

Experiment Control Software

John Skinner

March, 28, 2017

NSLS-II Facility Software Overview

NSLS-II Software Documentation — NSLS-II Software Documentation documentation - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Devices: collec... x ophyd.position... x Positioners — o... x Ophyd — ophy... x Interactive Co... x NSLS-II Softwa... x PV: Epics Proc... x StepScan: Epic... x python pro... x

nsls-ii.github.io

Search docs

Event-Based Architecture

Data Collection Quick Start

Data Analysis Quick Start

Sandbox

Deployment Details

Recommended Resources

Core Technologies

DATA COLLECTION

bluesky

ophyd

DATA ACCESS

databroker

DATA MUNGING

datamuxer

DATA EXPORT

suitcase

GITHUB LINKS

NSLS-II repositories

Wish List

Bug Reports

Docs » NSLS-II Software Documentation [View page source](#)

NSLS-II Software Documentation

We are deploying an event-based data collection and analysis framework.

Design Goals

- Provide an integrated, **end-to-end solution** for data collection and analysis.
- Support **streaming** data analysis, variously called “in-line” or “live.”
- Support **prompt** data analysis: immediate, semi-automated data analysis that can inform decisions made during an experiment
- Capture **metadata** to record a detailed snapshot of the hardware and — as much as possible — represent the user intention, the meaning of the measurements.
- Make datasets **searchable** with rich queries on metadata and data.
- Use **existing, open-source technologies and languages**; avoid inventing a domain-specific language.
- Leverage tools from the **open scientific software** community and in particular the **scientific Python** community, a mature and widely-used ecosystem of scientific code used in trusted, critical applications in research and beyond.
- Establish **clear, consistent interfaces** (meaning inputs and outputs, APIs) that allow project components to be used independently, extended, and interfaced with other, outside projects.
- Adhere to good software practices, especially code review and automated testing, with the goal of enabling **large-scale collaboration** while maintaining **stability and robustness**.

NSLS-II EPICS/Python Module - Ophyd

Ready-to-Use Devices — ophyd documentation - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Devices: collec... x ophyd.position... x Positioners — o... x Ophyd — ophy... x Interactive Co... x Ready-to-Use ... x PV: Epics Proc... x StepScan: Epic... x python pro... x

nsls-ii.github.io/ophyd/builtin-devices.html

github ophyd

ophyd
0.2.1+2.gf3f6591.dirty

Search docs

Overview of a Device
Interactive Command Interface
Ready-to-Use Devices
Positioners
Custom Devices
Signals
Area Detectors
Project Architecture

Docs » Ready-to-Use Devices [View page source](#)

Ready-to-Use Devices

These devices are have ready-made classes in Python. To configure them, the user need only provide a PV prefix and a name. Example:

```
from ophyd import EpicsMotor

# the two-theta motor
tth = EpicsMotor('XF:28IDC-ES:1{Dif:1-Ax:2Th}Mtr', name='tth')
```

class `ophyd.epics_motor.EpicsMotor`(*prefix*, *, *read_attrs=None*, *configuration_attrs=None*, *monitor_attrs=None*, *name=None*, *parent=None*, **kwargs) [\[source\]](#)

An EPICS motor record, wrapped in a `Positioner`

Keyword arguments are passed through to the base class, `Positioner`

Parameters:

- **prefix** (*str*) – The record to use
- **read_attrs** (*sequence of attribute names*) – The signals to be read during data acquisition (i.e., in `read()` and `describe()` calls)
- **name** (*str, optional*) – The name of the device
- **parent** (*instance or None*) – The instance of the parent device, if applicable
- **settle_time** (*float, optional*) – The amount of time to wait after moves to report status completion

class `ophyd.scaler.EpicsScaler`(*prefix*, *, *read_attrs=None*, *configuration_attrs=None*, *monitor_attrs=None*, *name=None*, *parent=None*, **kwargs) [\[source\]](#)

SynApps Scaler Record interface

NSLS-II Facility Scanning Software - BlueSky

The screenshot shows a Mozilla Firefox browser window with the URL `nsls-ii.github.io/bluesky/simple_api.html`. The page title is "(Optional) SPEC-like Interface - bluesky 0.0.1 documentation". The browser's address bar shows the search term "github ophyd".

The page content includes a navigation sidebar on the left with the following items:

- bluesky 0.0.1
- Search docs
- Plans
- Live Feedback and Processing
- Pausing and Resuming
- Recording Metadata
- Messages
- (Optional) SPEC-like Interface
- Conceptual Differences Between Bluesky and SPEC
- Specify Detectors
- Count
- Motor Scans
- Scans Tied to Particular Motors / Controllers
- Tweak

The main content area features the following text and code blocks:

[Docs](#) » (Optional) SPEC-like Interface [View page source](#)

(Optional) SPEC-like Interface

Conceptual Differences Between Bluesky and SPEC

Some scientists are familiar with [SPEC](#), a domain-specific language for hardware control. It is possible to imitate the SPEC workflow on top of bluesky. Of course, we still adhere to the Python syntax so that we can employ the full power of the general-purpose Python language.

It is useful to understand a key conceptual difference between bluesky and SPEC. SPEC treats the *specification* of an experiment ("move a motor from 1 to 5 in 5 strides") and its *execution* in one step. For example, in SPEC, typing

```
ascan th, 1, 5, 5
```

both specifies and executes the scan. Fundamentally, bluesky separates these steps: first we generate a "plan" (a set of granular instructions)

```
plan = AbsScanPlan(detectors, th, 1, 5, 4)
```

and then we pass the plan to a `RunEngine` for execution.

```
RE(plan)
```

To imitate the SPEC workflow, these two steps are lumped together.

```
ascan(th, 1, 5, 5) # this is bluesky's imitation of SPEC
```

NSLS-II Facility Experiment Database

The screenshot shows a Mozilla Firefox browser window displaying the NSLS-II Facility Experiment Database website. The browser's address bar shows the URL `https://nsls-ii.github.io/amostra/introduction.html`. The page has a blue header with the 'amostra' logo and a search bar. A left sidebar contains a navigation menu with items like 'amostra', 'Introduction', 'Overview', 'Schema', 'Setup', 'Online Sample Management Examples', 'Local Sample Management Examples', 'More Examples', and 'amostra package'. The main content area is titled 'Introduction' and 'Overview'. The 'Overview' section contains several paragraphs of text describing the amostra service, its RESTful API, and its flexibility in handling semi-structured data. The 'Schema' and 'Sample' sections are partially visible at the bottom of the page.

Introduction

Overview

amostra is a sample management service for scientific experiments. It is implemented as a web service with a RESTful api. Amostra is not a data acquisition tool that controls robots etc. It keeps track of an experiment/run's samples, sample groups(containers), and requests that act on these samples. The full amostra collection suit is available but optional for every beamline or application. Flexible No-SQL nature allows storing semi structured sample information (which I have proven works in every beamline so far). Amostra can be coupled with 3rd party applications such as sample management hardware(e.g. robots), PASS system, or csv import/export utility libraries.

The service comes with both an online api that communicates with the amostra service server and a local api that is capable of performing similar tasks offline(locally). The goal of this approach is to provide users with the ability to perform sample management tasks similar to their beamline environment when they have no access to amostra server. The flexible No-SQL nature allows storing semi structured sample information. What is meant by semi-structured?

Sample, Container, and Request documents all expect certain required fields such as time, uid, name, etc. and user can define any additional fields that the values of these fields can be string, array, document, or array of documents. I hide some of the complexity of semi-structured documents from the end user on the server by defaulting the required values to their appropriate values. If preferred, the user can define these parameters himself/herself. This is intended for those those that have existing tools -which use some other convention- or would like to build their own convention by populating these with values that make more sense for their experiment.

Relationships among collections are made available for any application to utilize and all relationships are completely optional. For instance, a beamline that is interested in this service for solely keeping track of their samples and have only one container, can use the **SampleReference** client, leaving **container** field empty. On the other hand, if an experiment has various containers, samples, and requests that are meaningful to some external source (either daq or analysis), one can use all three collections, using the relationships defined by this service. I talk about how these relationships can be utilized in the Schema section below. In addition to this, you could take a quick look at tutorial section for code examples.

Schema

Sample

The sample collection provides users with tools to save/restore/query attributes of their

LSDC – Rastering

LSDC

File Collect XRF Spectrum

DewarView PriorityView

1

- 1A XtalPuck1_11
 - 1-XtalSamp1_11_0
 - AS0126_raster
 - AS0126_standard
 - AS0126_standard
 - AS0125_standard
 - AS0125_standard
 - AS0125_standard
 - AS0124_standard
 - AS0124_standard
 - AS0124_standard
 - AS0123_standard
 - AS0123_standard
 - AS0122_standard
 - AS0122_standard
 - AS0122_standard
 - AS0122_standard
 - AS0121_standard
 - AS0121_raster
 - AS0121_standard
 - AS0121_standard

Collect Queue Puck to Dewar... Stop Collection Remove Puck... Mount Sample Queue All Selected Unmount Sample deQueue All Selected Expand All Collapse All Pause Queue Empty Queue

Control Master

Sample Control Energy Scan

Standard Collection

Acquisition

Oscillation Start: 13.0 Oscillation End: 13.4

Oscillation Range: 0.2 ExposureTime: 0.5

Transmission (%): 100.0 Energy (eV): 13479.78

Beam Width: 20.0 Beam Height: 20.0

Detector Dist. Readback: 400.00 SetPoint: 400.0 Move Detector

Protocol: raster Edge Resolution: 3.27

Raster Step 20.0 Coarse Fine Custom

Data Location

Base Path: /GPFS/CENTRAL/XF17ID2/jjakoncic/RA-301711_Feb02-2017 Browse...

Data Prefix: AS0126 File Number Start: 1

Data Path: /GPFS/CENTRAL/XF17ID2/jjakoncic/RA-301711_Feb02-2017/projID/AS0126/147/

Display Data (Albula)

Energy Scan

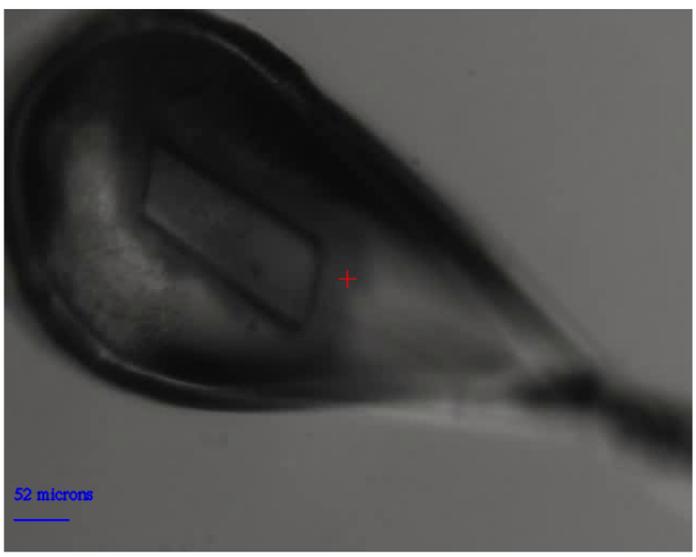
Priority Edit << < > >>

Add Requests to Queue

Apply Changes

Edit Screening Params...

Program Ready Shutter State: Closed Last File: /GPFS/CENTRAL/XF17ID2/jjakoncic/RA-301711_Feb02-2017/projID/AS0126/147/AS0126_Raster_10_3887_data_000001.h5



52 microns

Set DC Start Omega Readback: 13.60 SetPoint: 13.60 Move < 90 >

Video Source: LowMag HighMag Zoom Snapshot

Center Loop Draw Raster Clear 3-Click Center Save Center Select All Centerings

C2C Define Center Raster Explore Raster Select Define Raster Raster Evaluate By: Spot Count

Command>
Thu Feb 2 12:48:17 2017
runDCQueue()
Command>

LSDC – Rastering

LSDC

File Collect XRF Spectrum

DewarView PriorityView

1

- 1A XtalPuck1_11
 - 1-XtalSamp1_11_0
 - ASO126_raster
 - ASO126_standard
 - ASO126_standard
 - ASO125_standard
 - ASO125_standard
 - ASO125_standard
 - ASO124_standard
 - ASO124_standard
 - ASO124_standard
 - ASO123_standard
 - ASO123_standard
 - ASO123_standard
 - ASO12_standard
 - ASO12_standard
 - ASO12_standard
 - ASO12_standard
 - ASO121_standard
 - ASO121_raster
 - ASO121_standard
 - ASO121_standard

Collect Queue Puck to Dewar...
Stop Collection Remove Puck...
Mount Sample Queue All Selected
Unmount Sample deQueue All Selected
Expand All Collapse All
Pause Queue Empty Queue

Control Master

Sample Control Energy Scan

Standard Collection

Acquisition

Oscillation Start: 13.0 Oscillation End: 13.4
Oscillation Range: 0.2 ExposureTime: 0.5
Transmission (%): 100.0 Energy (eV): 13479.78
Beam Width: 20.0 Beam Height: 20.0
Detector Dist. Readback: 400.00 SetPoint: 400.0 Move Detector
Protocol: raster Edge Resolution: 3.27
Raster Step 20.0 Coarse Fine Custom

Data Location

Base Path: /GPFS/CENTRAL/XF17ID2/jjakoncic/RA-301711_Feb02-2017 Browse...
Data Prefix: ASO126 File Number Start: 1
Data Path: /GPFS/CENTRAL/XF17ID2/jjakoncic/RA-301711_Feb02-2017/projID/ASO126/147/

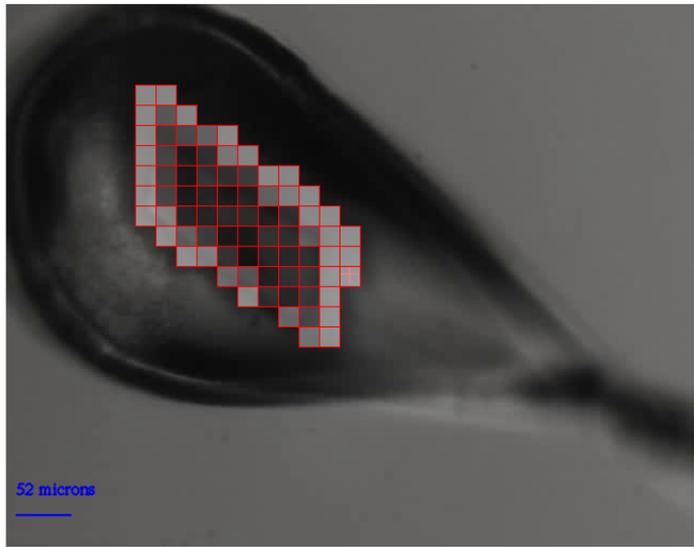
Display Data (Albula)

Energy Scan

Priority Edit << < > >>

Add Requests to Queue
Apply Changes
Edit Screening Params...

Program Ready Shutter State: Closed Last File: /GPFS/CENTRAL/XF17ID2/jjakoncic/RA-301711_Feb02-2017/projID/ASO126/147/ASO126_Raster_10_3887_data_000001.h5



52 microns

Set DC Start Omega Readback: 13.60 SetPoint: 13.60 Move < 90 >

Video Source: LowMag HighMag Zoom Snapshot

Center Loop Draw Raster Clear 3-Click Center Save Center Select All Centerings

C2C Define Center Raster Explore Raster Select Define Raster Raster Evaluate By: Spot Count

```
Command>  
Thu Feb 2 12:48:17 2017  
runDCQueue()  
Command>
```

LSDC – Helical Scanning

Main window (on xf17id1-ws2)

File Collect XRF Spectrum

PriorityView DewarView

1A Puck_1

- 1-samp_1_1
- 2-samp_1_2
- 3-samp_1_3
- 4-samp_1_4
- samp_1_4_vector
- 5
- 6
- 7-samp_1_7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16

1B Puck_2

- 1-samp_2_1
- 2-samp_2_2
- 3-samp_2_3
- 4-samp_2_4
- 5
- 6
- 7-samp_2_7
- 8
- 9
- 10
- 11
- 12

Sample Control | Energy Scan |

Standard Collection

Acquisition

Oscillation Start:	0.0	Oscillation End:	90.0
Oscillation Range:	0.2	ExposureTime:	0.2
Transmission (%):	100.0	Energy (KeV):	13.05
Beam Width:	30.0	Beam Height:	30.0
Edge Resolution:	2.0	Detector Distance:	150.88

Protocol:

Frames Per Point

Data Location

Base Path:

Data Prefix: File Number Start:

Data Path:

Display Data (Albula)

Energy Scan

Priority Edit

Omega Readback: 0.00 SetPoint:

Video Source: LowMag HighMag Zoom

C2C Raster Explore Raster Select Define Raster Raster Evaluate By:

Program Ready Last File: None

```
center_on_click(295.0,351.0,1,"screen",0)
Command>
Wed Mar 16 14:21:41 2016
center_on_click(283.0,219.0,1,"screen",0)
Command>
Wed Mar 16 14:21:55 2016
center_on_click(505.0,162.0,1,"screen",0)
Command>
```

LSDC – Energy Scan and Chooch Output (PyMCA GUI Classes)

The screenshot displays the PyMCA GUI interface. At the top, the window title is "Main window (on xf17id1-ws2)". The interface is divided into several sections:

- File Menu:** Includes "Collect" and "XRF Spectrum".
- View Modes:** "PriorityView" and "DewarView" are selected.
- Sample Control:** A tree view on the left shows sample lists "1A Puck 1" and "1B Puck 2". Below it are buttons for "Collect Queue", "Stop Collection", "Mount Sample", "Unmount Sample", "Expand All", and "Collapse All".
- Energy Scan:** A periodic table is displayed with element symbols. Below it are input fields for "Data Location", "Base Path" (set to "/nfs/skinner/testRun"), "Data Prefix", and "Data Path".
- Chooch Plots:** Two plots are shown. The top plot, titled "Chooch PLOT", shows a single black curve peaking at approximately x=12,662. The bottom plot, also titled "Chooch PLOT", shows two curves: a black "spline" curve and a red "fp" curve, both exhibiting a broad peak around x=12,662.
- Status Bar:** A green bar indicates "Program Ready" and "Last File: None".
- Command Line:** A terminal area at the bottom shows "Command>" prompts.

LIX Beamline Alignment and Commissioning using BlueSky

```
dscan(vfm.y1, -2., 2., vfm.y2, -2., 2., 40)

header, data = fetch_scan()
s_pos = data['vfm_y1_user_readback']
s_data = data['cam04_stats2_total']

plt.figure()
thresh = 0.95
opt_pos = np.mean(s_pos[s_data>np.max(s_data)*thresh])
plt.plot([opt_pos, opt_pos], [np.max(s_data), np.min(s_data)])
plt.plot(s_pos, s_data, s_pos[s_data>np.max(s_data)*thresh], s_data[s_data>np.max(s_data)*thresh], "o")
plt.show()
print("VFMy1, current position = %.3f , optimal position = %.3f" % (vfm.y1.position, opt_pos))

delta = opt_pos-vfm.y1.position
vfm.y1.move(vfm.y1.position+delta)
vfm.y2.move(vfm.y2.position+delta)

scn4y.move(scn4y.position+50.) # move to lim+
```

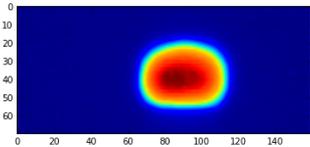
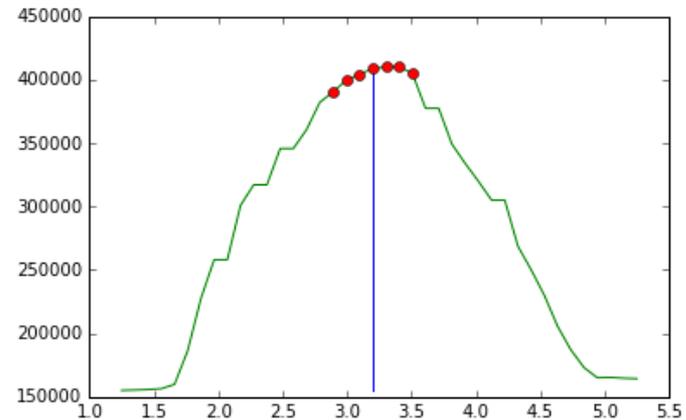


Table Cols: [StandardProsilica(prefix='XF:16IDA-BI{FS:4-Cam:1}', name='cam04', read_attrs=['stats1', 'stats2', 'st

seq_num	time	cam04_stats1_centroid_x	cam04_stats1_centroid_y	cam04_stats1_total	cam0
1	18:19:10.3	0.00	0.00	2.95e+05	
2	18:19:10.6	0.00	0.00	3.07e+05	
3	18:19:11.3	0.00	0.00	3.17e+05	
4	18:19:11.9	0.00	0.00	3.28e+05	
5	18:19:12.4	0.00	0.00	3.37e+05	
6	18:19:12.9	0.00	0.00	3.48e+05	
7	18:19:13.4	0.00	0.00	3.48e+05	
8	18:19:13.9	0.00	0.00	3.70e+05	
9	18:19:14.4	0.00	0.00	3.78e+05	
10	18:19:15.0	0.00	0.00	3.87e+05	
11	18:19:15.6	0.00	0.00	3.87e+05	
12	18:19:16.2	0.00	0.00	4.03e+05	
13	18:19:16.8	0.00	0.00	4.03e+05	
14	18:19:17.3	0.00	0.00	4.08e+05	
15	18:19:18.0	0.00	0.00	4.15e+05	
16	18:19:18.5	0.00	0.00	4.28e+05	
17	18:19:19.0	0.00	0.00	4.28e+05	
18	18:19:19.6	0.00	0.00	4.35e+05	
19	18:19:20.0	0.00	0.00	4.39e+05	
20	18:19:20.6	0.00	0.00	4.41e+05	
21	18:19:21.0	0.00	0.00	4.42e+05	
22	18:19:21.7	0.00	0.00	4.41e+05	
23	18:19:22.3	0.00	0.00	4.39e+05	
24	18:19:23.0	0.00	0.00	4.34e+05	
25	18:19:23.6	0.00	0.00	4.27e+05	
26	18:19:24.3	0.00	0.00	4.20e+05	
27	18:19:24.9	0.00	0.00	4.14e+05	
28	18:19:25.4	0.00	0.00	4.06e+05	
29	18:19:25.9	0.00	0.00	3.98e+05	



VFMy1, current position = 3.249 , optimal position = 3.198

To Do...

– From Martin Fuchs – FMX needs

- * Beamline state logging - authored and recoverable
- * Fully automatic beamline alignment: Calibrate all reproducible errors, Feedback on irreproducible ones.
- * Automation of mirror metrology (pencil beam scans, integration of Lei Huang's algorithms)
- * Active correction of the goniometer air bearing axis
- * User operated beam location (Governor BL, beam spot location, goniometer re-alignment)
- * Tuning of all goniometer positioners for maximum speed and low vibration operation
- * Control of the CRL transfocator
- * Full data collection automation, sample database back- and frontend
- * PI piezo scanner: Integration into the goniometer system (Yuan will help)
- * SmarGon multiaxis goniometer: Controls integration and strategy implementation
- * FMX secondary goniometer: Implement plate scanning functionality
- * Eiger control optimization - arming time, 4M ROI operation
- * Remote control of CryoStreams
- * Automatic dose logging with each data collection, intensity monitor readout

To Do...

From Jean Jakoncic - AMX Needs: Most items from FMX will apply to AMX too

In addition to FMX items...

- * Hutch temperature dependent beamline optimization (temp can range from 24 to 26 and things are moving quite a lot)
 - * Implementation of streaming interface for at least rastering and sample characterization using either edna or inhouse pipeline
 - * Implementation of the top view camera to be used for quick pin alignment and determining if pin is properly positioned
 - * Implementation of a routine to "unfocus" AMX beam from 5x5 to 20x20 microns beam using the pencil beam scanning
 - * Automated MAD data collection including (beam alignment)
 - * Implementation of inverse beam data collection
 - * Multiple data collection from a large sample or many samples in a loop
 - * Automated crystal location using the UV light and optical microscope
 - * Optimization of the rastering to enable a 400x300 microns raster using raster stepf of 5 microns in 30 secs or better (target time of 15 secs ?) in 9 M
 - * Implementation of "data privacy"
 - * Automated data back up on external hard drive provided by users
 - * Automated data transfer to user's institutional storage
 - * Implementation of a real time database allowing users to track experiments in real time (data collection and data processing)
 - * Implementation of monitoring tools for the automation (such at temp, expected time to exchange sample, warm up gripper ...) on LSDC
 - * Implementation of additional pipelines in LSDC
 - * Tools allowing a sanity check of gonio ?
- Automation:
- * Optimization of sample exchange timing
 - * Implementation of in-ln2 sample visualization (work 1/2 done so far)
 - * Automated robot repeatability check at the beginning of each day or week (routines)

To Do...

– From Lin Yang– LIX Needs

Data collection

High throughput: sample manager to associate data with samples. Improve reliability during data collection (e.g. prevent beam missing sample due to insufficient volume)

In-line HPLC: coordinate with HPLC software, retrieve UV/RI data and save into the data store. Improve reliability during data collection (e.g. automatic bubble removal)

Micro-beam mapping: optimization of scan software (minimize scan time, likely just a matter of implementing things other beamlines already have)

Data processing

Common: Data browser to review existing data (DAMA is working on something, need to develop different “plug-ins” to deal with different types of data visualization and processing)

Solution scattering: buffer subtraction. Exclusion of outliers (Hugo started something already). Sample-buffer pairing either comes from the database, or based on HPLC data (work with CSI??). Identification of issues such as aggregates.

Microbeam mapping: thumbnail mosaic generation; optimization of contrast; regeneration of maps.

Data analysis

Solution scattering: R_g , $P(r)$, model generation (make use of existing software package, should be able to set up batch runs easily, how to store/present the results)

Mico-beam mapping: vectorial tomography

LSBR and ABBIX Controls Staff (Present and Past)

Controls Lead – John Skinner

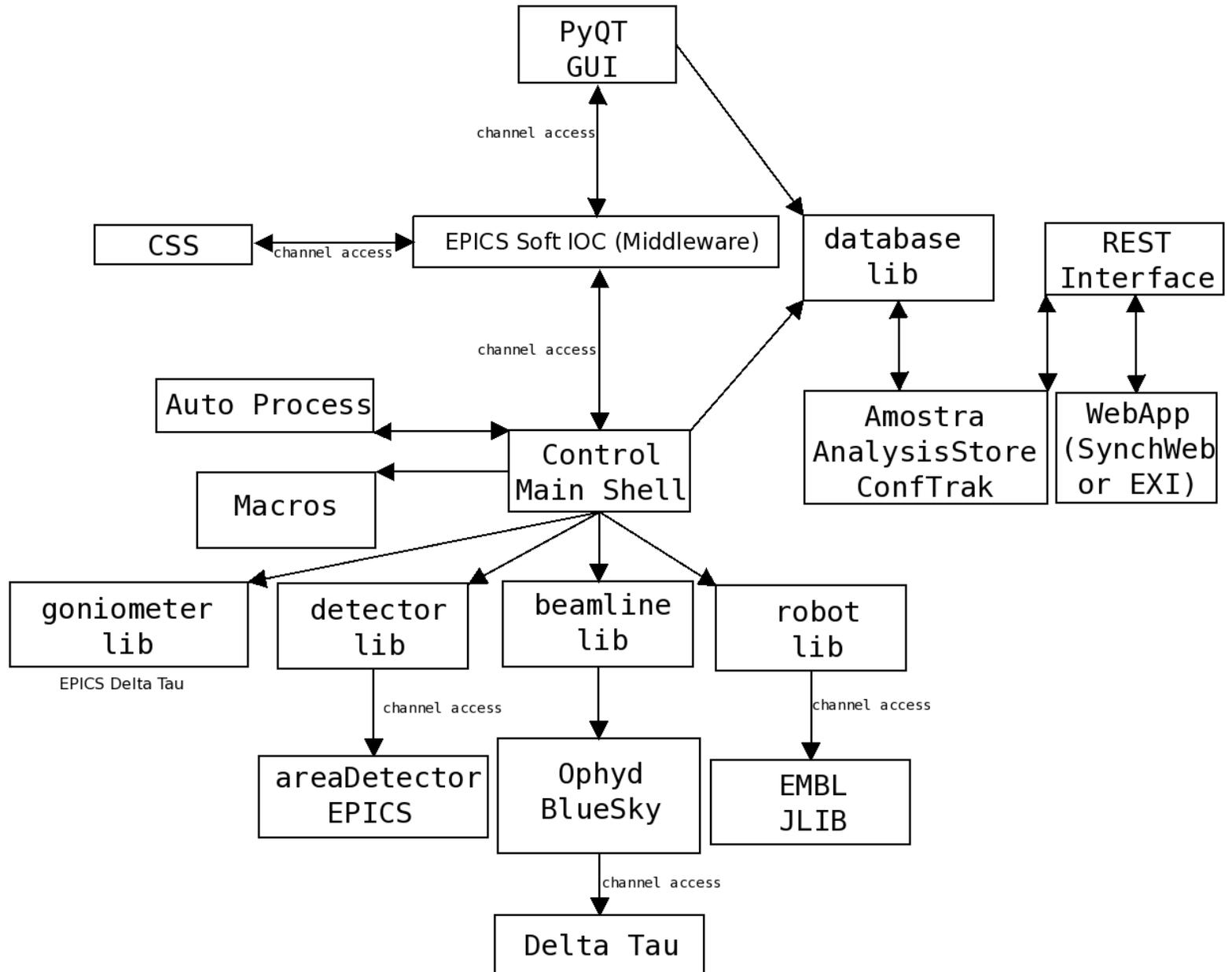
Motion Control Engineers – Stu Myers, Bruno Martins

Experiment Database Interfaces– [Hugo Slepicka](#)

Experiment Control Software and GUIs – John Skinner (*MX), [Hugo Slepicka \(LIX\)](#)

Automation – Kun Qian, [Nicolas Guichard](#)

LSDC Architecture



LSDC – Experiment and Sample Management Database

simplified mongo schema

“Collections”: (basically equivalent to RDBMS “Tables”)

- **Container:** id, type, name, owner, item_list
- **Sample:** id, type, name, owner, request_list, result_list, run_number
- **Request:** id, type, owner, timestamp, sample_id, priority, request_object
- **Result:** id, type, owner, timestamp, request_id, result_object

That’s basically it! Everything else is in the data/application, which can add any additional fields needed without altering the schema!

Much more useable for other techniques!

