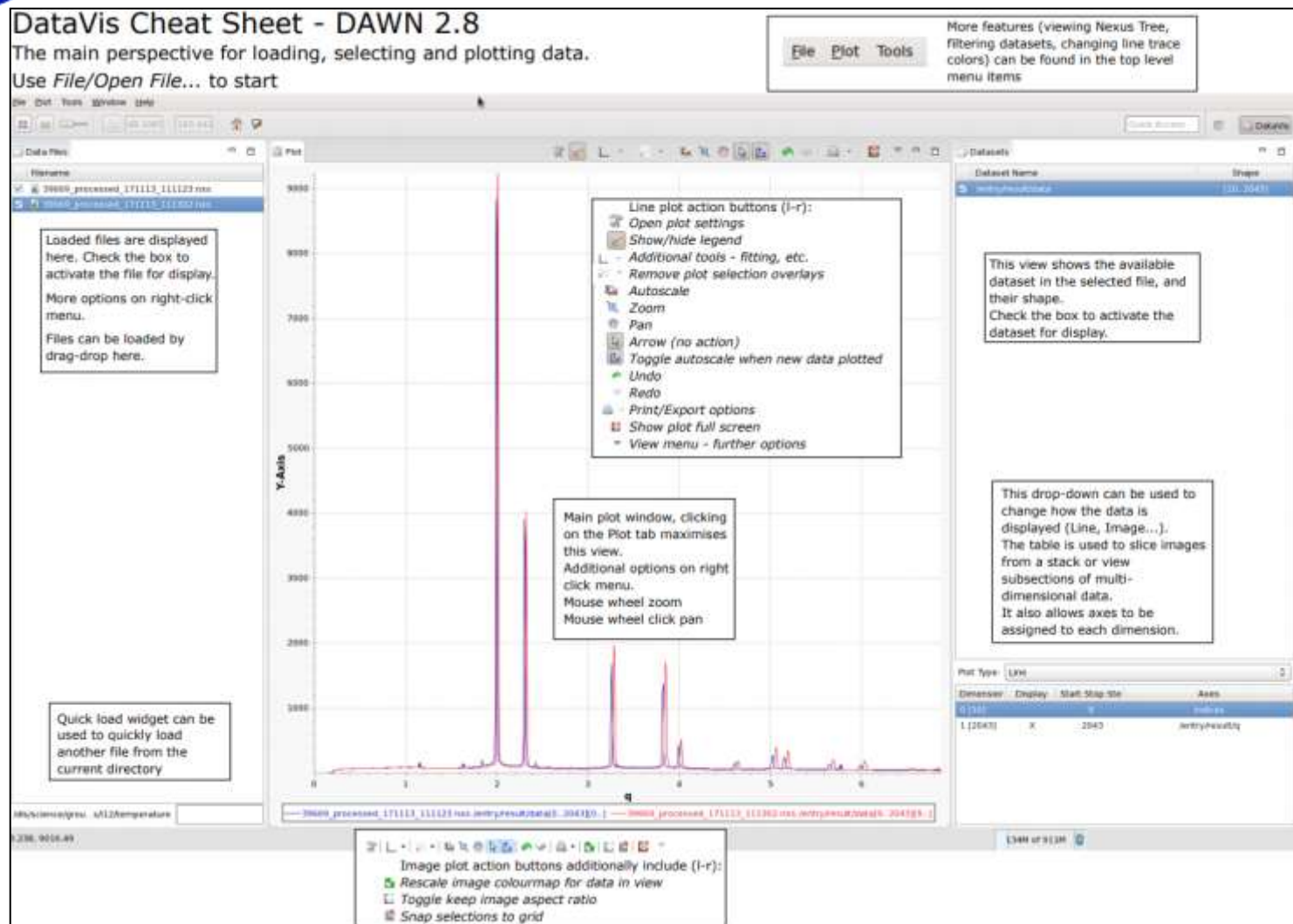
Assign axes to each dimension

The screenshot shows the DataVis Perspective interface. A dropdown menu is open, displaying various plot types: Line, Image, Text Table 1D, Text Table 2D, Surface, Waterfall, Hyper3d, Volume, Volume - Orthogonal Slices, and Hyper4d. Below the dropdown, the "Plot Type" is set to "Image". A table below the plot type shows the assignment of axes to dimensions. A red arrow points from the text "Assign axes to each dimension" to the "Y" label in the "Display" column of the table. Another red arrow points from the text "Choose how to display the selected dataset from the dropdown" to the dropdown menu.

Dimension	Display	Start:Stop:Step	Axes
0 [100]		0	/entry/DET/stagex
1 [180]		0	/entry/DET/stagea
2 [20]	Y	:20	indices

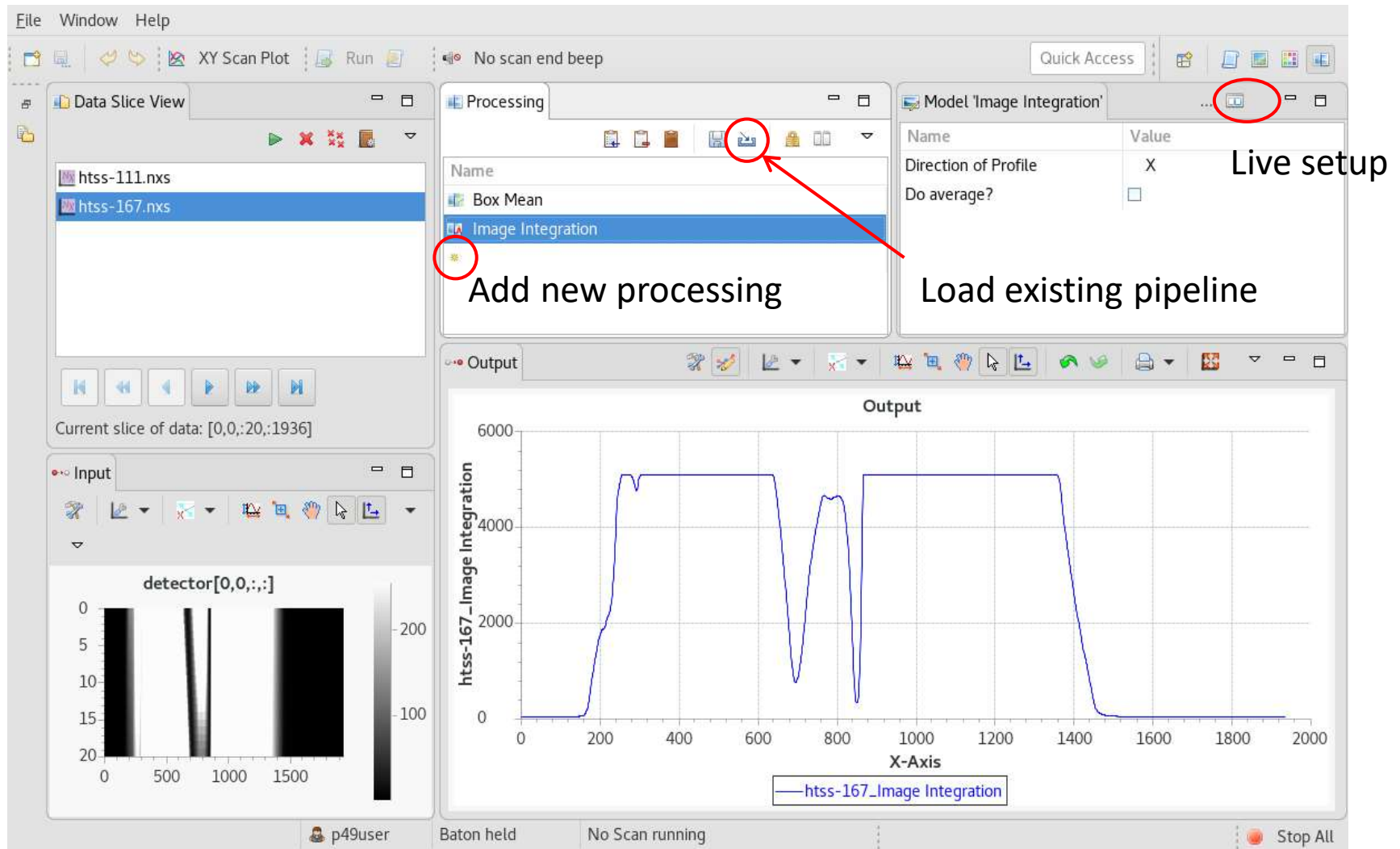


[https://dawnsci.org/assets/pages/using\\_dawn/DataVisCheatSheet.pdf](https://dawnsci.org/assets/pages/using_dawn/DataVisCheatSheet.pdf)





# Live Processing





# Live Processing

## Set up data for processing

Select dataset, axes, whether to process as images [2D] or lines [1D] and which dimensions of the array are the data dimensions

Select dataset: /entry/DET/data [13, 100, 20, 1936]

☐ Line [1D] ☒ Image [2D]

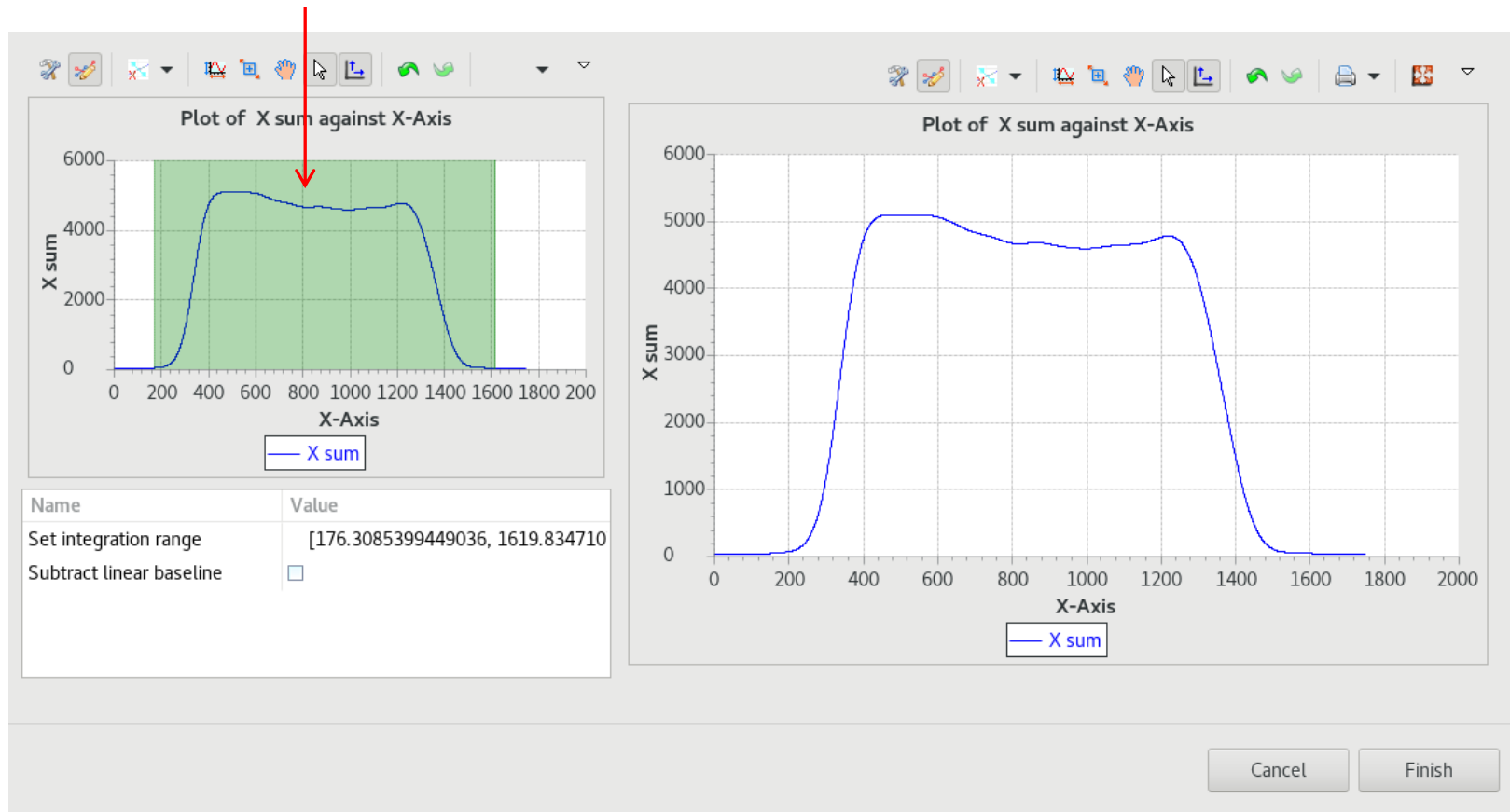
Dimension	Display	Start:Stop:Step	Axes
0 [13]		:	/entry/DET/stagex_value
1 [100]		:	/entry/DET/stagea_value
2 [20]	Y	:	indices
3 [1936]	X	:	indices

detector[0,0,:,:]

Cancel Finish

# Live Processing

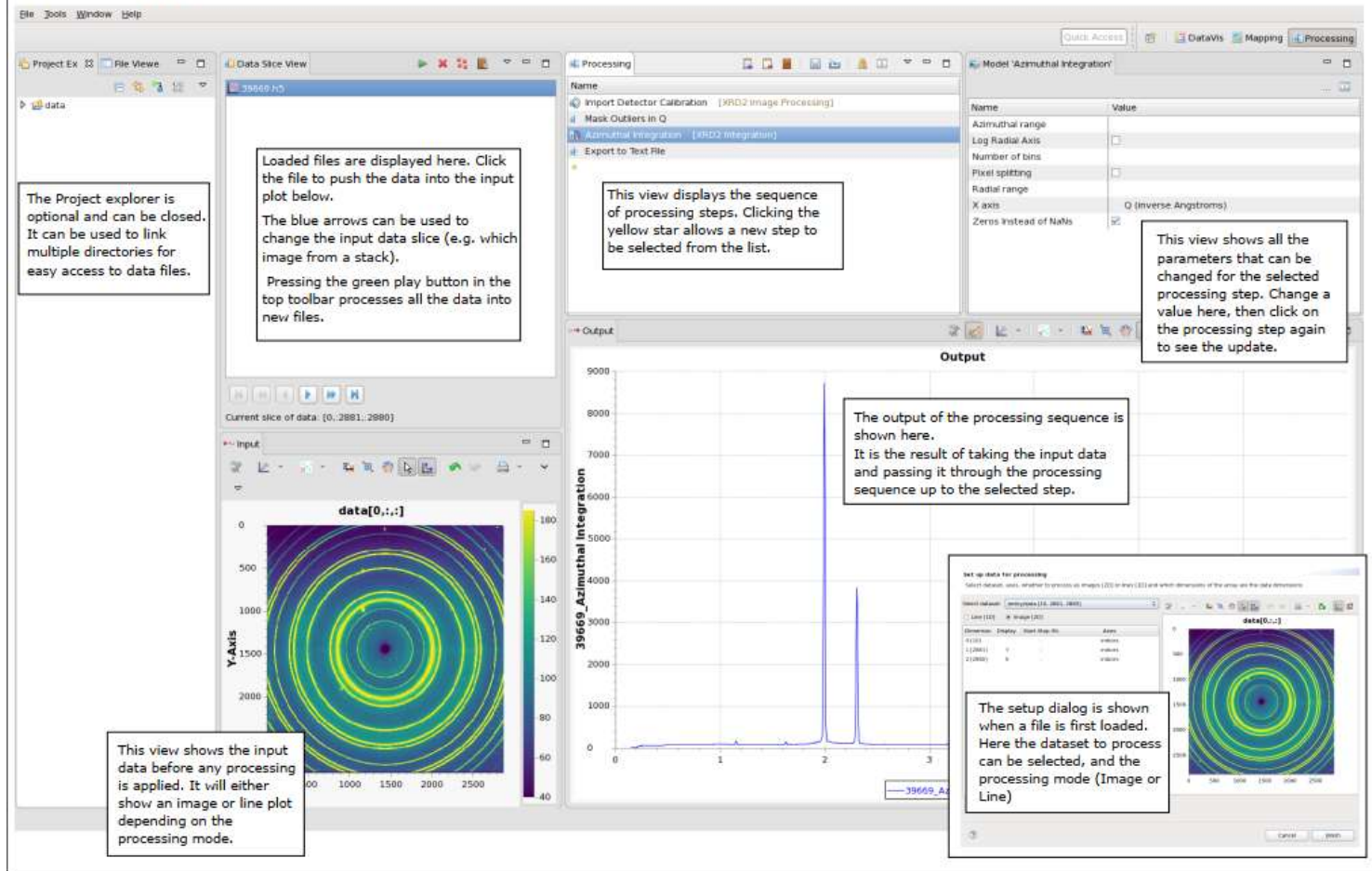
Set range



## Processing Cheat Sheet - DAWN 2.13

Build a processing sequence and apply it to data.

Use *File/Open File...* to start



The Project explorer is optional and can be closed. It can be used to link multiple directories for easy access to data files.

Loaded files are displayed here. Click the file to push the data into the input plot below.

The blue arrows can be used to change the input data slice (e.g. which image from a stack).

Pressing the green play button in the top toolbar processes all the data into new files.

This view displays the sequence of processing steps. Clicking the yellow star allows a new step to be selected from the list.

This view shows all the parameters that can be changed for the selected processing step. Change a value here, then click on the processing step again to see the update.

The output of the processing sequence is shown here. It is the result of taking the input data and passing it through the processing sequence up to the selected step.

This view shows the input data before any processing is applied. It will either show an image or line plot depending on the processing mode.

The setup dialog is shown when a file is first loaded. Here the dataset to process can be selected, and the processing mode (Image or Line)

**Model 'Azimuthal Integration'**

Name	Value
Azimuthal range	
Log Radial Axis	<input type="checkbox"/>
Number of bins	
Pixel splitting	<input type="checkbox"/>
Radial range	
X axis	Q (inverse Angstroms)
Zeros instead of Nans	92

**Set up data for processing**

Select dataset (see whether to process as images (I) or lines (L)) and which dimensions of the array are the data dimensions

Dimension	Display	Start Step No.	Axis
0 (I)			values
1 (I)			values
2 (I)			values



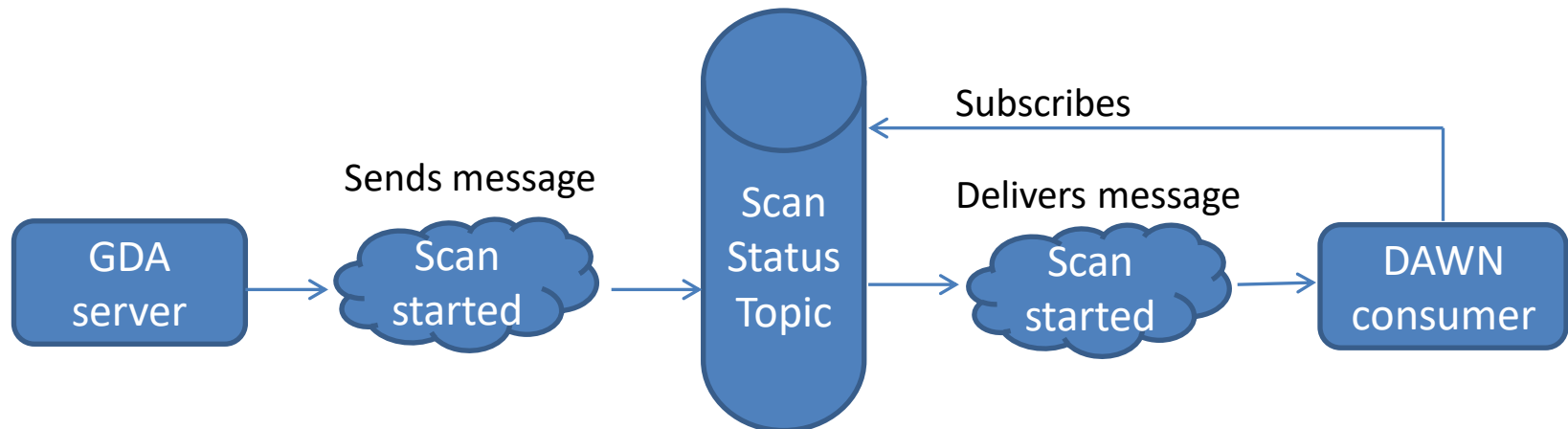
# Exercise: Make your own mean integration pipeline



- Open the `mean_integrate.nxs` file you used for the live processing in Part 1 and inspect it
- Clear the pipeline, and then recreate it using the star button to add processing steps
- Save it with a distinct name somewhere obvious
- Start a new scan using the old `mean_integrate.nxs` template, and the one you just created
- Hopefully they produce the same result!
- Play around with other processing steps (e.g. Gaussian filter)

# Live Processing

- GDA can submit pipelines to DAWN for live processing
- Publish-subscribe mechanism using ActiveMQ
- Other subscribers can also be written e.g. using Python






ActiveMQ  
Web console  
on port 8161


Scan topic →

Processing topic ↗

‘Processing  
finished’ message  
triggers final  
reconstruction



# ActiveMQ<sup>TM</sup>



The Apache  
Software Foundation  
<http://www.apache.org/>

Home | Queues | Topics | Subscribers | Connections | Network | Scheduled | Send Support

Topic Name  Create

### Topics

Name	Number Of Consumers	Messages Enqueued	Messages Dequeued ↓	Operations
org.eclipse.scanning.status.topic	3	863	2580	Send To Delete
scisoft.operation.STATUS_TOPIC	2	112	224	Send To Delete
gda.event.command_server	1	196	154	Send To Delete
org.eclipse.scanning.response.device.topic	0	73	73	Send To Delete
org.eclipse.scanning.request.device.topic	1	73	73	Send To Delete
org.eclipse.scanning.command.topic	3	25	71	Send To Delete
org.eclipse.scanning.ack.topic	0	24	24	Send To Delete
gda.event.GDAMetadata	1	3	3	Send To Delete





# Live Processing - Details

- DAWN consumer receives a JSON message on scan start and writes it to disk:
  - `{visit folder}/tmp/operationBean.json`
- JSON file contains the path to the scan data files
- Consumer then starts a new DAWN process on the cluster which reads the JSON file
- Log files also stored in `{visit folder}/tmp`
- *Note:* Test rig is setup slightly differently





# Live Processing - Details

- On beamlines, the DAWN Consumer and ActiveMQ processes run on ixx-control
- On the test rig, ActiveMQ runs under ProcServ:
  - `ioc-connect BL4xP-EA-PRCO-01`
  - Ctrl-X to reboot



# More flexible processing

- Status information about the scan is sent to an ActiveMQ topic: *gda.messages.scan*
- A script can subscribe to this topic to do arbitrary processing during or after a scan
- See visualhulls for an example
- There is a python3 module called *scansubscriber* to help

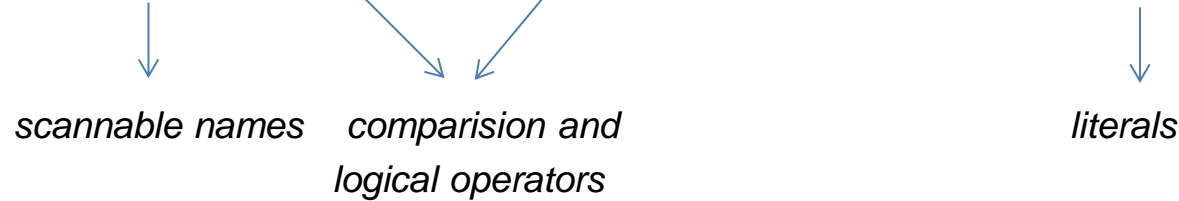
```
{
  "status": "UPDATED",
  "scanNumber": 0,
  "filePath": "/dls/htss/data/2019/0-0/htss-299.nxs",
  "detectors": [
    "BL49P-ML-SCAN-01"
  ],
  "scannables": [],
  "swmrStatus": "ACTIVE",
  "visitDirectory": "/dls/htss/data/2019/0-0",
  "scanDimensions": [
    100,
    100
  ],
  "processingRequest": {
    "visualhulls": [
      "/localhome/p49user/vh.json"
    ]
  },
  "percentageComplete": 18.38
}
```



# Watchdogs

- *Watchdogs* provide a mechanism to pause and resume a scan based on an external condition
- General case: *ExpressionWatchdog*

*ring\_current*  $\geq$  1.0 && !*portshutter.equals('Closed')*



Scan paused when expression becomes FALSE

- Special case: *TopupWatchdog*  
Monitors *SR-CS-FILL-01:COUNTDOWN*  
Configurable warmup and cooloff periods (ms)



# Configuring Watchdogs

## 1. Create an instance of *DeviceWatchdogModel*

- Give it an ID (name)
- Define the properties (dependent on the type of watchdog)
- Ensure it's linked to by the GDA server configuration

*ExpressionWatchdog:*

```
<bean id="noBeam" class="org.eclipse.scanning.api.device.models.DeviceWatchdogModel">  
  <property name="expression" value="ring_current >= 1.0"  
  <property name="message" value="Beam has been lost"/>  
</bean>
```

*TopupWatchdog:*

```
<bean id="beamTopup" class="org.eclipse.scanning.api.device.models.DeviceWatchdogModel">  
  <property name="countdownName" value="machineTopupMonitor"/>  
  <property name="message" value="Paused during topup"/>  
  <property name="warmup" value="5000"/>  
  <property name="cooloff" value="4000"/>  
  <property name="modeName" value="ringModeMonitor"/>  
</bean>
```



# Configuring Watchdogs

2. Create an *ExpressionWatchdog* or *TopupWatchdog* instance and link it to the model

*Example: ExpressionWatchdog*

```
<bean id="noBeam" class="org.eclipse.scanning.api.device.models.DeviceWatchdogModel">
  <property name="expression" value="ring_current >= 1.0"
  <property name="message" value="Beam has been lost"/>
</bean>

<bean id="expressionWatchdog"
class="org.eclipse.scanning.sequencer.watchdog.ExpressionWatchdog" init-method="activate">
  <property name="model" ref="noBeam"/>
  <property name="enabled" value="true" />
</bean>
```



# Watchdogs

Watchdogs are controlled at run time via the Jython console:

- `enableWatchdogs()`
- `disableWatchdogs()`
- `listWatchdogs()`



# Logs

- On the test rig, GDA logs are stored here:  
[/dls\\_sw/p4x/software/gda\\_logs](/dls_sw/p4x/software/gda_logs)
- But can use the GDA log monitor from launcher or:
  - > `module load gda`
  - > `gda`



- EPICS logs are in the IOC:  
</dls/p4x/epics/logs/BL4xP-EA-IOC-01>
- DAWN processing logs can be found here:  
</dls/p4x/epics/logs/BL4xP-EA-PRCO-0x>



# Practical Exercises



The scripts located in </localhome/p4xuser/training> each trigger a problem somewhere in the stack...

- Restart the machine to ensure a fresh start and confirm scans are working
- Run one of the scripts
- Start a new scan from GDA
- Try to figure out what goes wrong and how to fix it!
- No cheating!