Scientific Data Acquisition Service Level Description

FRIB-S30206-RC-008646-R001

Issued 4 August 2023

Prepared by

8/4/2023

X

Giordano Cerizza
Scientific Data Acquisition Group Leader
Signed by: 5eea326e-b12e-4a74-91af-a747ef366434

Reviewed by

8/7/2023

X

Sean Liddick
Associate Director for Experimental Science
Signed by: liddick

Noted by

8/5/2023

X

Jill Berryman
Manager for User Relations
Signed by: berrymaj

Approved by

8/4/2023

X

Georg Bollen
Experimental Systems Division Director
Signed by: bollen

Table of Contents

Revision History

Revision	Issued	Changes
R001	4 August 2023	Original Issue

Authorizing Document

None.

Authorized Documents

None.

Authorized Committees and Boards

None.

Named Program Roles

None.

Awareness Training

None.

Enabling Training

None.

1 Objective

This document describes the level of service FRIB will be able to provide for Scientific Data Acquisition for user experiments as well as related FRIB user responsibilities.

2 Definitions

SDAQ: Scientific Data Acquisition.

FRIBDAQ: General data acquisition frameworks for both data acquisition and analysis.

ESD: Experimental Systems Division.

DAQ: Data Acquisition.

DAQ Software: Software necessary to acquire data from an experimental system and store to disk.

PAC: Program Advisory Committee (PAC) is in charge of approving experiments that were proposed and recommends for beam-time allocation to the Director. Each approved experiment has a designated Spokesperson and an alternate Spokesperson.

ERR: The Experimental Readiness Review (ERR) establishes the suitability to schedule an experiment and defines the experiment scope and configuration. All experiments must go through the FRIB ERR process before scheduling [1].

3 Overview

The SDAQ Group within ESD develops, maintains, and supports software and hardware solutions to fulfill the needs and the requirements of the FRIB Scientific Experimental Program. FRIBDAQ is a software suite that provides a flexible and extensible framework for handling the data flow produced by nuclear physics experiments. It aims to solve the top-level problem of managing the data stream by breaking it down into smaller problems solved by smaller applications. Therefore, it is a collection of tools that can be assembled into more complicated applications. This approach enables FRIBDAQ to be a modular system capable of tackling a wide range of experimental setups, from small calibrations setups to merging multiple independent data acquisitions into unified systems. Part of FRIBDAQ is SpecTcl, a C++ framework for analysis that allows experimenters to write custom analysis software that seamlessly integrates with a histogramming engine and viewer. SpecTcl enables the quick creation of histograms from the data and the ability to create 1D and 2D gates and apply them to histograms on the fly. FRIBDAQ documentation is maintained and publicly available [2].

4 Standard Configuration

4.1 Supported Hardware

FRIBDAQ supports a big collection of VME, CAMAC modules that can be configured via text files. It also provides full support for the XIA Pixie 16(32) digitizers, CAEN 17xx digitizers, CAEN FERS, GET electronics, and the RD51 Collaboration Scalable Readout System (SRS) [3].

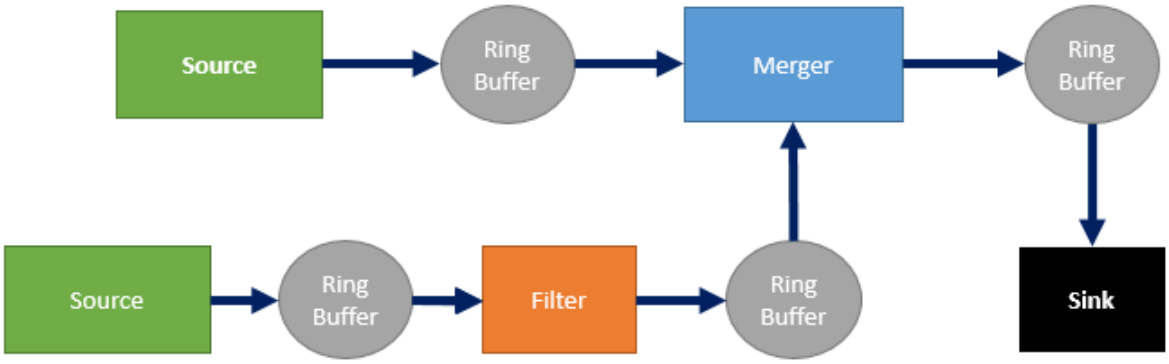
4.2 Supported Software

4.2.1 FRIBDAQ Support

A major component of FRIBDAQ is the ability for it to manage data flows. A data flow is the path each element of data follows between the point of its originating process and destination

process(es). The backbone of data flow in NSCLDAQ is the *ring buffer*, a piece of memory that is accessible to multiple processes. The idea is that ring buffers are holding spaces for data. Processes can deposit data into them and others can read data out of them.

The data flow comprises four processes: *data source*, *data sink*, *filter*, *merger*. It is useful to visualize the data flow for any given setup with a flow chart, as shown below:



The four processes can be described as:

- *Data source*: only outputs data. It does not read data from any upstream process. A good example of a data source is a Readout program. Readout programs interface with hardware and then output their data. It is most often the case that data sources output their data to a ring buffer. This process requires user-specific code to have a working readout system.
- *Data sink*: only reads data from an input source (typically a ring buffer). There are many examples of data sinks. Two good examples are processes that read and write the data stream to a file and processes that help users inspect the data.
- *Filter*: programs that have both an input and an output. Filters perform lots of tasks, two of which are transforming data between formats and inline data integrity checking. This process requires user-specific code to achieve its goal.
- *Merger*: characterized by having multiple inputs and a single output. There is only one example of such a program in FRIBDAQ, and that is the event builder. FRIBDAQ offers timestamp-based event building from multiple sources by sorting and combining events within a user-defined time interval. This process may require a user-specific code to load the correct timestamping library.

The main tool for the data flow at higher level is the ReadoutGUI. It is a graphical user interface that user's interact with to run their experiment. Its purpose is to centralize controls over the entire experiment. Example of its features are control over data recording, consolidation of output from the various processes in the pipeline, setting up the pipeline, monitoring the health of the data sources, and managing the directory of recorded data files.

The libraries and the binaries of the programs are distributed by the SDAQ Group and available throughout the DAQ network (/usr/opt/fribdaq). Examples of configurations are available on the webpage [2].

Maximum data rates (per crate) for the supported system can be found below:

	Maximum data rates (Mbyte/s/crate)
VME	~320
XIA	~109
GET	~100
SRS	~3200

For any experiment setup with expected data rates higher than the one listed above, please contact the SDAQ Group for support.

4.2.2 SpecTcl Support

SpecTcl (Spectra + Tcl) implements an extension to the Tcl/Tk scripting language as its command language. This allows users to not only type simple commands at the program, but to write scripts of commands and write graphical user interfaces interfaces with SpecTcl. SpecTcl's purpose is making it easy to create histograms, define conditions (gates), and apply those conditions to spectra.

Events packaged in accordance with the format of some data acquisition system's outer format arrive at the SpecTcl *buffer decoder* object. The buffer decoder is responsible for knowing about the other format of the data (or data envelope). It opens those envelopes and produces events. Each events passes through a logical pipeline of *event processors*. The event processors are bits of software needed to tailor SpecTcl to a specific analysis problem. Each event processor of the pipeline has access to the raw event and the parameters that have been produced by previous stages of the pipeline. The parameters that have been produced by this event processing pipeline are then passed on to the histogrammer. It uses a set of dictionaries of parameters, spectra gates and gate applications to, given the parameters of each event, determine how to increment the spectra.

Important points about SpecTcl:

- Event processing is done as background processing so from the user interface, including the command interpreter, is live all the times.
- It can accept event data from two types of sources: files and UNIX pipes (a form transfer of standard output to some other destination). The UNIX pipe data source allows SpecTcl to be attached to online data acquisition systems to do analysis of live data.
- Creation of histograms, gates, and application of gates to spectra is dynamic and can occur while analysis is in progress.

SpecTcl performs its analysis using several classes of objects. Once the event is unpacked, it is the configuration of these objects rather than any programming on the user part that determines how the analysis is performed.

The types of objects used in the analysis are:

- *Parameters*: parameters are the output of the event processors, the code that connects SpecTcl to the event data from the experiment that's being analyzed. Typically, event data consists of data from some set of digitizer hardware. Each digitizers may package the data in different way.

- One can also derive parameters by performing computations on other parameters. For example, one can take a raw digitizer channel value and apply some calibration function to it. Parameters that are the results of computations on other parameters are called *computed parameters*, or *pseudo parameters*.
- *Spectra*: they are histograms and the "output" of SpecTcl. SpecTcl is normally run in conjunction with a visualization program. Originally, the program was Xamine (now legacy). More recently a new visualization program called QtPy (CutiePie) provides visualization based on the PyQt toolkit and takes advantage of the Python3 libraries.
- SpecTcl can locate spectra in a shared memory region. This permits high speed access to bulk spectrum data by visualization programs. Said spectra are said to be *bound* to the displayer.
- *Gates*: they are conditions. Gates can be primitive regions of interest that are drawn on spectra. They can also be Boolean combinations of other gates. Those are called *compound gates* while the former are called *primitive gates*. What makes a gate useful is that it can be *applied* to a spectrum. When a spectrum has a gate applied to it, the spectrum is only incremented for events that make that gate *true*.
- *Filters*: usually analysis proceeds in several stages. Often, each stage of analysis refines the set of events that are interesting to the experiment. The unpacking of each event from raw format can be time consuming as well. A filter is a way to save a new event file that consists of a subset of the parameters of a subset of events.

The libraries and the binaries of the programs are distributed by the SDAQ Group and available throughout the DAQ and the Office network (/usr/opt/spectcl). Examples of implementation of event processors and configurations are available on the webpage [2].

5 Support Level

On-site support is normally available from 8 a.m. to 5 p.m. on working days. On-call support for critical technical assistance during the experiment outside of the normal working hours is available. Any request outside the working days has to be addressed by contacting the operator in charge (OIC), who will then contact the SDAQ Group. For non-critical support, the SDAQ Group is reachable via daqhelp@frib.msu.edu. The SDAQ Group follows the documented "FRIB Roles and Responsibilities of Scientific Users" [4].

5.1 User Support in Preparation for an Experiment

The SDAQ Group answers technical questions for users during the preparation of experiment proposals. During the first ERR, users can explicitly request DAQ support for PAC approved experiments.

The SDAQ Group provides the following support:

- Provide stable and reliable software packages and documentation necessary to run correctly the FRIBDAQ and SpecTcl.
- Assist users during the hardware setup and debugging stages of the DAQ setup by offering best practices.
- If needed, fix software bugs with the highest priority to avoid beam time losses during the experiment.

5.2 User Support during an Experiment

The SDAQ Group provides the following support during the experiment:

- Assist users in inspecting and understanding problems concerning the readout that cause faulty data or no data at all.
- Assist users to address IT related problems (i.e. networking, computing, storage…).

6 Additional Support

The SDAQ Group is open to adaptation, adoption, and creation of new tools, if they align with the needs of the FRIB Experimental Scientific Program. The process follows these steps:

- FRIB Users communicate the request for new hardware/software support to the SDAQ Group for a technical evaluation of feasibility, estimate of effort, and potential overall impact.
- The request will be sent to the DAQ Committee for prioritization with input from the FRIB Manager of User Relations (if the request is specific to an experiment proposal).
- Development and deployment.

7 References

[1] "FRIB Experiment Readiness Review Program" FRIB-S30206-PG-000033
[2] http://docs.nscl.msu.edu/daq/newsite/index.php
[3] "Development of the scalable readout system for micro-pattern gas detectors and other applications" JINST 8 (2013) C03015
[4] "Roles and Responsibilities of the Scientific Users" FRIB-S30206-PG-000028