



# AIDA Beam Telescope Requirements for a Common DAQ

Hanno Perrey

- 1 AIDA Beam Telescope DAQ Requirements
- 2 EUDAQ 2.0
- 3 Device-under-Test Integration into EUDAQ 2.0

# From EUDET to AIDA Beam Telescope

## EUDET style: “Event-based”

- One trigger per slowest DAQ system in the telescope
- No triggers from TLU while at least one system is *BUSY* (e.g. Mimosa26 double-frame readout: 115.2  $\mu$ s)
- Data is written in single stream
- Low rates/data-taking efficiency for fast (LHC-type) DAQ systems

## AIDA style: “Particle-based”

- TLU issues trigger for every particle (i.e. scintillator signal)
- Triggers stop only on *VETO* from any DAQ
- **High-rate studies** possible
- **Much higher data-taking efficiency** at high-rate beams
- Telescope DAQ needs to cope with **high data rates** (e.g. continuous readout of Mimosa28 quad-planes)



Data in asynchronous streams from various DAQs

# Implications and To-do List for Telescope Framework

## Hardware

- Integrate with AIDA TLU (common clock, timestamp triggers)
- Mimosa-based planes with fast device (FEI4/TimePix) for timestamping of hits in reconstruction
- Continuous readout of Mimosa planes (if triggers present)

## DAQ Software (EUDAQ)

- Reduce disk IO/network bottlenecks: DAQs (can) store data locally
- Add all available timing (meta) information (from TLU/DAQ clocks) into data format
- Need hooks for online data verification and monitoring (next slides)

## Analysis Software (EU Telescope)

- Merge data streams from different DAQs based on timing information and hits (already done by many groups)
- Output timetagged tracks

## Reminder: EUDAQ Goals and Features

- Generic framework for data acquisition
- OS independent: Linux, Mac OSX, Windows
- Modular and flexible design
- Provides central **DAQ control**, **data handling and storage**, log collection, **online monitoring**
- Components communicate via TCP/IP and can run on different networked machines
- Focused on easy and flexible integration of the device under test including pre-existing DAQs
- All hardware communication done by “**Producers**” with equal rights

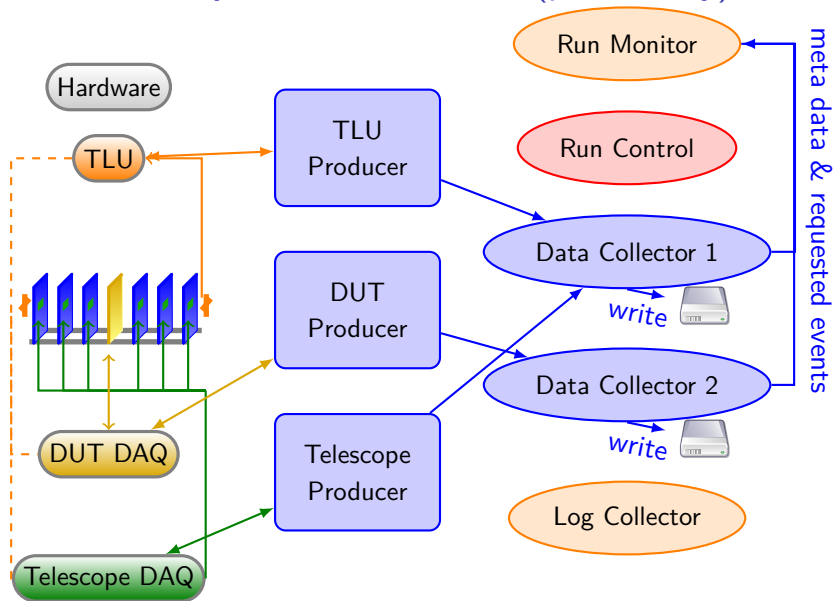
# AIDA Beam Telescope DAQ: EUDAQ 2.0

- Accepts data packets from DAQs covering:
  - ▶ single trigger (“classic mode”)
  - ▶ list of (timestamped) triggers
  - ▶ time range (shutter/data driven devices)
  - ▶ both time range and trigger list
- Writes data into multiple streams e.g. DAQ local storage
- Online verification in this scheme:
  - ▶ separate on-the-fly event building (full or partial for specific triggers)
  - ▶ cross-check timestamps e.g. against TLU information
  - ▶ merged data can be used for online monitoring/immediate offline analysis
- **Will be backward-compatible** to old integration efforts (no more than a recompilation should be needed)



Easier integration with a wider range of DAQ concepts

## Schematic Layout of EUDAQ 2.0 (preliminary)



# Integrating a Device into Telescope/EUDAQ 2.0

## TLU

- Receive clock, trigger and/or shutter signals
- **Timestamp events** (trigger, shutter open/close, ...)

## Producer to interact with RunControl and Hardware

- Implemented in C++, but Python interface exists (see backup)
- **Receives commands from RunControl** (Configure, Run Start/Stop)
- Talks to the DAQ hardware, receives data from there
- Sends its data to a DataCollector (**optional**)
- Gain flexibility with **new event format** and **multiple data streams**  
⇒ wider range of devices suitable for (full) integration

## Optional: Data Converter Plugin

- Converts native device format into defined structure
- Used for conversion into LCIO and for online monitoring

## Status of EUDAQ development

- A lot of maintenance has been done on EUDAQ over last months and is ongoing (git, CMake, platform-independence, Python interface, ...)
- RunControl can already assign a different DataCollector to each Producer and the resulting files can be merged offline
- Implementation of new event format started
- Best strategy for online (partial) event building being investigated
- A handful of developers committed to this task (both PhD and seniors)



## Summary and Outlook

- EUDAQ is a very flexible DAQ framework and has been used successfully for a long time with EUDET-family of telescopes
- Support one-trigger-per-particle operation of AIDA beam telescope requires extension of EUDAQ and removal of IO/network bottlenecks
- Key changes: timestamped events, multiple data streams
- Makes integration with EUDAQ 2.0 easier and more flexible than before
- Development on EUDAQ is ongoing, contributions are welcome!

Download the newest version and follow the development at  
<http://eudaq.github.io>

# Overview Backup Slides

- 4 Integrating a Device under Test into EUDAQ
- 5 Running EUDAQ

# Integrating a DUT into EUDAQ

## The Anatomy of a *Producer*

- A Producer needs to implement command receiving methods of the Producer base class:
  - ▶ OnConfigure, OnStartRun, OnStopRun, Terminate
- It configures the hardware according to the config received from RunControl
- It (optionally) sends its data to the data collector:  
either in raw format or converted to a custom StandardEvent class
- It (optionally) logs status/error messages

Example code is provided!

→ see also shortened examples on the following slides

## Example C++ (Pseudo-) Code for a EUDAQ Producer

```

class ExampleProducer : public eudaq::Producer {
public:
    ExampleProducer(){}
    virtual void OnConfigure(const eudaq::Configuration & config) {
        // .... configure your hardware
    }
    virtual void OnStartRun(unsigned param) {
        // .... prepare for and start run
    }
    virtual void OnStopRun() {
        // .... stop your DAQ
    }
    void ReadoutLoop() {
        while (true) {
            // while running:
            // ..... get raw data, put it into RawDataEvent and send
        }
    }
};

int main(int /*argc*/, const char ** argv) {
    ExampleProducer producer(); // Create a producer
    producer.ReadoutLoop();     // And set it running...
    return 0;
}

```

## Example Python Code for a EUDAQ Producer

```
#!/usr/bin/env python2
execfile('PyEUDAQWrapper.py') # load the ctypes wrapper
from time import sleep
import numpy # for data handling
# create PyProducer instance
pp = PyProducer("testproducer", "tcp://localhost:44000")

# wait for CONFIGURE cmd from RunControl
while not pp.Configuring:
    sleep(1)
# ... do your config stuff here ...
pp.Configuring = True
# check for RUNSTART cmd from RunControl
while not pp.StartingRun:
    sleep(1)
# ... prepare your system for the immanent run start
pp.StartingRun = True
# starting to run main DAQ loop
while not pp.Error and not pp.StoppingRun and not pp.Terminating:
    # prepare an numpy array for raw data storage
    data = numpy.ndarray(shape=[1,3], dtype=numpy.uint64)
    # .... get your data from your hardware ...
    pp.SendEvent(data) # send event off
```

# Running EUDAQ

- OS independent (Linux, Mac OSX, Windows)
- Components can be run on different networked machines
- Different interfaces available: Qt GUI, console, Python

The screenshot displays three main EUDAQ interfaces:

- eudaq Run Control:** A control panel with fields for Config (n\_cms\_coins), Run (Start/Log/Stop), GeoID (1), and Status (Run Number: 1569, Events Built: 0, Rate: 0.000000, File Bytes: 0.000000, TLU Status: 0.0, 0.0, 0.0). It also shows a table of connections and a Data Collector status window.
- EUDET Telescope Online-Monitor 1.0beta21:** A monitoring interface with a tree view on the left and six heatmaps on the right. The heatmaps show correlations between different detector channels. A large red watermark "Online Monitor" is overlaid on the top-right heatmap.
- EUDAQ Log Collector:** A log viewer showing a table of system events. A large red watermark "Log Monitor" is overlaid on the table.

| type          | name      | state       | connection        |
|---------------|-----------|-------------|-------------------|
| DataCollector |           | OK          | 192.168.2.3:55187 |
| LogCollector  |           | OK          | 192.168.2.3:55182 |
| Monitor       | OnlineMon | OK          | 192.168.2.3:55196 |
| Producer      | TLU       | OK: Started | 192.168.2.3:55194 |
| Producer      | MimosA    | OK: Started | 192.168.2.3:55191 |

| Level  | From | Search | Received     | Sent         | Level  | Text  | From            | File                                    | Function                                |
|--------|------|--------|--------------|--------------|--------|---|-----------------|---|---|
| 4-INFO | All  |        | 11.16.56.721 | 11.16.56.721 | 4-INFO | Connection from Producer TLU (192.168.2.3:49944)      | DataCollector   | DataCollecto...                         | OnConnect(const eudaq::ConnectionInfo&) |
|        |      |        | 11.16.56.722 | 11.16.56.721 | 4-INFO | Connection from Monitor OnlineMon (192.168.2.3:57631) | LogCollector    | eULog.hh:97                             |   |
|        |      |        | 11.20.11.874 | 11.20.11.874 | 4-INFO | Configuring (n_cms_coins)                             | RunControl      | RunControl.c...                         | Configure(const std::string&, int)      |
|        |      |        | 11.20.14.205 | 11.20.14.205 | 4-INFO | Configured (n_cms_coins)                              | Producer...     | OnConfigure(const eudaq::Configuratio&) |   |
|        |      |        | 11.33.17.380 | 11.33.17.380 | 4-INFO | Configuring (n_cms_coins)                             | RunControl      | RunControl.c...                         | Configure(const std::string&, int)      |
|        |      |        | 11.33.19.762 | 11.33.19.762 | 4-INFO | Configured (n_cms_coins)                              | Producer:TLU    | TLUProducer...                          | OnConfigure(const eudaq::Configuratio&) |
|        |      |        | 11.33.36.477 | 11.33.36.477 | 4-INFO | Configuring (n_cms_coins)                             | ProducerMim...  | NProducer.c...                          | OnConfigure(const eudaq::Configuratio&) |
|        |      |        | 11.33.36.736 | 11.33.36.736 | 4-INFO | Configured (n_cms_coins)                              | RunControl      | RunControl.c...                         | StartRun(const std::string&)            |
|        |      |        | 11.33.52.635 | 11.33.52.635 | 4-INFO | Starting Run 1569                                     | DataCollector   | DataCollecto...                         | OnPrepareRun(unsigned int)              |
|        |      |        | 11.33.53.637 | 11.33.53.637 | 4-INFO | Preparing for run 1569                                | Monitor:Onli... | Monitor.cc:68                           | OnStartRun(const int&)                  |
|        |      |        | 11.33.59.651 | 11.33.59.651 | 4-INFO | Starting Run 1569                                     |                 |   |   |