

Introduction to

BECC

& association with  Bliss

BEC is the new Experiments Control System for SLS 2.0 beamlines

in-house development

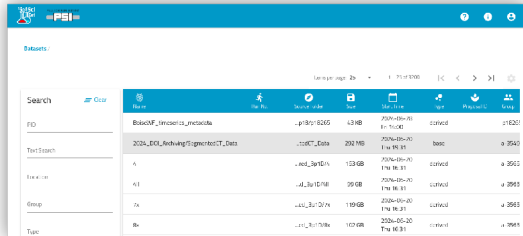
inspired by  bluesky (NSLS-II)



Open source software with BSD license
[Code repository](#) hosted on PSI gitlab

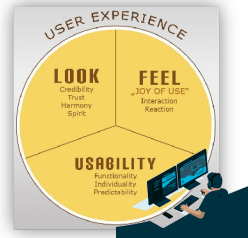


Data catalogue: SciCat



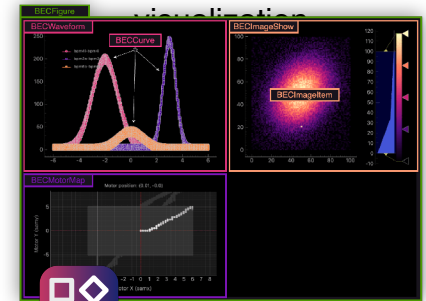
ID	File	Size	Created	Modified	Access	Owner
2024_BOL_..._products	..._BOL_..._products	43 MB	2024-09-11 14:10:41	2024-09-11 14:10:41	public	21820
2024_BOL_..._spectra	..._BOL_..._spectra	202 MB	2024-09-11 14:10:41	2024-09-11 14:10:41	public	21820
A	..._BOL_..._A	133 MB	2024-09-11 14:10:41	2024-09-11 14:10:41	public	21820
K1	..._BOL_..._K1	35 MB	2024-09-11 14:10:41	2024-09-11 14:10:41	public	21820
Temp	..._BOL_..._Temp	199 MB	2024-09-11 14:10:41	2024-09-11 14:10:41	public	21820
Temp	..._BOL_..._Temp	102 MB	2024-09-11 14:10:41	2024-09-11 14:10:41	public	21820

User interaction



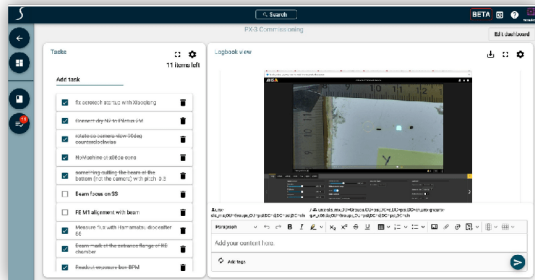
Experiment scripts, user sequences

Live data



BEC Widgets

E-logbook: SciLog




Data Processing

Data Acquisition: Ophyd devices



2D detectors Jungfrau IM, std-daq, AreaDetector...



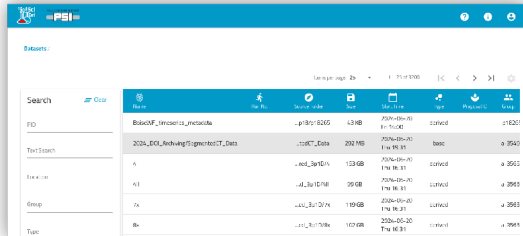
Beamline Device control

Data Archiving



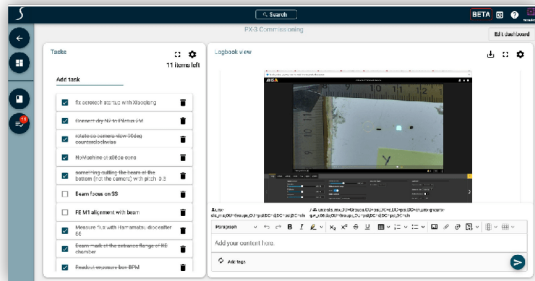
Storage Infrastructure

Data catalogue: SciCat

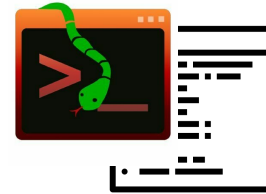
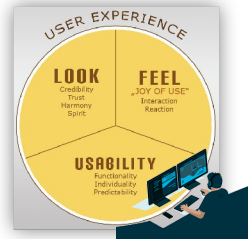


ID	File	Size	Created	Modified	Access
2025_BOL_..._products	..._products	43 MB	2025-09-01	2025-09-01	2025-09-01
2025_BOL_..._spectra	..._spectra	202 MB	2025-09-01	2025-09-01	2025-09-01
A	...	133 MB	2025-09-01	2025-09-01	2025-09-01
K1	...	35 MB	2025-09-01	2025-09-01	2025-09-01
...

E-logbook: SciLog

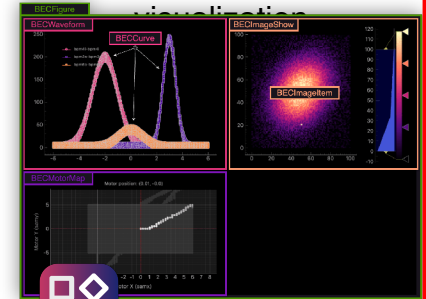


User interaction



Experiment scripts,
user sequences

Live data



BEC Widgets

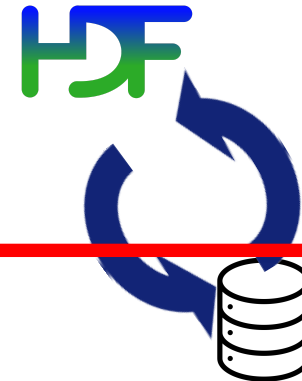


Data Processing

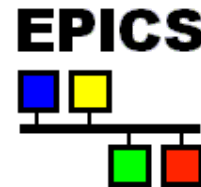
Data Acquisition: Ophyd devices



Data Archiving



2D detectors
Jungfrau JM, std-daq,
AreaDetector...

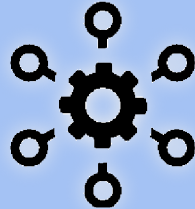


Beamline Device control

Storage Infrastructure



Written in Python



Services-oriented
architecture



redis

Redis as data buffer,
message broker



Devices control via
Ophyd (Bluesky)

sub-project: [ophyd_devices](#) 

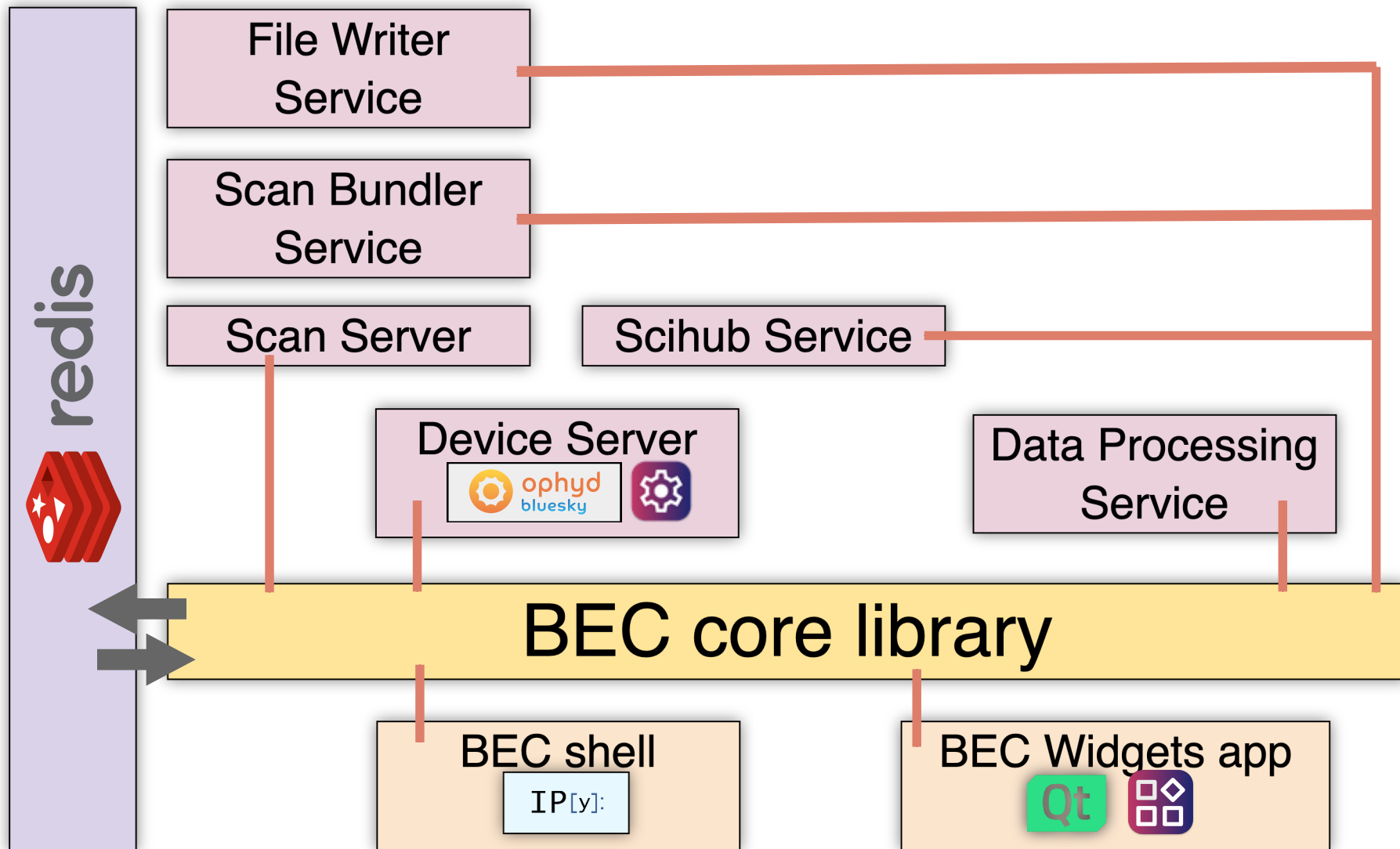
IP[y]:

IPython shell

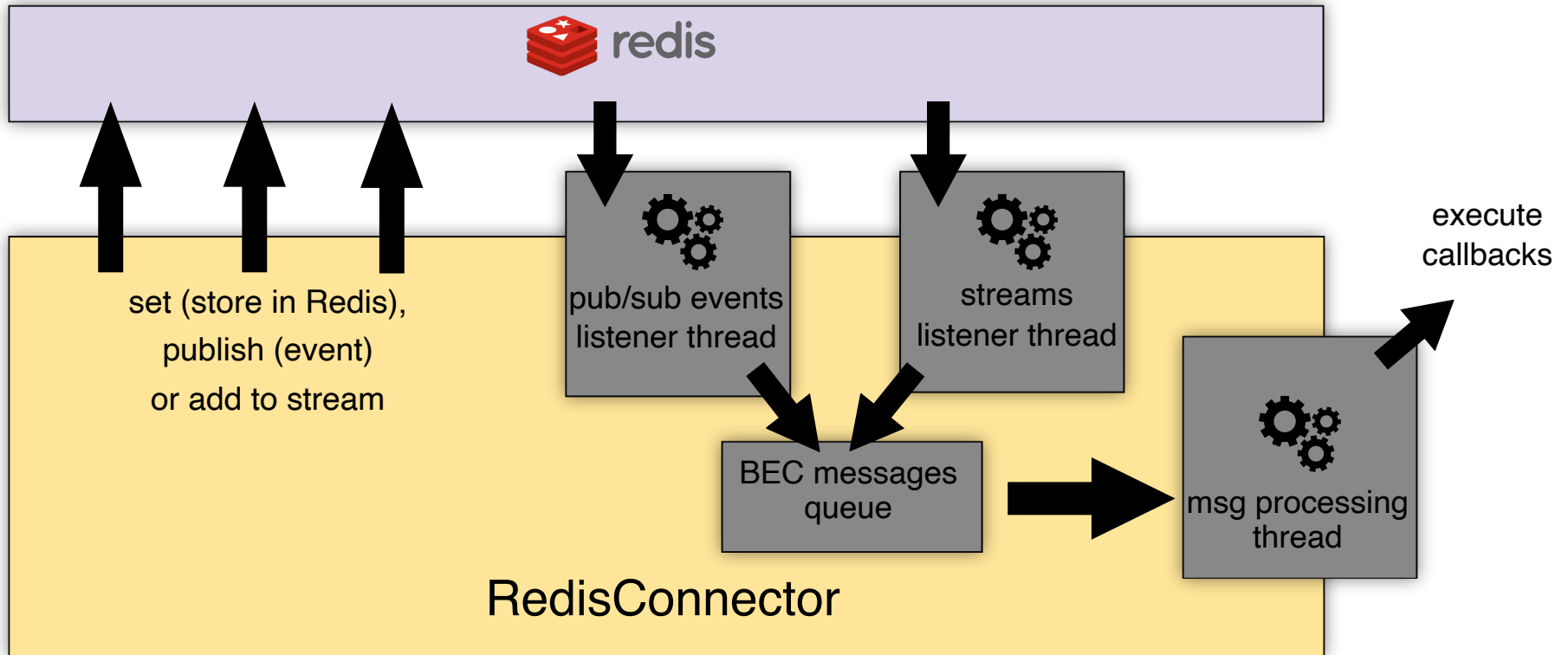


Graphical framework
based on Qt

sub-project: [bec_widgets](#) 



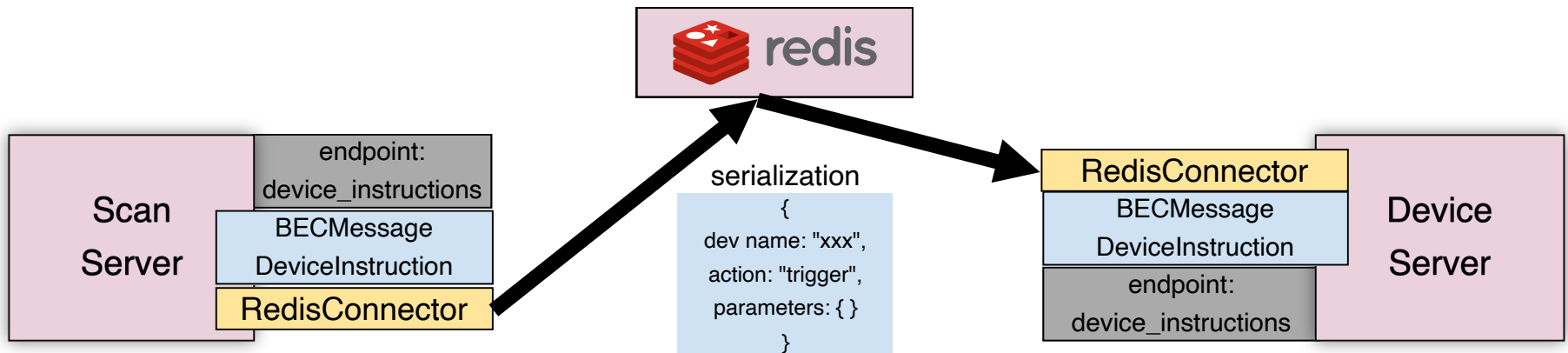
BEC core library object to communicate with Redis, used by all services



Messages that can be sent or received by the Redis Connector are defined using **Pydantic models**, and all derive from the **BECMessage** base class

Pydantic ensures data validation for the message data fields

BEC services connect to endpoints : named targets which specify the expected message type, and the action to take in Redis (store, publish, ...)



BEC configuration is defined in text files in YAML format

BEC servers configuration example

/home/matias/dev/example_servers_cfg.yaml

```
redis:
  host: 127.0.0.1
  port: 24878
mongodb:
  host: "localhost"
  port: 27017
scibec:
  host: http://localhost
  port: 3030
  beamline: TestBeamline
service_config:
  abort_on_ctrl_c: False
  enforce_ACLS: False
  file_writer:
    plugin: default_NeXus_format
    base_path: /tmp
  log_writer:
    base_path: /tmp
```

Beamline devices configuration example

/home/matias/dev/example_configuration.yaml

```
samx:
  readoutPriority: baseline
  deviceClass: ophyd_devices.SimPositioner
  deviceConfig:
    delay: 1
    limits:
      - -50
      - 50
    tolerance: 0.01
    update_frequency: 400
  deviceTags:
    - user motors
  enabled: true
  readOnly: false
bpm3a:
  readoutPriority: monitored
  deviceClass: ophyd_devices.SimMonitor
  deviceConfig:
    deviceTags:
      - beamline
  enabled: false
  readOnly: false
```

1. Starting BEC servers



2. Starting command line, loading configuration, starting a scan



https://git.psi.ch/psd_deployments/configs/sls/{beamline}



https://git.psi.ch/psd_deployments/configs/sls/{beamline}

Configuration files (YAML),
tell which versions on which hosts
have to be deployed



```
1 x06da-bec-001.psi.ch:
2   bec_version: main
3   ophyd_devices_version: main
4   bec_widgets_version: main
5   bec_plugins:
6     pxiii_bec: main
```

bec.yaml

```
x06da-bec-001.psi.ch:
  bec_redis_host: localhost
```

bec_console.yaml

https://git.psi.ch/psd_deployments/configs/sls/{beamline}

Configuration files (YAML),
tell which versions on which hosts
have to be deployed



<triggers>



ANSIBLE

```
1 x06da-bec-001.psi.ch:
2   bec_version: main
3   ophyd_devices_version: main
4   bec_widgets_version: main
5   bec_plugins:
6     pxiii_bec: main
```

bec.yaml

```
x06da-bec-001.psi.ch:
  bec_redis_host: localhost
```

bec_console.yaml

https://git.psi.ch/psd_deployments/configs/sls/{beamline}

Configuration files (YAML),
tell which versions on which hosts
have to be deployed



```
1 x06da-bec-001.psi.ch:
2   bec_version: main
3   ophyd_devices_version: main
4   bec_widgets_version: main
5   bec_plugins:
6     pxiii_bec: main
```

bec.yaml

```
x06da-bec-001.psi.ch:
  bec_redis_host: localhost
```

bec_console.yaml

<triggers>



ANSIBLE

<execute roles>

Console deployment: BEC IPython,
BEC Widgets app



Server deployment: BEC servers,
BEC beamline-specific code (plugins)



https://git.psi.ch/psd_deployments/configs/sls/{beamline}

Configuration files (YAML),
tell which versions on which hosts
have to be deployed



```

1 x06da-bec-001.psi.ch:
2   bec_version: main
3   ophyd_devices_version: main
4   bec_widgets_version: main
5   bec_plugins:
6     pxiii_bec: main
    
```

bec.yaml

```

x06da-bec-001.psi.ch:
  bec_redis_host: localhost
    
```

bec_console.yaml

<triggers>



ANSIBLE

<execute roles>

Console deployment: BEC IPython,
BEC Widgets app



Server deployment: BEC servers,
BEC beamline-specific code (plugins)



"bec_deployment" directory with source code + bec_venv Python virtual environment
with BEC packages installed via "pip -e" (editable/developer mode)

In no particular order :

Phase 1	Phase 2
Debye (X01DA)	MicroXAS (X05LA)
cSAXS (X12SA)	Phoenix (X07MB)
PXI PXII PXIII (MX)	RIXS (X03MA)
SuperXAS (X10DA)	Xtreme (X07MA)
SIM (X11MA)	VUV (X04DB)
PolLux (X07DA) + NanoXAS (X07DB)	XIL (X09L)
Addams (X04SA)	Optics (X05DA)
<i>Tomcat (X02DA, x2) - waiting for network connectivity</i>	Diagnostics (X01DD, X08DB)



Deployed and in use by BL scientist(s)



Deployed, not in use yet



No deployment yet



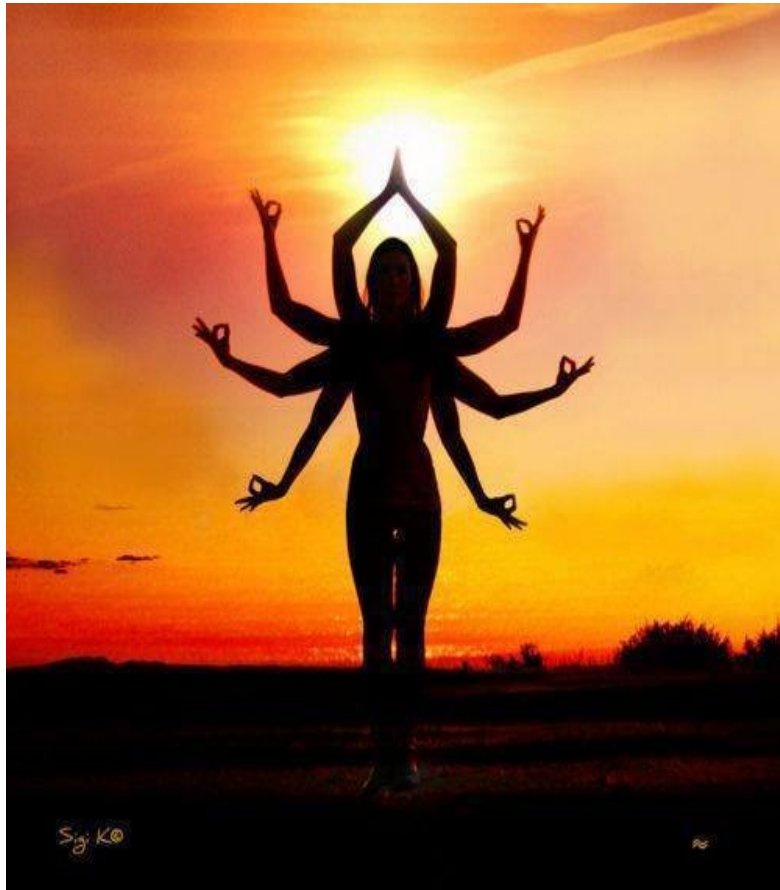
AWI Department Update

3rd of September, 2024

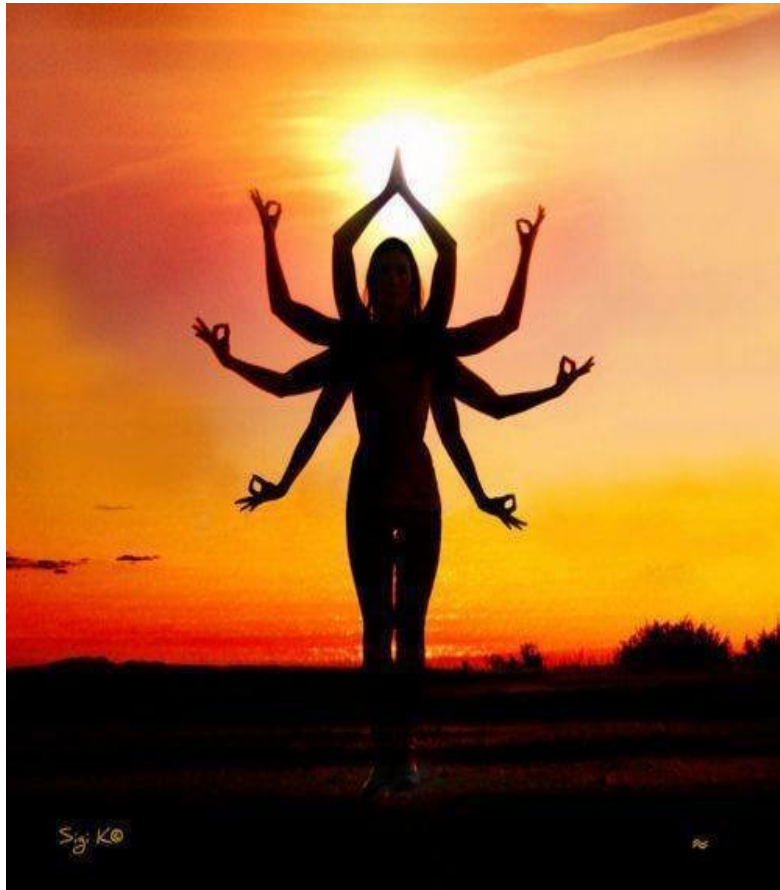
BEC &



Developed at ESRF since 2016 - designed for the EBS Upgrade



Developed at ESRF since 2016 - designed for the EBS Upgrade



Python library with tools



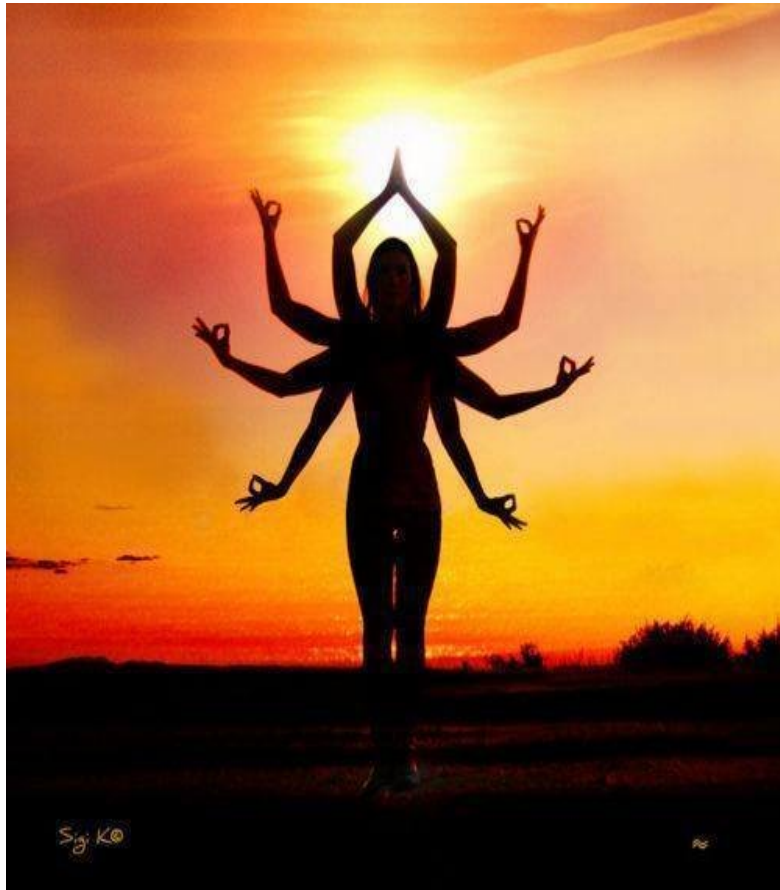
Command Line Interface, web terminal

Configuration application

Live visualization

Data service & file writer

Developed at ESRF since 2016 - designed for the EBS Upgrade



Python library with tools



Command Line Interface, web terminal

Configuration application

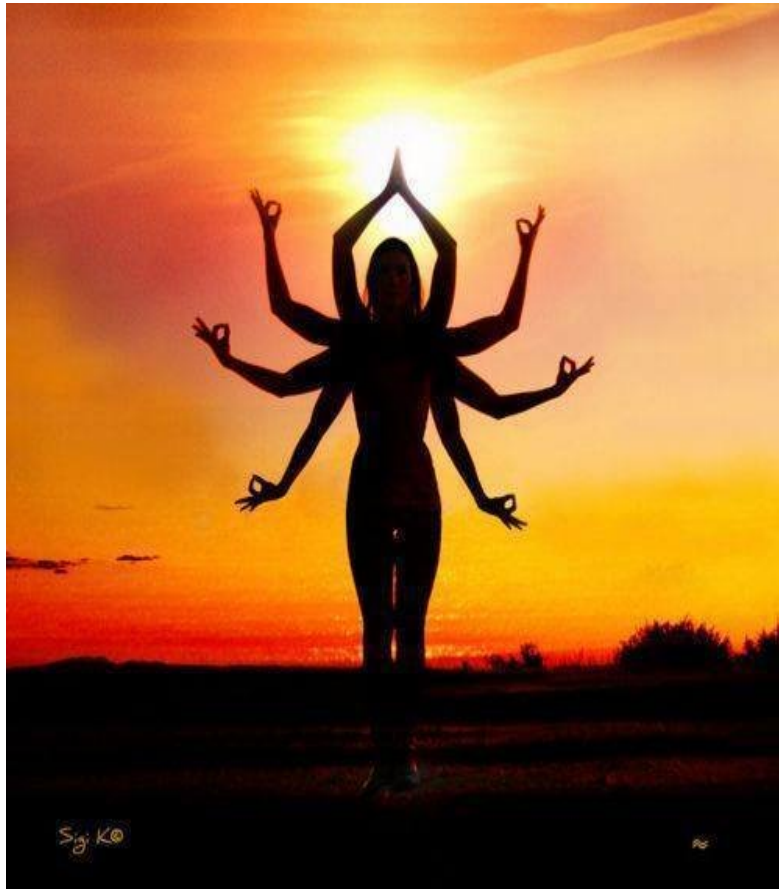
Live visualization

Data service & file writer

Sequencer for any kind of acquisition procedures

TANGO hardware control
or built-in drivers

Developed at ESRF since 2016 - designed for the EBS Upgrade



Python library with tools



Command Line Interface, web terminal

Configuration application

Live visualization

Data service & file writer

Sequencer for any kind of acquisition procedures

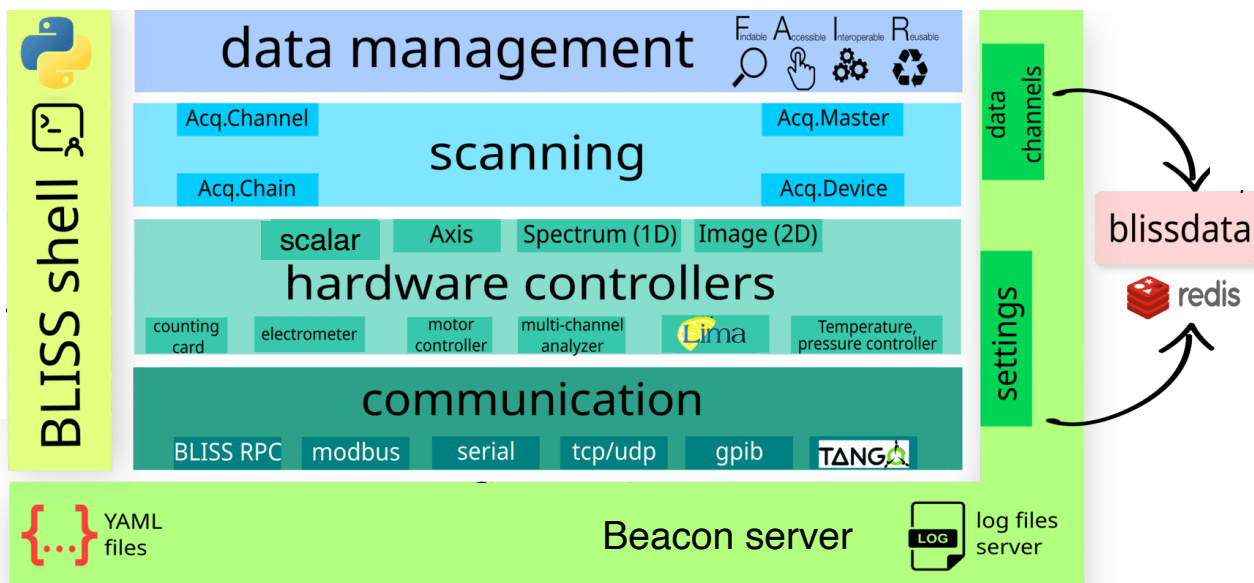
TANGO hardware control
or built-in drivers

At the heart of the ESRF software ecosystem

Implementation of Data Policy

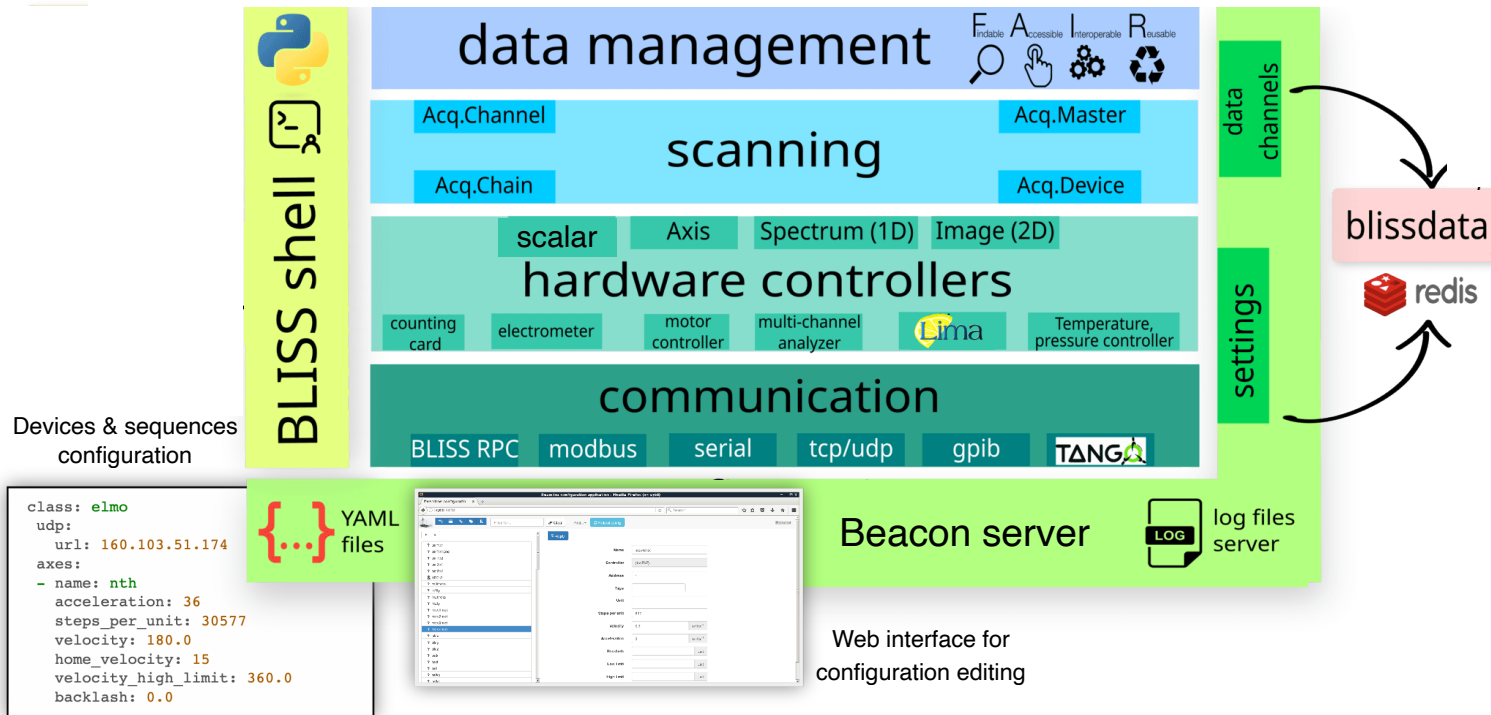
Graphical Interfaces

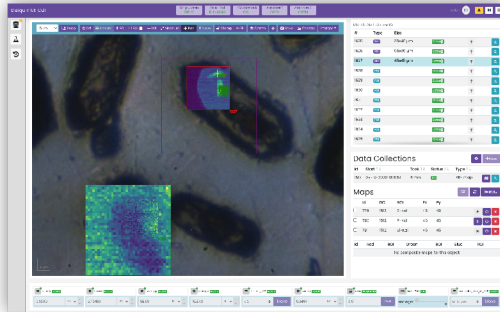
Online Data Analysis





BLISS overview





Daiquiri: ESRF web application framework for beamlines

blissterm (web terminal)

REST API

BLISS shell

data management E Findable A Accessible I Interoperable R Reusable

scanning

Acq.Channel Acq.Master

Acq.Chain Acq.Device

hardware controllers

scalar Axis Spectrum (1D) Image (2D)

counting card electrometer motor controller multi-channel analyzer Temperature, pressure controller

communication

BLISS RPC modbus serial tcp/udp gpib

data channels

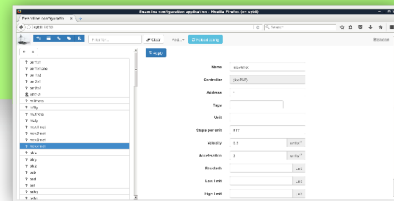
settings

blissdata

Devices & sequences configuration

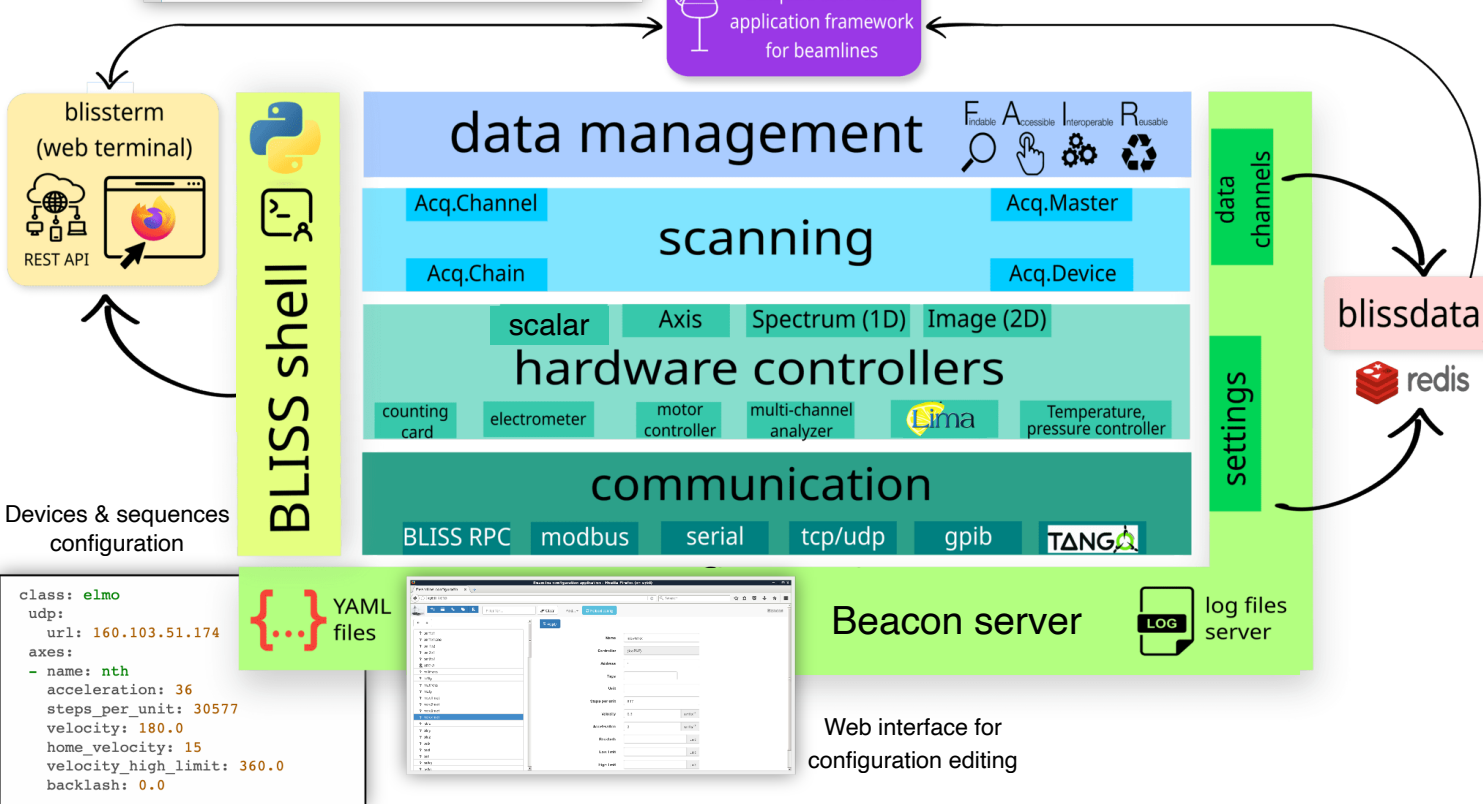
```
class: elmo
udp:
  url: 160.103.51.174
axes:
- name: nth
  acceleration: 36
  steps_per_unit: 30577
  velocity: 180.0
  home_velocity: 15
  velocity_high_limit: 360.0
  backlash: 0.0
```

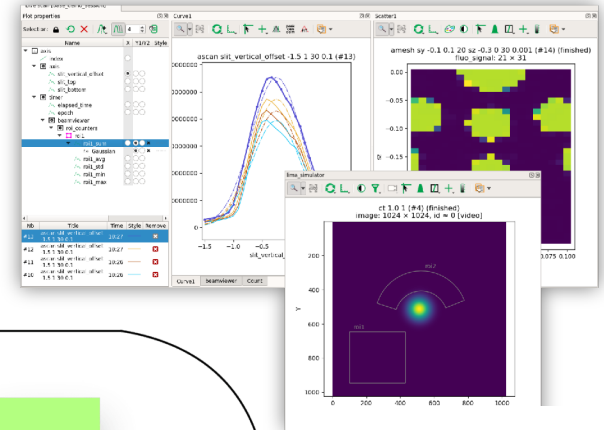
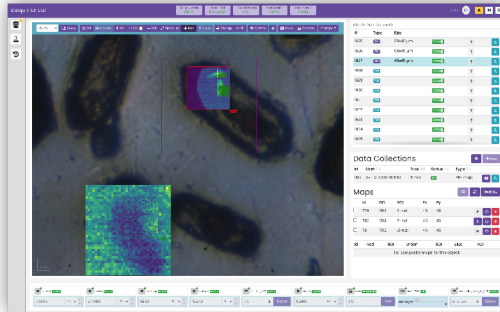
YAML files



Beacon server log files server

Web interface for configuration editing



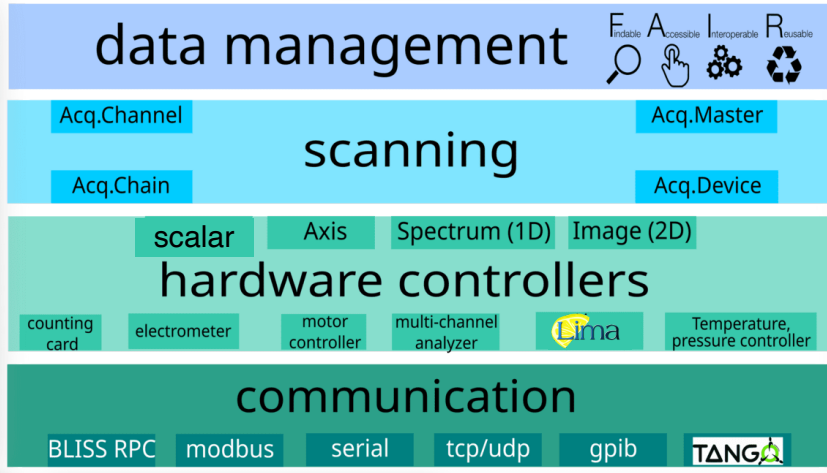


Daiquiri: ESRF web application framework for beamlines

blissterm (web terminal)

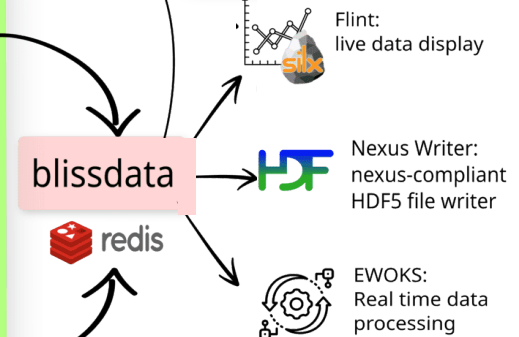
REST API

BLISS shell



data channels

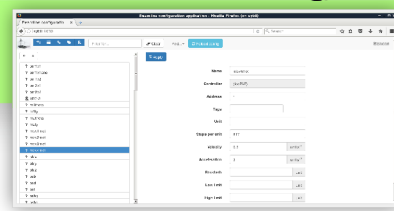
settings



Devices & sequences configuration

```
class: elmo
udp:
  url: 160.103.51.174
axes:
  - name: nth
    acceleration: 36
    steps_per_unit: 30577
    velocity: 180.0
    home_velocity: 15
    velocity_high_limit: 360.0
    backlash: 0.0
```

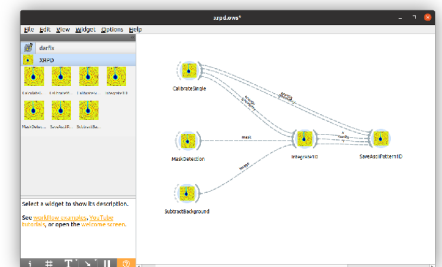
YAML files



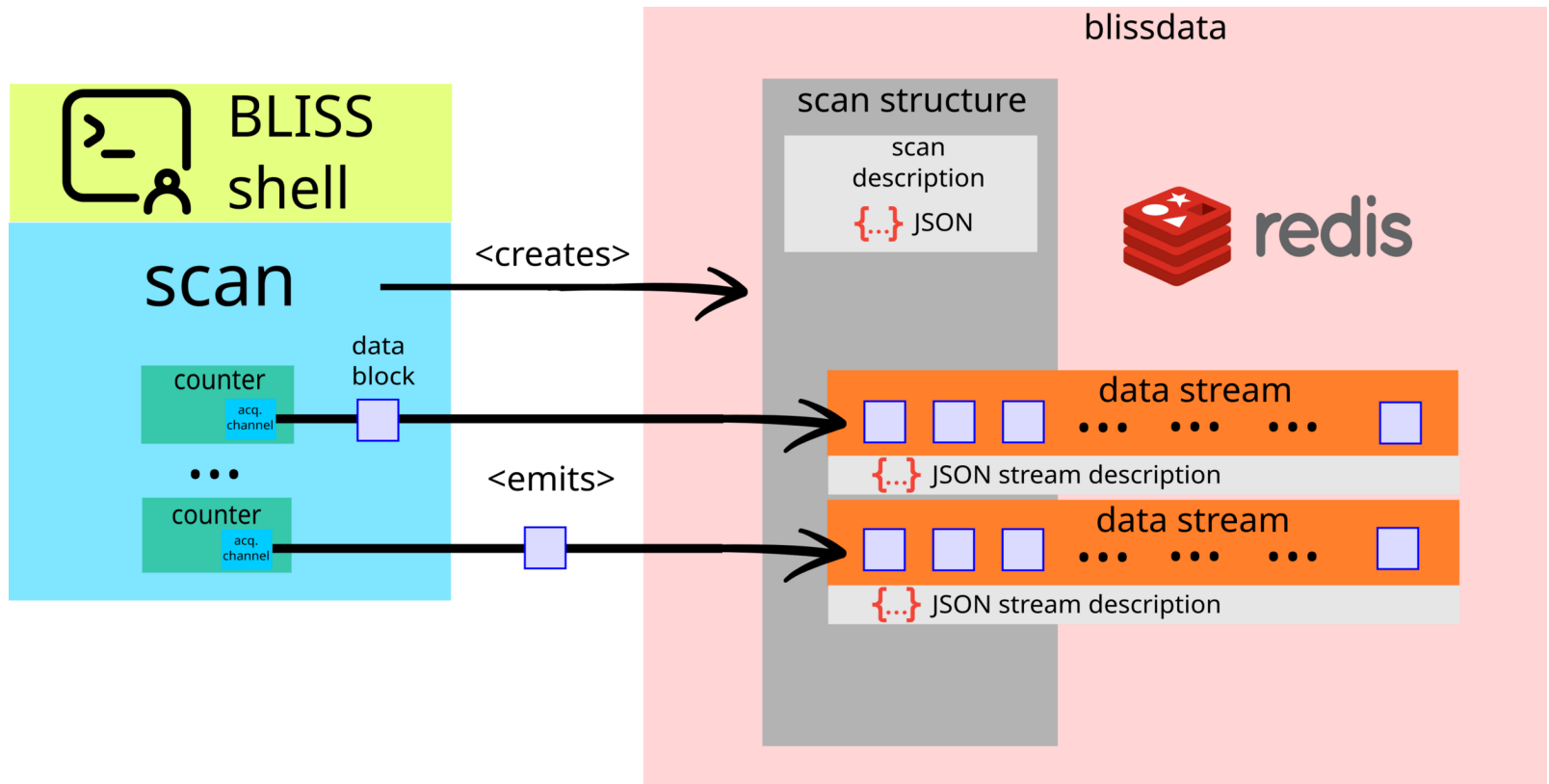
Beacon server

log files server

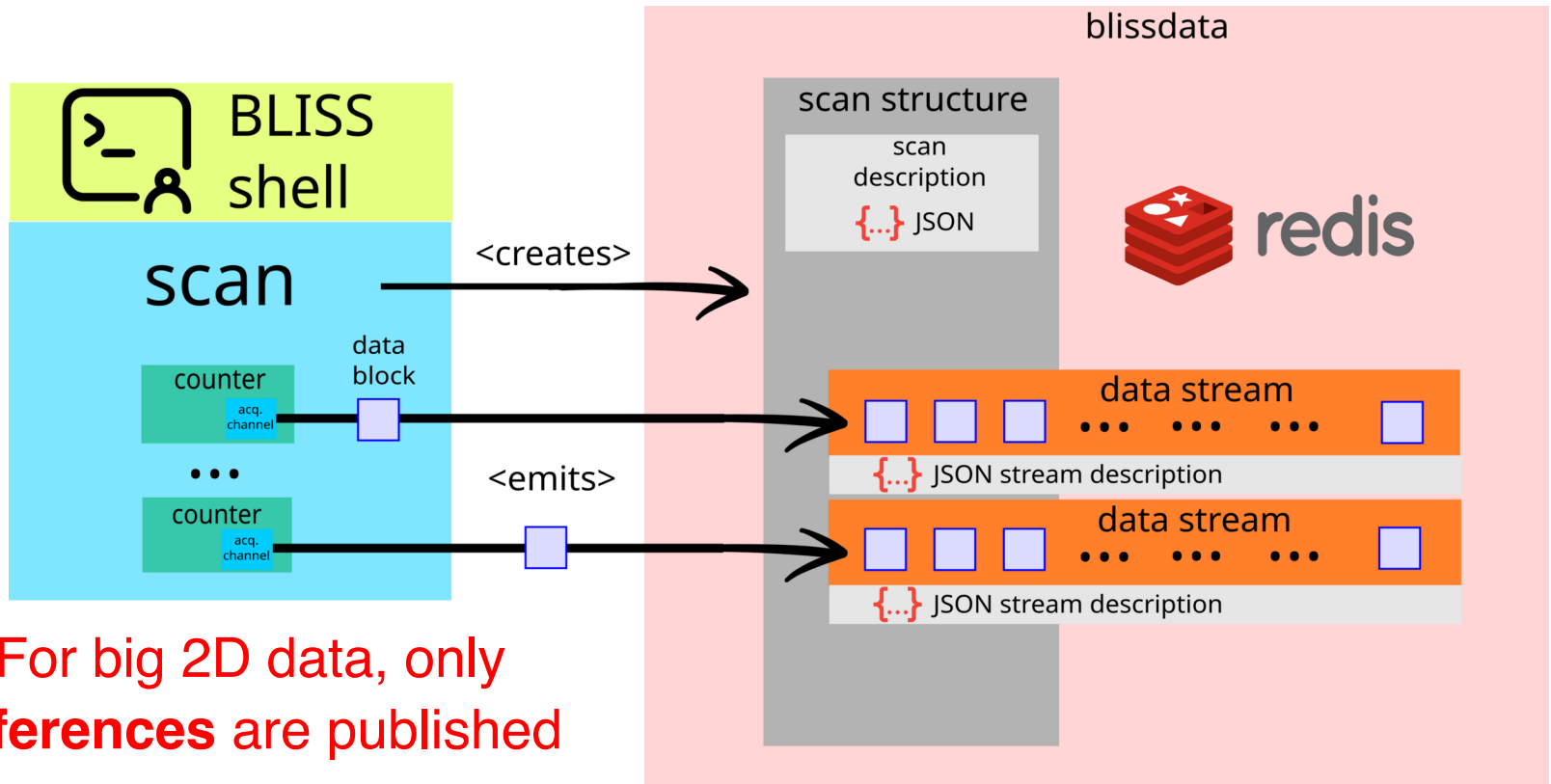
Web interface for configuration editing



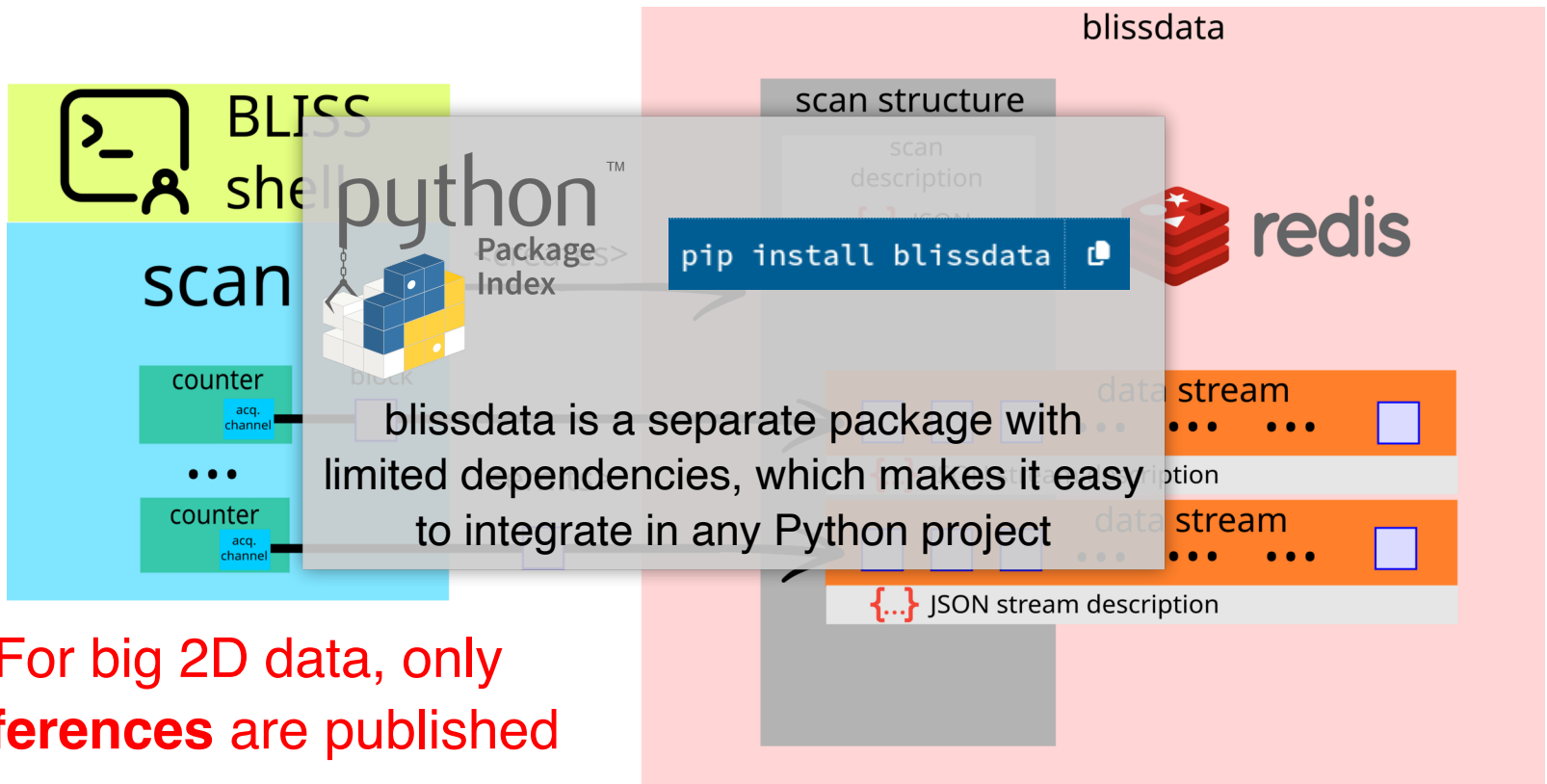
BLISS relies on blissdata to publish acquisition data to Redis



BLISS relies on blissdata to publish acquisition data to Redis



BLISS relies on blissdata to publish acquisition data to Redis



For big 2D data, only **references** are published



Symbiosis



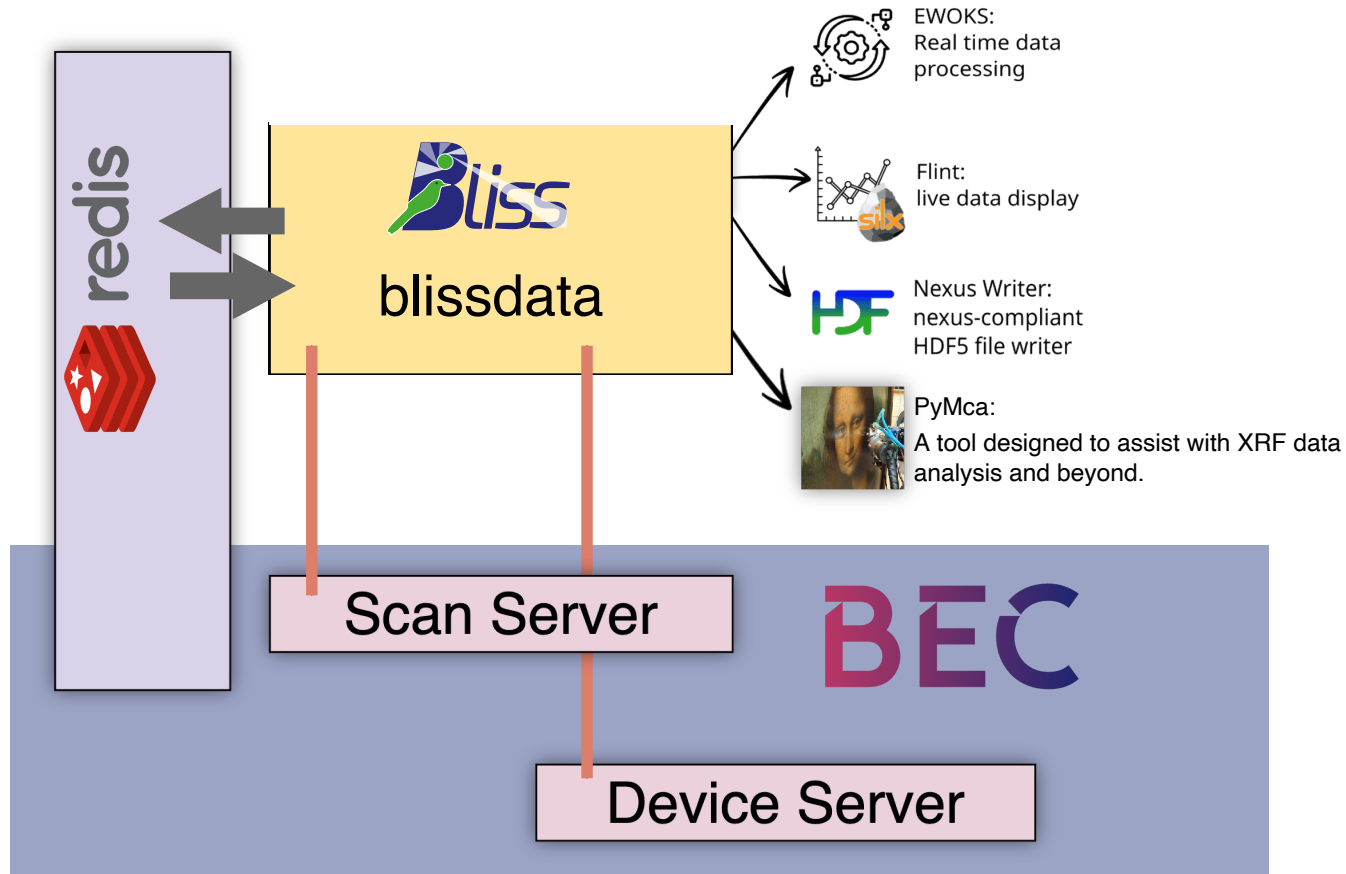


In which areas both BEC and BLISS could benefit each other ?

Where is it worth collaborating ? "Return on investment"

What would be beneficial for the users community ?

Publishing BEC scans with blissdata gives access to ESRF data writer, display and processing tools

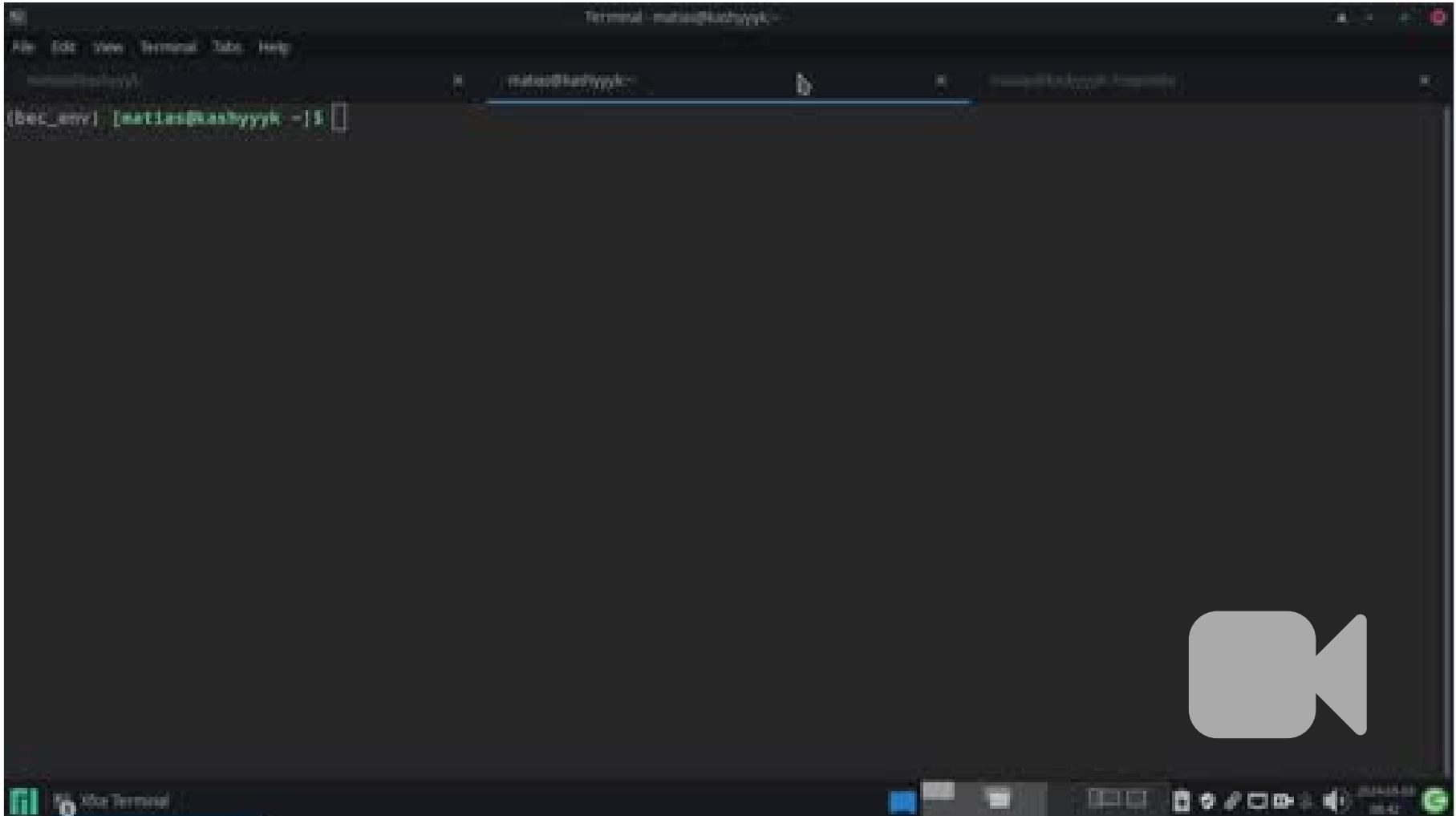




BEC with BLISS Writer and Flint



⚠ Work in progress ⚠





Conclusion

