Data acquisition software

Wing, M. (UCL) et al

06 December 2017
Abstract:
The overarching software chosen to provide the data acquisition (DAQ) for beam tests using multiple linear collider detector prototypes is the EUDAQ framework. The EUDAQ software was originally developed for the EUDET beam telescope and provides a flexible framework which has been used in many different beam tests. In order to provide the necessary flexibility to be able to control and readout data for the varying types of linear collider detector prototypes, the EUDAQ software has undergone a significant update, called EUDAQ2. The software supports detectors with different trigger schemes and very different readout speeds and this has been validated in beam tests with multiple detectors. Along with EUDAQ2, DQM4HEP, forms the basis of the software developed in this workpackage for use in detector beam tests. DQM4HEP has been developed to monitor the data quality and environmental conditions. Both codes have undergone significant development, been benchmarked and tested in various conditions, already used in common beam tests with multiple detectors, and documented for further use. This report summarises the two software codes.
AIDA-2020 Consortium, 2017
For more information on AIDA-2020, its partners and contributors please see www.cern.ch/AIDA2020

The Advanced European Infrastructures for Detectors at Accelerators (AIDA-2020) project has received funding from the European Union’s Horizon 2020 Research and Innovation programme under Grant Agreement no. 654168. AIDA-2020 began in May 2015 and will run for 4 years.

Delivery Slip

<table>
<thead>
<tr>
<th>Authored by</th>
<th>Name</th>
<th>Partner</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M. Wing</td>
<td>UCL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>R. Ete</td>
<td>DESY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y. Liu</td>
<td>DESY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>J. Kvasnicka</td>
<td>DESY</td>
<td></td>
</tr>
<tr>
<td>Edited by</td>
<td>L. Lapadatescu</td>
<td>CERN</td>
<td>25/11/2017</td>
</tr>
<tr>
<td>Reviewed by</td>
<td>F. Sefkow [Scientific coordinator]</td>
<td>DESY</td>
<td>01/12/2017</td>
</tr>
<tr>
<td>Approved by</td>
<td>F. Sefkow [Scientific coordinator] Steering Committee</td>
<td>06/12/2017</td>
<td></td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

1. INTRODUCTION

2. THE EUDAQ SOFTWARE
   - EUDEAQ2 SOFTWARE CONCEPT
   - FINITE-STATE MACHINE AND RUN CONTROL IN EUDEAQ2
   - USE OF EUDEAQ2 IN BEAM TESTS

3. THE DQM4HEP SOFTWARE
   - DQM4HEP CONCEPT
   - USE OF DQM4HEP IN BEAM TESTS

4. SOFTWARE RELEASES
   - EUDEAQ2 RELEASE
   - DQM4HEP RELEASE

5. SUMMARY AND OUTLOOK

6. REFERENCES

ANNEX: GLOSSARY
Executive summary

This report describes the two main software codes developed for common data acquisition. They have both undergone significant development and testing and this is summarised. In Section 1, an introduction to the deliverable and the codes is given. In Section 2, the EUDAQ2 software for controlling the data to and from the detector is described. This contains a functional description and also the applications it has been used in so far. In Section 3, the DQM4HEP software used for monitoring the data quality and environmental conditions of the beam tests is described. This section also contains details of the code’s functionality as well as its successful beam tests. In Section 4, details of the software releases are given. In Section 5, a brief summary is given and plans and outlook for the software.

1. INTRODUCTION

Two codes, EUDAQ2 and DQM4HEP, underpin the software of the data acquisition package. The EUDAQ2 software controls the passing of data to and from a detector or combination of detectors. It can control a given detector directly or interface to a detector-specific data acquisition (DAQ) system. The DQM4HEP software is a flexible and generic code that can be used by a given detector or combination of detectors to monitor the data quality (DQM) or the environmental conditions (slow control).

During the initial phase of this project the EUDAQ software was upgraded from version 1 to 2. The new version provides a more generic package, not linked to a given piece of hardware, improved the modularity, as well as being able to read out detectors with very different timing or triggering structures. The EUDAQ2 software has been used in several different beam tests, initially validating the EUDAQ2 software, and now being used as the central system.

The monitoring of data quality is an essential aspect of any particle physics experiment or beam test. Usually tools to perform DQM are developed for a given system and experiment or beam test. The software framework DQM4HEP has been developed such that it should be able to allow all groups to use it as their DQM framework. It should be generic and portable.

Several milestone reports have already documented the various advances in the software and this report brings them together and summarises these developments. Where new work has happened since the previous reports, this is also highlighted. A particular strength of the new software is that they have already been used in common beam tests involving multiple detectors. This demonstrates that they have met the goals of a common software and fulfil the requirements of this deliverable.
2. THE EUDAQ SOFTWARE

The EUDAQ2 software has been described in milestone reports MS46 [1] and MS62 [2] as well as in the user manual available from a dedicated web-site [3]. In this section, these are summarised as well as the developments noted since the milestone reports.

2.1. EUDAQ2 SOFTWARE CONCEPT

The EUDAQ software was originally developed for the EUDET beam telescope and provides a flexible framework which has been used in many different beam tests. However, in order to provide the necessary flexibility to be able to control and readout data for the varying types of linear collider detector prototypes, the EUDAQ software has undergone significant update, called EUDAQ2. EUDAQ version 1 was originally developed for use with the EUDET beam telescopes [4] and linked to the trigger logic unit (TLU) control system. EUDAQ1 has been used in beam tests where the telescope provided an external reference point for the detector under test. However, the software was not independent of the beam telescope and a strong requirement on EUDAQ2 was to provide a truly generic framework. The development also provided a more modular framework using modern C++11.

The other strong driver for EUDAQ2 is the ability to run with different kinds of triggering or event-building methods and then combine these to produce a common event for a given beam particle interaction with a multiple-detector system. Linear collider detectors have potentially very different characteristics, for example, detectors can be externally of self-triggered and/or have very different rate capabilities (i.e. continuous data taking or with a fixed readout). EUDAQ2 can cope with these different circumstances and provide a common and coherent event for each particle interaction.

Clearly a given detector can use EUDAQ2 as its dedicated DAQ system and this makes the integration with other detectors simpler. However, this is not imposed and it is assumed that a detector may have its own DAQ system and EUDAQ2 will interface to this so that integration with other detectors is possible. The two possibilities, of being able to be a detector DAQ or interface to a detector-specific DAQ system, is an extra capability and flexibility built into EUDAQ2’s design. Although this DAQ concept is focused on working for linear collider detectors, it can also be used by other detectors from other projects should they be able to meet the interface specifications [5] to the DAQ system developed here. The hardware and software are freely available and so can be used by other detector communities.

The overall architecture of the EUDAQ2 software is shown in Figure 1. The run control connects to the following components. Each hardware that produces data will have a producer which initialises, configures, stops and starts the hardware by receiving commands from the run control. It also reads out data and sends it to one or more data collectors. The data collector sends data to disks for storage; the data may or not be converted [6] and these data as well as the raw data may be stored. The data collector may also send data to a monitor which generates online monitoring plots for display. The log collector receives messages from all processes and displays them to the user, as well as writing them to a file.

Since initial release of EUDAQ2, the software has undergone some small updates and improvements, correcting bugs and improving performance, such as having a multi-threaded data collector. Log and status messages have been unified and formatted consistently to aid the user. A python interface has been introduced so that data analysis can be performed using python. This is encompassed in a new version, EUDAQ2.1.
2.2. FINITE-STATE MACHINE AND RUN CONTROL IN EUDAQ2

A finite-state machine (FSM) is a computational model in which a machine or component can be in only one of a finite number of states. Changing the current state is implemented via pre-defined transitions. An FSM is implemented in the run control, which then controls the states and transitions of a given detector or piece of hardware. Obvious examples of the states detectors can be in are configured, running or error. Initially EUDAQ had one additional state, unconfigured. These four states, however, had their limitations and an additional state, uninitialised, was introduced, in response to requests from the community. This separates, for example, the setting up of different parameters and initialisation of the hardware. As the hardware initialisation can often be time-consuming it is done at the beginning and not repeated each for each new reconfiguration.

The new version of the FSM is shown in Figure 2 [7]. The state of the run control is determined by the lowest state of the connected client in the following priority: Error, Uninitialised, Unconfigured, Running. This means, for example, that even if only one connection is in the Error state, the whole machine will be in that state. This prevents such mistakes as running the system before every component has finished the configuration.

The run control is the process which manages the full EUDAQ system. There should always be only one instance of the run control for a runtime setup. All other data taking processes must have the network location of the run control and announce themselves to the run control. The run control takes in the paths of the ini/config files and picks up and sends necessary information to each of the connected processes. As the front-end point to users, the run control will wait for user input by command line, or more usually, a GUI button click and issue e.g. a data taking command to the whole EUDAQ system.

As shown in Figure 1, the run control is central to the EUDAQ system and links to all other processes. The run control GUI window is shown in Figure 3 for an example setup. Depending on the status of a given process, they can be initialised, configured or re-configured, data taking can be started and...
stopped and the software can be terminated. With the FSM state from each of the connected processes, the run control determines a system FSM state depending on a user defined logic or default behaviour. Details of the use of the run control are given in the EUDAQ manual [3]. The possible states are given by the FSM, and Figure 3 shows how these states are controlled.

![Figure 2: The finite-state machine in the EUDAQ2 software [6].](image)

The extra flexibility allowed by the new FSM and run control are the improvements for the EUDAQ2 run control. This allows it to be used in more complex beam tests as discussed later in this report.

![Figure 3: Screen shot of the run control operating with an example setup.](image)
2.3. USE OF EUDAQ2 IN BEAM TESTS

The EUDAQ2 software was validated and thoroughly tested in a beam test carried out at DESY with the AHCAL and beam telescope as well as a TLU and a beam interface (BIF) module to record the beam timestamp. The principal challenge in this set up and that which EUDAQ2 is particularly designed to overcome is the very different triggering of the two main components. The AHCAL is self-triggered and oriented to the ILC timing structure of spills with high activity interspersed with significant periods of dead time. The beam telescope, however, is externally triggered and data read out with a continuous rolling shutter. Beam tests were performed during October and December 2016 and February 2017 in which the systems were synchronised and the firmware and software debugged.

The control hardware for the AHCAL (and other CALICE calorimeters) is the so-called clock and control card (CCC) which needed modification to be able to connect to the TLU which is the central control unit for future beam tests of multiple linear collider detectors. Other new developments were new producers for the AHCAL, beam telescope and TLU. Given the complexity of events, a new event builder was also required which also meant a new data collector was written. However, the data format used previously was maintained so that DQM4HEP could be used for monitoring.

Validation of the system and the use of the EUDAQ2 software is shown in Figure 4. This shows an image of the correlation between the spatial hits in the beam telescope and in the AHCAL. The correlations remained stable over long runs and over multiple runs.

![Correlated](image.png)

*Figure 4: Image showing correlation of hits in the beam telescope and in the AHCAL.*

The EUDAQ2 software has also been used for a beam test during May and June 2017 of the ATLAS inner tracker (ITK) upgrade. The Producers and DataCollectors for the EUDET Telescope, FE-I4 pixel chip reference plane and ITK strip module were updated in order to be interfaced to the EUDAQ2 software. A user specific Run Control, was implemented to support the automatic restart of data taking and the possibility to do parameter scans. To bridge the hardware DAQ of the ITK module and EUDAQ2, a ROOT interface was written. Since none of the telescope, FE-I4 reference plane and ITK strip module generate a timestamp, the EUDAQ DataCollector synchronised events by trigger number. The DataCollector could be configured to write native EUDAQ data or converted LCIO object data online. The legacy OnlineMonitor to graphically monitor the data quality was used. Although not a requirement of work-package, this demonstration of the use of EUDAQ2 with detectors from other communities, shows its flexibility and provides an extra verification of its
DATA ACQUISITION SOFTWARE

Deliverable: D5.3
Date: 06/11/2017

validity. The EUDAQ software has been used since in another ITK beam test in DESY in October 2017. The beam test was to analyse the charge sharing in the silicon sensors. The EUDAQ2 software performed well.

3. THE DQM4HEP SOFTWARE

It is important that data can be both examined and checked for its quality in real-time during a test beam. This is useful both for the understanding of hardware and for shifters who are taking data. Anomalous information can indicate such things as faulty wiring, unresponsive or dead channels, damaged detector components, and many other hardware faults or irregularities that are important to identify. While shifters are operating, online monitoring allows them to identify issues such as incorrect beam energy or placement, or reduced beam rates, both of which can affect the rate of data taking or the quality or usefulness of the information.

Extensive technical documentation on DQM4HEP and its use for data quality monitoring is given in milestone reports MS67 [9] and MS68 [10].

3.1. DQM4HEP CONCEPT

DQM4HEP is designed to be able to fulfil the requirements of data quality and slow control monitoring in a generic way. The structure of the program allows for independent components of the framework to be used, not used, or exchanged, by isolating the sections of the program that are specific to the user or use-case. The components that are case-specific – the analysis and standalone modules – are written in standard C++ code, meaning they are capable of performing any data unpacking, processing or analysis that is necessary; the framework then handles packaging this information in a useful way and networking to transmit it to where it is needed.

The flexibility of the framework is key to it being generic. The event data model (EDM) abstraction means that the framework does not need to have special rules for handling particular types or structures of events, which is what allows it to handle anything that can be packed into, decoded from, and accessed by normal C++ methods. The ability to run multiple instances of each process of the framework is also key to its flexibility. This allows users to, for example, run different analyses for different detector sub-components, operate in online or offline modes, and run multiple analyses in parallel or distributed over several networked computers to reduce computational load. These features are all useful for online DQM. Online data quality monitoring is a critical component of test beam software, necessary for real-time response to conditions in the test beam and helpful for further developing the software and understanding the response of the hardware. Offline mode allows data to be "re-run" through monitoring and DQM tests to compare discrepancies that may not have been identified during the online run, as well as to iterate and test new or improved quality tests on old data to compare their results.

The global online architecture is shown in Figure 5. Each box represents a separate process which are linked by network communication, either TCP/IP or HTTP. The TCP/IP communication is handled by DIM [11], a light software developed by the DELPHI experiment, and the HTTP communications by Mongoose [12]. The colour code refers to the different users that have to interact within the framework while deploying it for a given setup. The DAQ box on the left is tagged as an engineer task as the link between the DAQ and the DQM4HEP framework; this depends on the EDM and event streaming. This is usually implemented once by the DAQ engineer who is usually more familiar with low level programming. The blue boxes are DQM internal processes. The run control server is a single process within a setup that receives the start of run and end of run signals from the DAQ and forwards them to listening DQM4HEP applications. The current protocol in use to receive
the signals is HTTP. Other simpler software can provide this functionality and be easily embedded in a DAQ without having the DQM4HEP dependency. The event collector servers are the first entry points in the framework. Multiple collectors can be run independently within a setup. For example, one could collect events coming from the DAQ event builder and others from the different sub-detectors events. The monitor element collector servers are the last exit points of data before inspection by the shifters. They collect monitor elements coming from user analysis and redistribute them to shifters on visualization interfaces. Again, multiple monitor element collectors can be run within a deployment. The different analyses can thus send their monitor elements to different collectors, reducing the load in terms of memory, bandwidth and CPU for single collectors. The orange boxes refer to so-called "DQM modules". These applications process data and produce monitor elements that are collected by monitor element collectors. As the setup is specific to an experiment, the analyses are also specific and thus have to be developed according to the data sent by the DAQ. As for the EDM, the analyses are encapsulated within plugins. They are loaded at runtime by a dedicated application. The boxes at the bottom of Figure 5 (job control servers and job control GUI) are related to job control functionalities. Indeed, the picture shows that during a deployment, the number of involved processes to manage is important. Starting manually each process on different hosts can be a heavy and painful task. To efficiently manage all these programs, job control servers are run as UNIX daemons on each involved host of the setup and manage the processes (start, stop, get status, get log files, ...). All these servers are steered remotely by the shifters through a dedicated user interface. Finally, the green boxes are the endpoints of the system: the visualization interface.

Figure 5: The global online architecture of DQM4HEP.
3.2. USE OF DQM4HEP IN BEAM TESTS

DQM4HEP has already been used in combined beam tests for multiple-detector prototypes: the AHCAL and beam telescope; and SDHCAL and SiWECAL.

During SDHCAL beam tests, DQM4HEP was interfaced with the data acquisition system using a shared memory (shm) feature, allowing it to access information online. In AHCAL beam tests, the framework was used in "nearly-online" mode; completed runs in LCIO format were accessed by the LCIO file service over network-attached storage as soon as an individual run file was finished. The file service can be run on files as they are being written but as events are loaded into memory, only events that were present in the file at that time were available for monitoring.

DQM4HEP was extremely useful as a tool to identify issues with the detectors during testing. For example, during AHCAL tests, where new scintillator tile layers were being tested for the first time, the hitmaps allowed quick and simple visual identification of any scintillator tiles whose electronics were noisy or dead. These were plainly visible as erroneous hits for noisy channels, or blank squares for dead channels. In SDHCAL tests, the current in the resistive plate chambers is expected to be near 0 while taking data. Thanks to the monitoring it was possible to prevent starting bad runs when the monitored current was too high, saving time and various resources. Hit maps of many GRPCs of the SDHCAL were recorded during a test beam. Some of the chambers show a lack of hits in their centre. This was due to an overflow of incoming gas in the chamber, which leads to its inflation and thus explains this lack of hits. By looking at these maps, the shifters were able to spot the problem, correct for this and restart a new run with a more stable detector.

Figure 6 shows a variety of plots from beam tests, demonstrating the use of the framework both as a monitoring tool and a slow control tool. Figures 6 (top-left), (top-right), (bottom-left) and (bottom-right) are from a combined SDHCAL+SIWECAL beam test in June 2016 at the CERN-SPS. Figure 6 (middle-left) is from an AHCAL test at the CERN-SPS beam in May 2017, and Figure 6 (middle-right) is from a combined AHCAL and beam telescope test at DESY in February 2017.

4. SOFTWARE RELEASES

4.1. EUDAQ2 RELEASE

The release of the EUDAQ2 software is publicly available. Information on the EUDAQ software, both EUDAQ2 and the previous versions, can be found on the dedicated web-site:

http://eudaq.github.io

where information on downloading the source code is given; this can also be found at:

https://github.com/eudaq/eudaq

The accompanying manual can also be found on the EUDAQ web-pages or downloaded directly at:


Questions and issues should be addressed to the EUDAQ developers team using the support and bug tracker page:

https://github.com/eudaq/eudaq/issues
Figure 6: Various plots from the monitoring interface recorded during AHCAL and SDHCAL beam tests. (top-left) Hit map of layer six of the SDHCAL with the beam off, showing noisy channels. (middle-left) A plot showing a hadronic shower within the SDHCAL. (top-right) Hit map of a single layer of the AHCAL. (middle-right) Spatial correlation plots of the AHCAL and beam telescope. (bottom-right) A plot of the ambient air pressure around the SDHCAL detector. (bottom-left) A plot of applied high voltage (HV) on each layer of the SDHCAL detector.
4.2. DQM4HEP RELEASE

Most of the information related to the DQM4HEP software can be found on the Github collaboration page:

https://github.com/DQM4HEP

The software version on which this document is based can be found on the release page:

https://github.com/DQM4HEP/dqm4hep/releases/tag/v01-04-04

Issues, questions and more general support from the DQM4HEP team can be found on this page:

https://github.com/DQM4HEP/dqm4hep/issues

A dedicated email has also been setup for user support:

dqm4hep@gmail.com

5. SUMMARY AND OUTLOOK

The EUDAQ2 software is released as the basic software framework for use in beam tests with multiple linear collider detectors. The software is freely available for download and is accompanied by documentation and a manual to aid the user. The software has been shown to work in a beam test involving the CALICE analogue hadronic calorimeter and the EUDET beam telescope along with other devices and detectors. The software has also been used in ATLAS pixel detector upgrade beam tests. The EUDAQ software was used in a CMS HGCAL and CALICE combined beam test with the intention to upgrade to EUDAQ2 for future beam tests. The DQM4HEP framework provides a reliable method for online monitoring and data quality monitoring for physics test beam data that is generic, flexible and scalable. It has proven that it can adapt to different detector types, including detectors with different event and read-out structures. It has also proven that it is capable of handling common test beams with more than one detector, correlating the information received from the data acquisition to create a picture of what is happening in all detectors. The use of the two software codes, EUDAQ2 and DQM4HEP, in beam tests and their availability to the wider community constitute the meeting of this deliverable. The software will be used in future beam tests for other Linear Collider detectors and is available for other systems as well.
6. REFERENCES


## ANNEX: GLOSSARY

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAQ</td>
<td>Data acquisition</td>
</tr>
<tr>
<td>EUDAQ</td>
<td>General name for the family of data acquisition software</td>
</tr>
<tr>
<td>EUDAQ2</td>
<td>Principal software for data acquisition for linear collider detector beam tests</td>
</tr>
<tr>
<td>DQM</td>
<td>Data quality monitoring</td>
</tr>
<tr>
<td>DQM4HEP</td>
<td>Principal software for data quality monitoring for linear collider beam tests</td>
</tr>
<tr>
<td>TLU</td>
<td>Trigger logic unit used to control detectors, providing a trigger and/or timing</td>
</tr>
<tr>
<td>AHCAL</td>
<td>Analogue hadronic calorimeter prototype detector</td>
</tr>
<tr>
<td>CALICE</td>
<td>Calorimeter R&amp;D for Linear Collider detectors collaboration</td>
</tr>
<tr>
<td>CCC</td>
<td>Clock and control card for Linear Collider calorimeters</td>
</tr>
<tr>
<td>ITK</td>
<td>Inner tracker detector of the ATLAS experiment</td>
</tr>
<tr>
<td>FE-I4</td>
<td>Pixel chip used here in the ITK beam test</td>
</tr>
<tr>
<td>EUDDET</td>
<td>EU project on Detector R&amp;D towards the International Linear Collider</td>
</tr>
<tr>
<td>SDHCAL</td>
<td>Semi digital hadronic calorimeter prototype detector</td>
</tr>
<tr>
<td>SiWECAL</td>
<td>Silicon–tungsten electromagnetic calorimeter prototype detector</td>
</tr>
<tr>
<td>LCIO</td>
<td>Linear Collider input/output persistency framework and event data model for linear collider detector studies</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>Transmission control protocol / internet protocol</td>
</tr>
<tr>
<td>DIM</td>
<td>Distributed information management</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext transfer protocol</td>
</tr>
<tr>
<td>Mongoose</td>
<td>An HTTP client and server library written in C/C++</td>
</tr>
<tr>
<td>HGCAL</td>
<td>High granularity calorimeter being developed for the CMS upgrade</td>
</tr>
<tr>
<td>ATLAS</td>
<td>A toroidal LHC apparatus experiment at the LHC</td>
</tr>
<tr>
<td>CMS</td>
<td>The compact muon solenoid experiment at the LHC</td>
</tr>
</tbody>
</table>