Argonne NATIONAL LABORATORY

DATA QUALITY VERIFICATION TOOLS

BARBARA FROSIK

Principal Software Engineer
Scientific Software Engineering & Data Management
Advanced Photon Source
Argonne National Laboratory

March 20, 2017

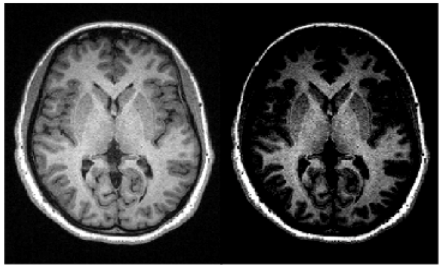
In collaboration with beamline 32-ID scientists
Francesco De Carlo
Doga Gursoy
et al

THANK YOU

- Francesco De Carlo, Doga Gursoy, Rafael Vescovi, Vincent De Andrade; Jun-Sung Park, John Hammomds, Nicholas Schwarz, Arthur Glowacki, Mark Rivers, Tim Madden, Sinisa Veseli, Tekin Bicer
- Thank you for great ideas, help, support, consultations.

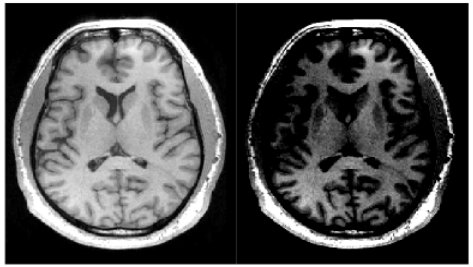
USE CASES

ANAT : Unexpected Inhomogeneity



Normal Contrast *High Contrast**

None: signal intensity uniform throughout image



Normal Contrast *High Contrast**

Expected: inconsistent signal intensity fits coil profile

Normal Contrast *High Contrast**

Unexpected: inconsistent signal intensity does not fit coil profile

*Signal inhomogeneity is most clearly visible at high contrast, adjusted by raising the Minimum Brightness.

http://cbs.fas.harvard.edu/usr/mcmains/CBS_MRI_Quality_Control_Workshop.pdf

VERIFICATION

32-ID Tomography, Transmission x-ray microscopy, Radiography, Phase contrast imaging

- Verifying PVs before start of experiment
- Verifying collected hdf data file structure
- Verifying collected hdf data
  - Frame mean value within limits
  - Frame standard deviation within limits
  - Frame mean value and mean of means does not exceed delta

BLOCKED PROJECTIONS

32-ID Tomography, Transmission x-ray microscopy, Radiography, Phase contrast imaging

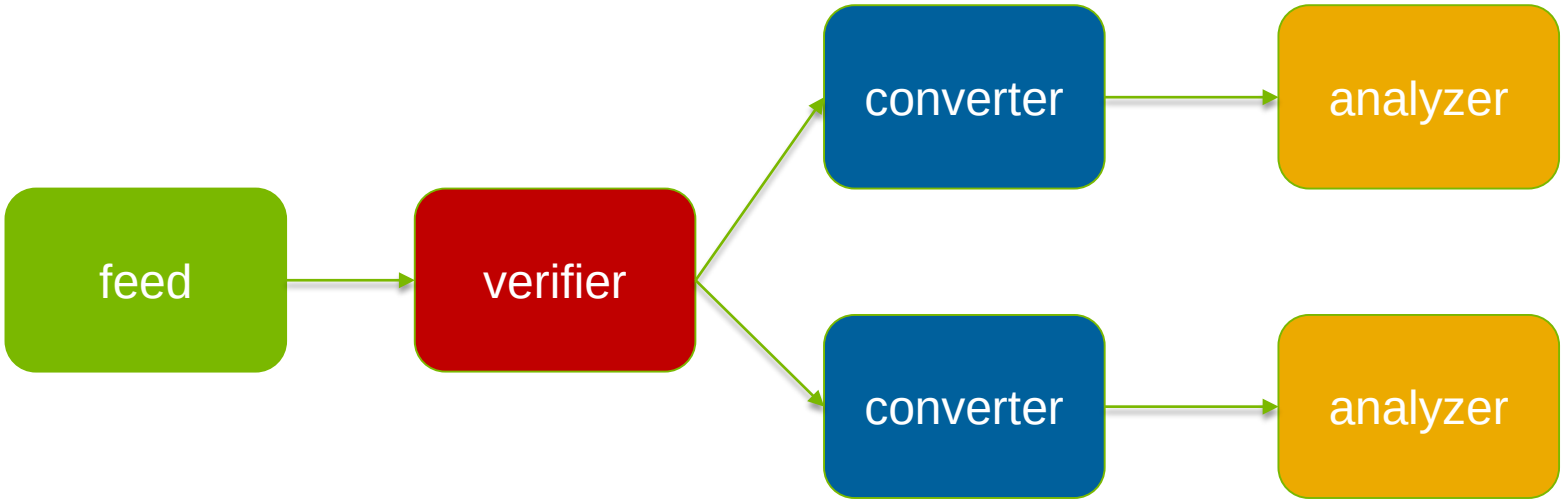
- For some samples certain projections are blocked, and the characteristic of the corresponding data frames is the same as ‘data dark’.
- By applying characteristic (limits) for ‘data’, the blocked frames can be found in a data file and reported. The indexes of blocked frames are used by reconstruction scripts.

DETECT SATURATION

1-ID High-energy x-ray diffraction

- The intensity of single points may exceed a limit (saturate)
- If there are many of such points, the experiment may be said unsuccessful
- The verifier can detect this condition in a data set
- With a new detector it may be possible to monitor the data in a real time, so the experiment can be stopped on bad quality

FUTURE DATA PROCESSING ARCHITECTURE

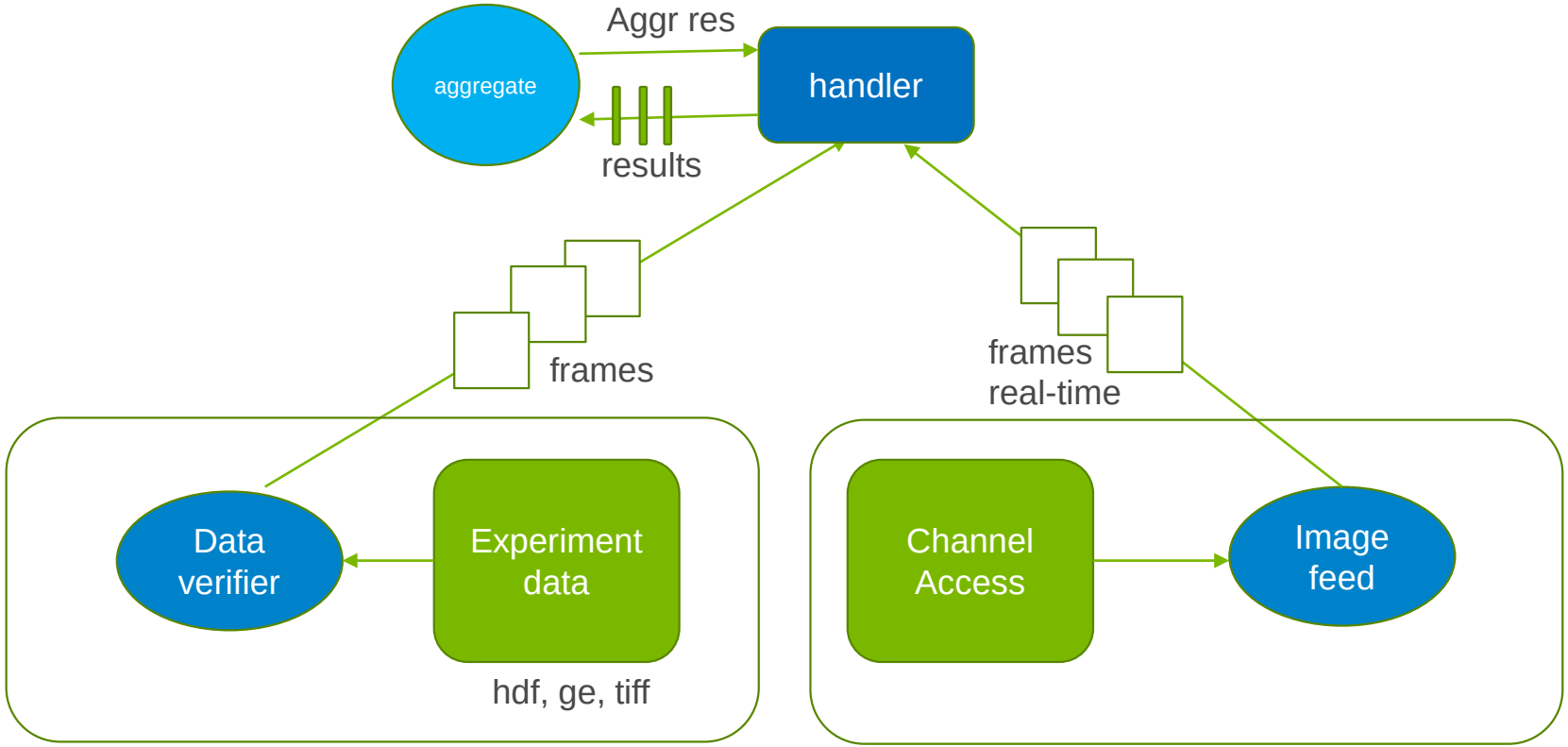


VERIFIERS

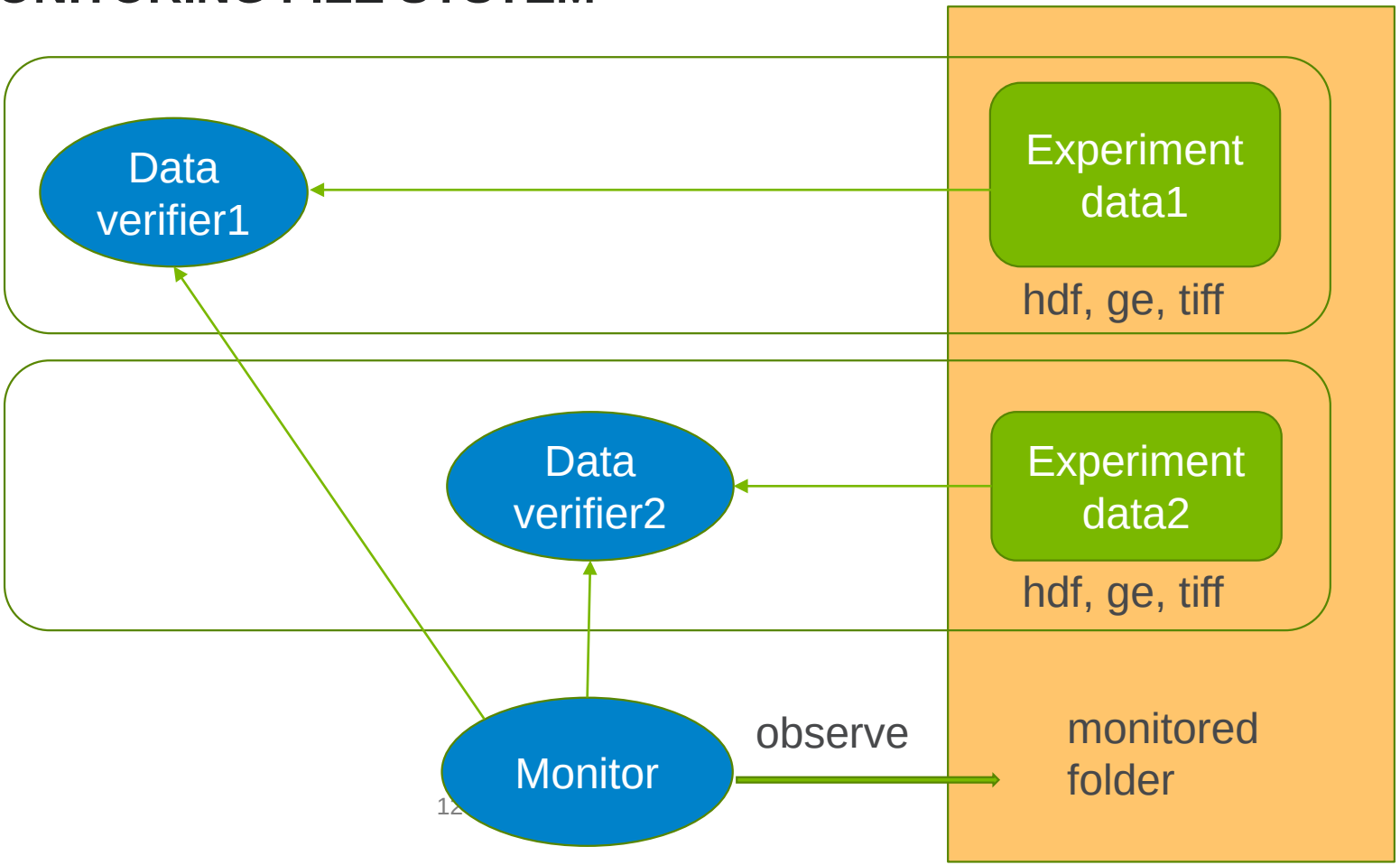
VERIFIERS

- PV – verifies settings of process variables; can be run before experiment, as well as periodically during experiment
- Quality results verifiers
  - Data verifier – verifies quality of experiment data in a file (hdf, ge, tiff)
  - Monitor – monitors directory for experiment data files, and runs data verifier on discovery
  - Structure – used for hdf type files. Verifies that the tags are in sync with gathered data.
- Real-time – verifies quality of experiment data while experiment is collecting data

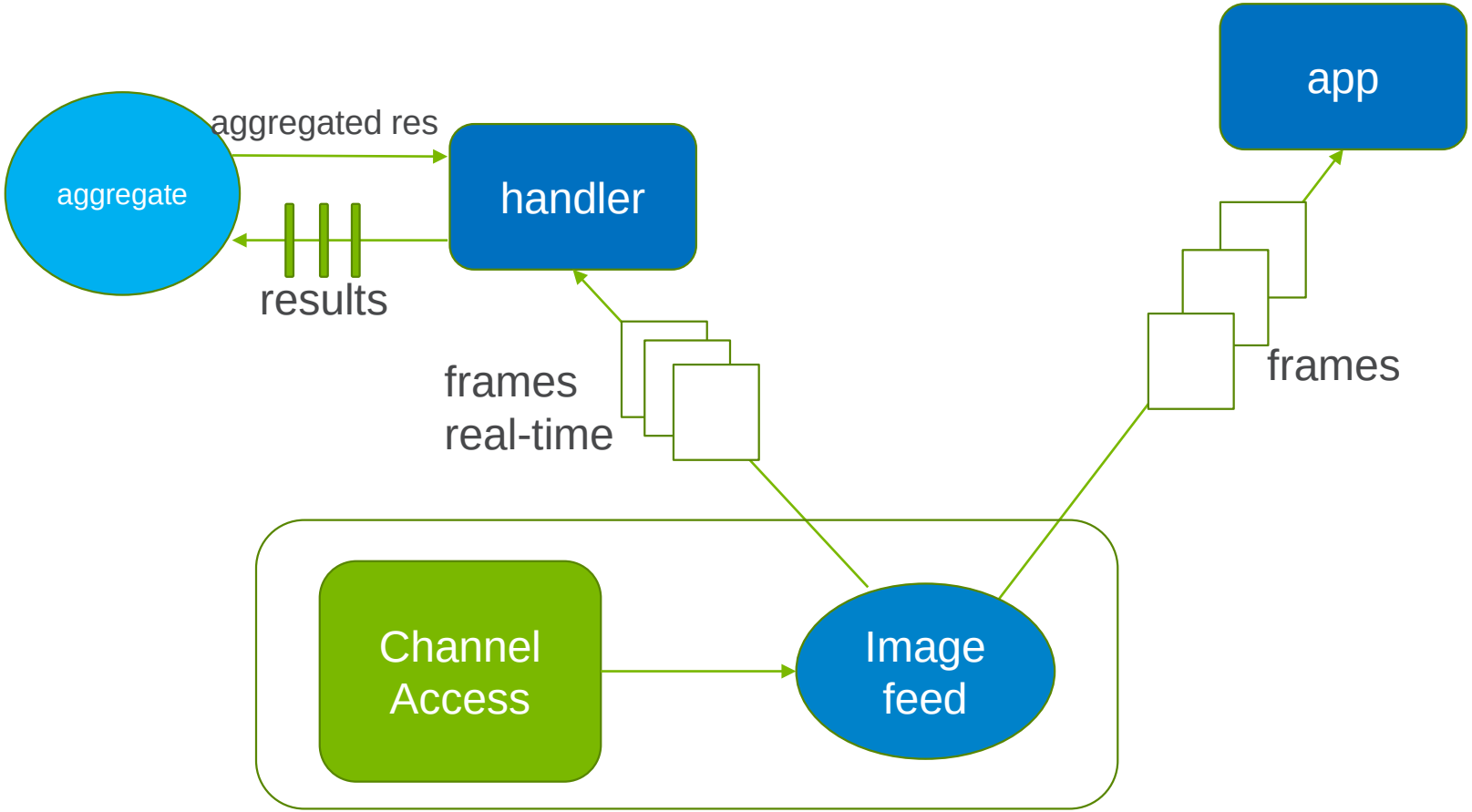
VERIFIER ARCHITECTURE



MONITORING FILE SYSTEM



LIVE FEED USAGE



QUALITY CHECKS

- Functions targeted to validate specific experiment data
- The checks are configured
- Easy to add quality check function to the framework
- Check on each frame (ex: mean value expected in certain bounds)
- Check on the collected so far frames (ex: number of saturation points exceeded limit)
- The same function can be used to assess experiment data taken by different detector by using different limits

CONFIGURATION EXAMPLE

- 'pv_file' = /home/beams/USR32IDC/.dquality/32id_nano/schemas/pvs.json
- 'limits' = /home/beams/USR32IDC/.dquality/32id_nano/schemas/limits.json
- 'quality_checks' = /home/beams/USR32IDC/.dquality/32id_nano/schemas/quality_checks.json
- 'time_zone' = America/Chicago
- 'extensions' = .hd5, .HD5, .hdf5, .HDF5, .h5, .H5
- #real-time verifier
- 'feedback_type' = console
- 'detector' = BBF1
- 'detector_basic' = cam1
- 'detector_image' = image1
- 'no_frames' = 20

PV.JSON

```
{"S:SRcurrentAI" : {
        "greater_than" : 60.0,
        "less_than" : 102.0 },
"ID32ds:Energy.VAL" : {
        "greater_than": 5.0,
        "less_than" : 30.0},
 "ID32us:Gap.VAL" : {
        "greater_or_equal" : 3.0,
        "less_or_equal" : 50.0},
    "32ida:BraggEAO.VAL" : {
        "greater_or_equal" : 5.0,
        "less_or_equal" : 25.0  } }
```

LIMITS.JSON

```
{  "data" : {
      "mean" : {
           "low_limit" : 400,
           "high_limit" : 600    },
     "stat_mean" : {
          "low_limit" : -150,
          "high_limit" : 150      },
      "std" : {
           "low_limit" : 150,
          "high_limit" : 200       }   }  }
```

QUALITY_CHECKS.JSON

{ “data” :
{ "QUALITYCHECK_MEAN" : ["STAT_MEAN"], "QUALITYCHECK_STD" : []},
“data_white” :
{ "QUALITYCHECK_MEAN" : [], "QUALITYCHECK_STD" : []},
“data_dark” :
{ "QUALITYCHECK_MEAN" : [] }
}

VERIFICATION RESULTS

- The results are provided in a text file. It lists the result for each frame, and for each check method.
- The output of verification is a list of frame indexes that did not pass the quality checks. The list can be used as an input to subsequent calculations.
- The real-time verifier provides immediate feedback. Currently the user can choose console and log file. There are plans to add live feedback on the experiment status web page.

HOW TO USE IT

- Under Documentation Examples tab there are code snippets for each verifier.
  - The examples are run with the assumption that a configuration file exists in a folder defined by “instrument”
- The github contains another suite of examples in a check.py and check_rt.py files
  - dquality.pv.verify(conf)
  - dquality.monitor.verify(conf, folder, int(num_files))
  - dquality.data.verify(conf, fname)
  - dquality.hdf.verify(conf, fname)
  - dquality.realtime.real_time.verify(conf, report_file, sequence)

LIVE IMAGE FEED

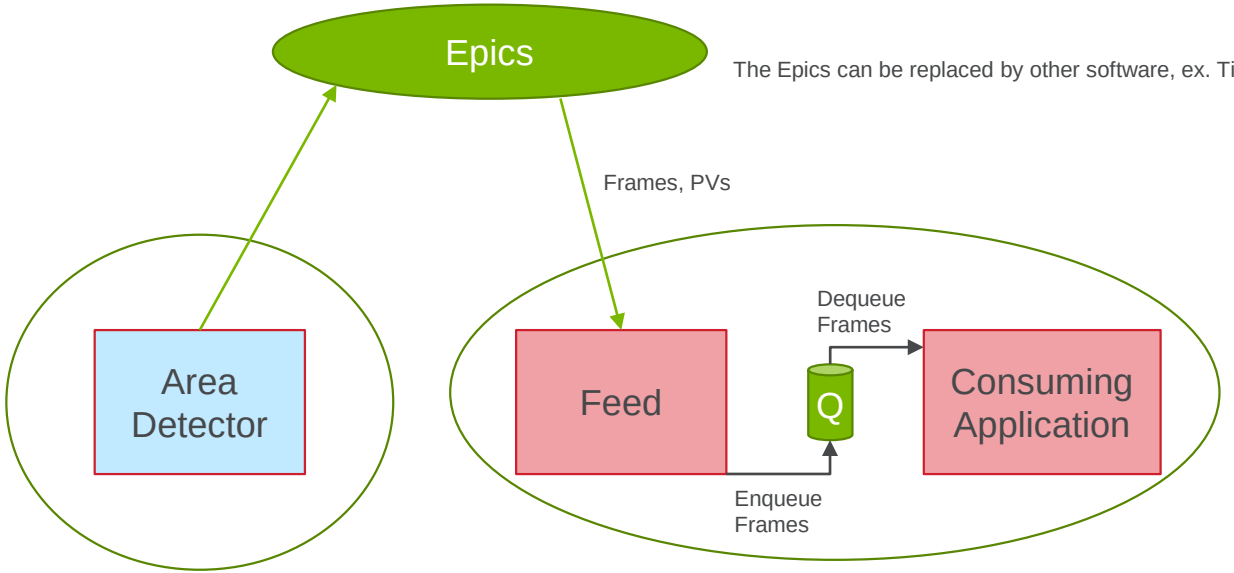
LIVE IMAGE FEED

- Uses Channel Access plugin
- Uses pyepics
- It provides a high level python API to retrieve image data at the time it is recorded
- User needs to write defined methods in an adapter.py file and define Area Detector prefixes in a configuration file

ADAPTER

- Acts as a link between feed and consuming application
- Has three methods that need to be implemented:
  - start_process: it parses arguments, and starts the consuming process. Passes in the queue on which the frames are received.
  - parse_config: reads configuration and sets required variables. Since the configuration is specific for the application, it is handled in the adapter.
  - pack_data: encapsulates the frame into object defined by a consuming application.

DESIGN



ADAPTER FOR REAL-TIME VERIFIER

- https://github.com/bfrosik/data-quality/blob/master/dquality/realtime/adapter.py

LINKS

- https://github.com/bfrosik/data-quality
- http://data-quality.readthedocs.org/
- http://cbs.fas.harvard.edu/usr/mcmains/CBS_MRI_Quality_Control_Workshop.pdf
- bfrosik@anl.gov

- Contact Barbara Frosik or Nicholas Schwarz for assistance in employing verifier on your beamline

QUESTIONS?
THANK YOU FOR YOUR ATTENTION