



Mu2e DAQ and slow control systems

Antonio Gioiosa

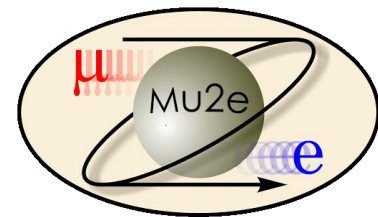
Università di Pisa, INFN Pisa

Level 3 Manager of Trigger &

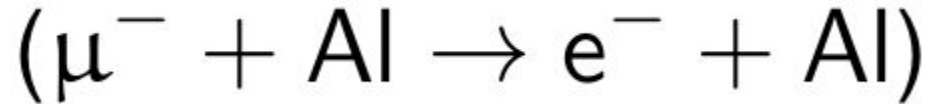
Data Acquisition System Design & Test

DIS-XXVIII International Workshop 2021

April 14, 2021

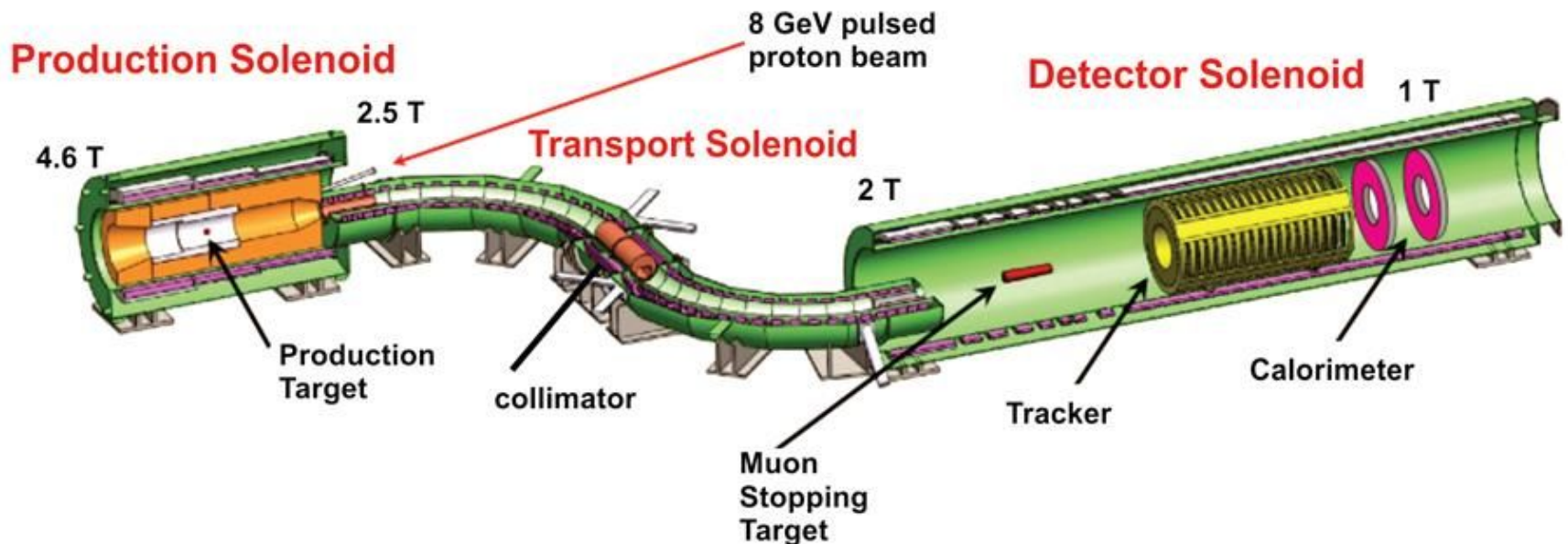


The Mu2e Experiment at Fermilab



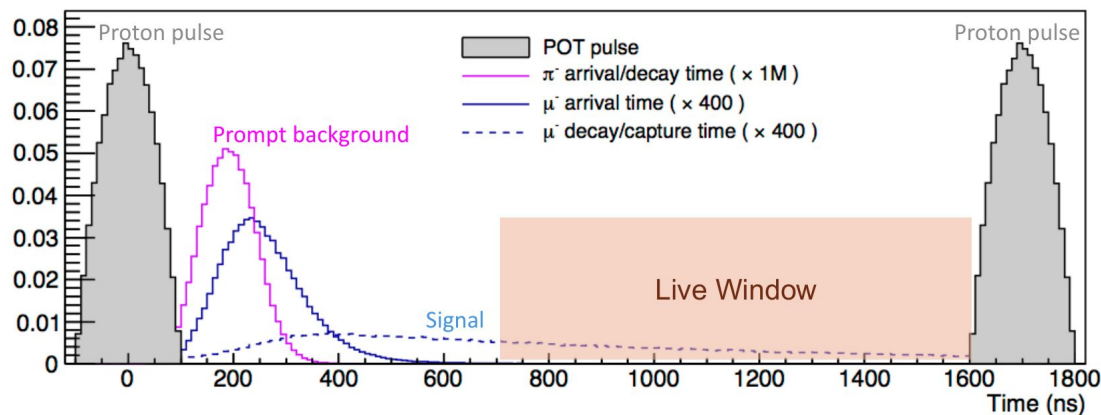
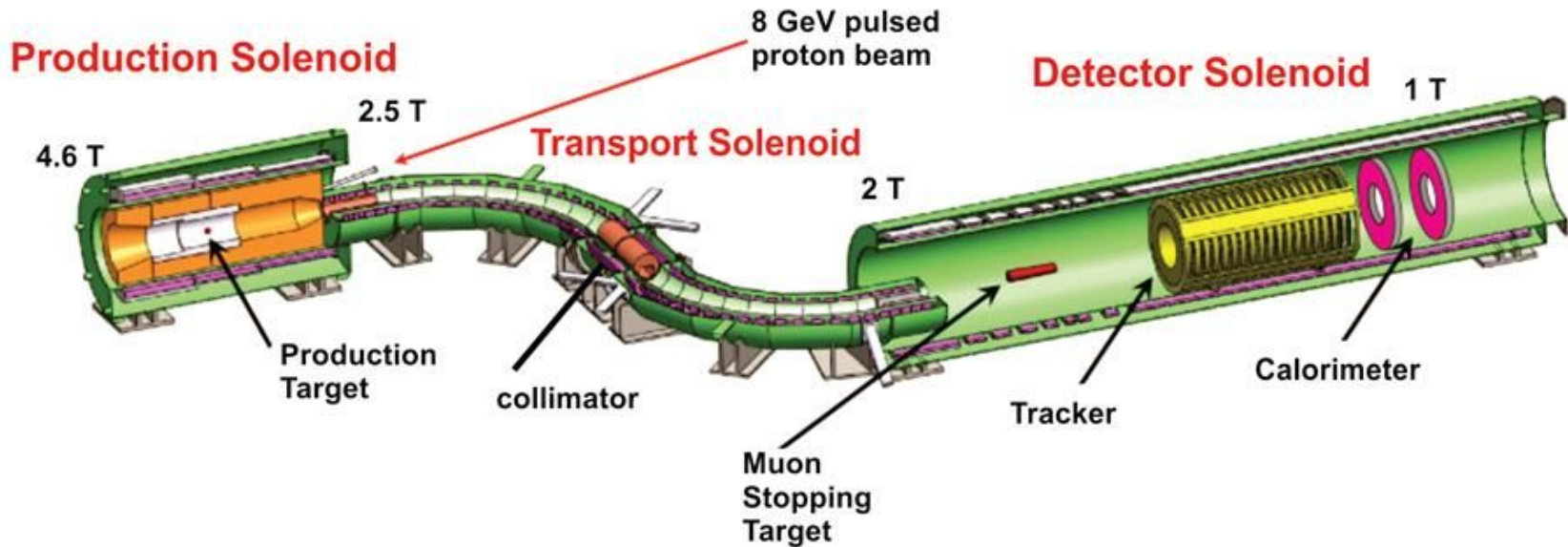
With the expected experimental sensitivity, Mu2e will improve the SINDRUM II limit ($7.0 \cdot 10^{-13}$) of four orders of magnitude

(Mu2e intends to reach a single event sensitivity of $3.0 \cdot 10^{-17}$, assuming we will run for three years, with $3.6 \cdot 10^{20}$ protons, with a run time of $6.0 \cdot 10^7$ s, requiring a background level below 1 event)



The Mu2e Experiment at Fermilab

The signal we are looking for is a delayed monoenergetic electron with an energy of just under 105 MeV (muon mass)

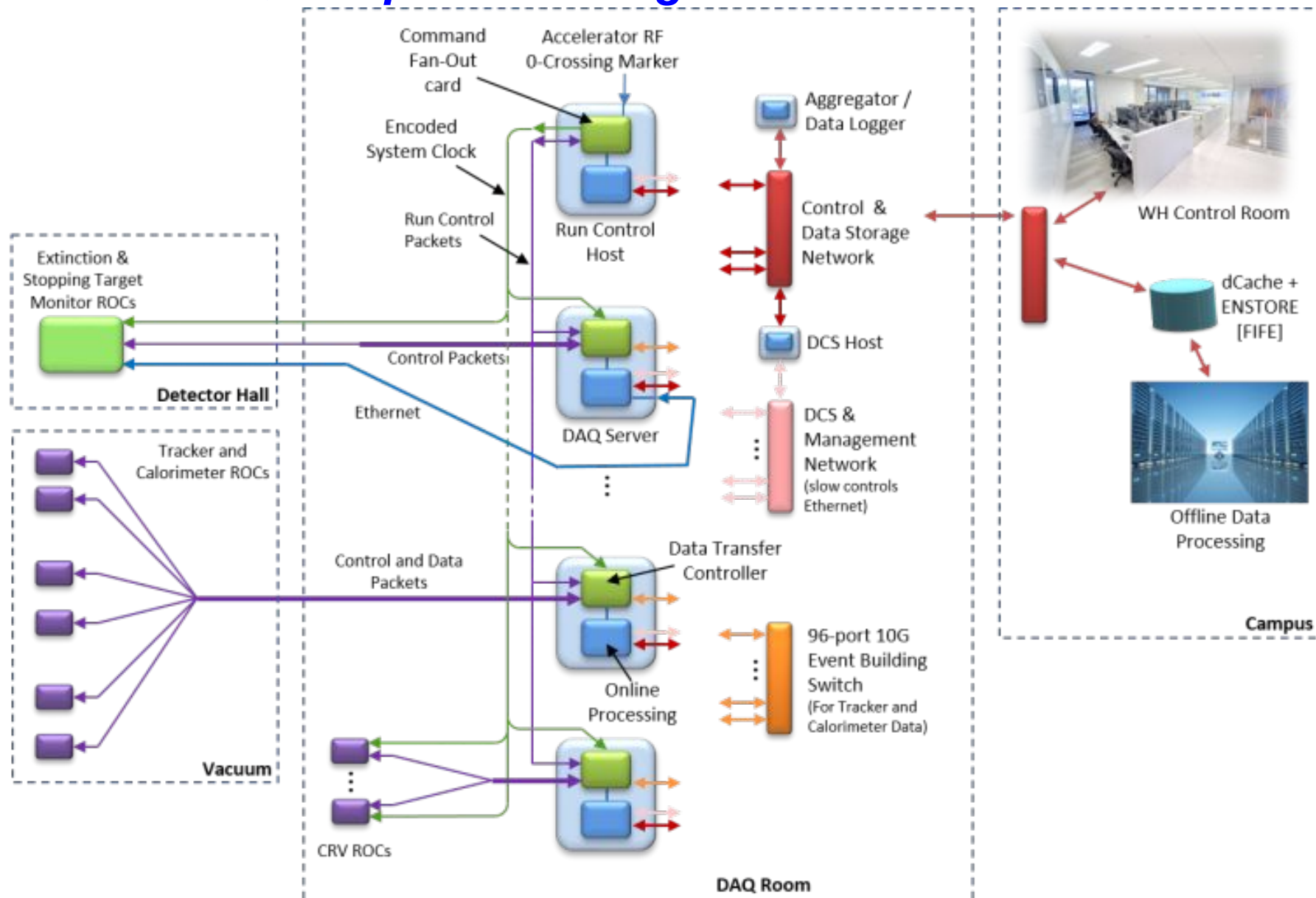


Mu2e TDAQ and Slow Control integration

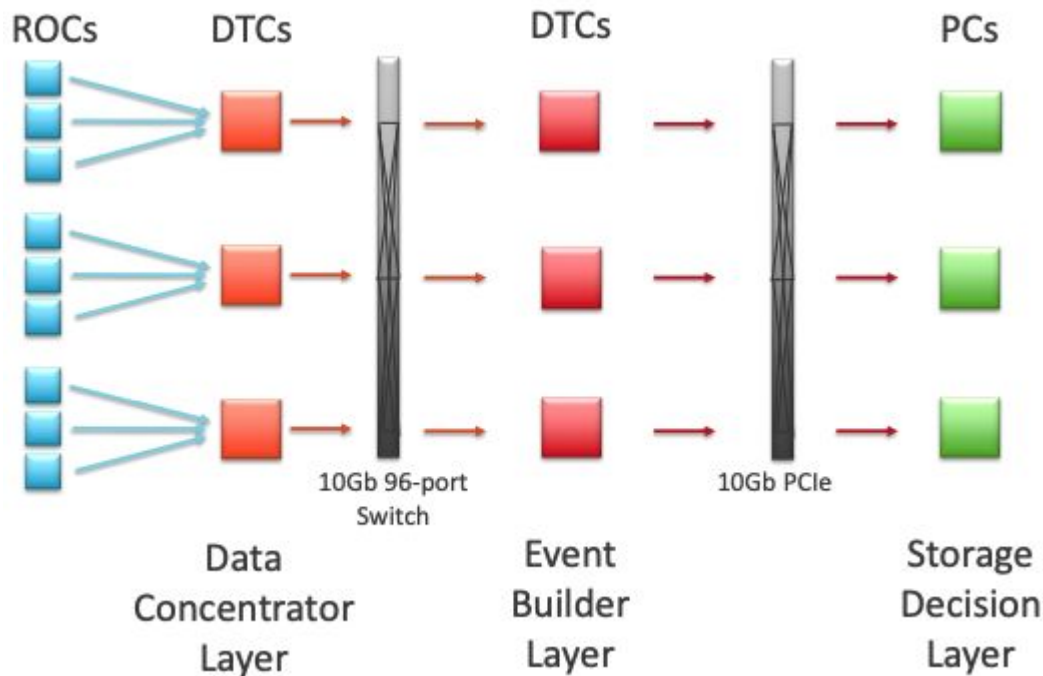
Summary:

- Mu2e TDAQ components Diagram
- Mu2e TDAQ Readout scheme
- Online DAQ (***otsdaq***) overview
- Slow control and its integration in ***otsdaq***
 - **Monitoring** and Slow Controls GUI
 - Slow Controls **Integration** with ***otsdaq*** State Machine and **Alarm handling**
- Conclusions

Mu2e TDAQ components Diagram



TDAQ Readout scheme



- 396 ROCs 69 DTCs (Kintex-7) for data readout and event building
- Large front end buffers to average over long off-spill time
- 800 threads on 40 nodes for HLT → ~5 ms per event
- ~40 GB/s data read out to storage decision layer, ~280 MB/s written to disk

Mu2e Online DAQ solution: *otsdaq*

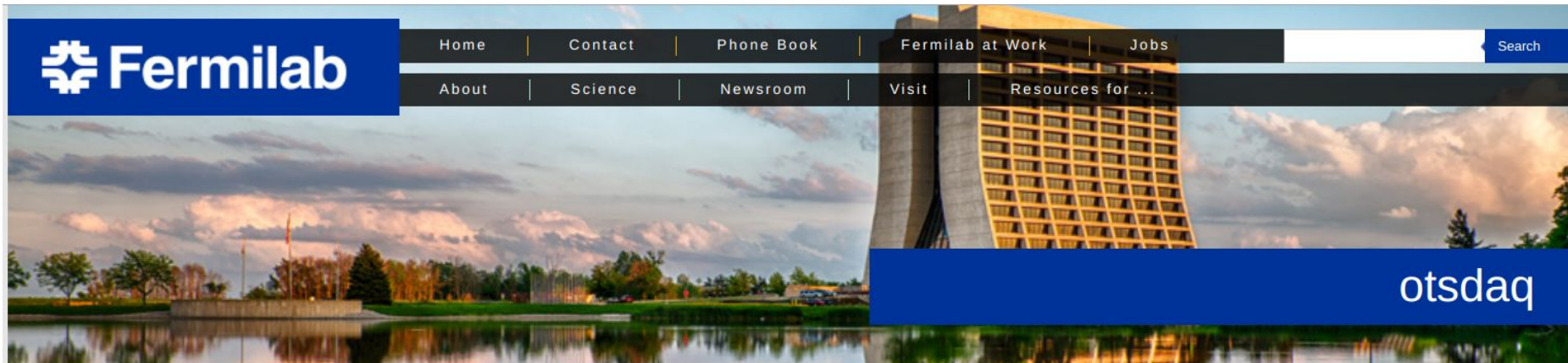


otsdaq overview

Acronym for “off-the-shelf data acquisition.”

- ***otsdaq*** is a Ready-to-Use data-acquisition (DAQ) solution aimed at test-beam, detector development, and other rapid-deployment scenarios
- it uses the ***artdaq*** DAQ framework under-the-hood, providing flexibility and scalability to meet evolving DAQ needs
- ***otsdaq*** provides a library of supported front-end boards and firmware modules which implement a custom UDP protocol
- Developments are in two directions: **server** side and **web** side.
- An integrated Run Control GUI and readout software are provided, preconfigured to communicate with ***otsdaq*** firmware

More info at **otsdaq** web page <https://otsdaq.fnal.gov/>



otsdaq

Project Homepage

Source Code Documentation

User Manual

Tutorials (User/Expert Training)

"First Demo" tutorial



otsdaq is a Ready-to-Use data-acquisition (DAQ) solution aimed at test-beam, detector development, and other rapid-deployment scenarios. *otsdaq* uses the *artdaq* DAQ framework under-the-hood, providing flexibility and scalability to meet evolving DAQ needs. *otsdaq* provides a library of supported front-end boards and firmware modules which implement a custom UDP protocol. Additionally, an integrated Run Control GUI and readout software are provided, preconfigured to communicate with *otsdaq* firmware.

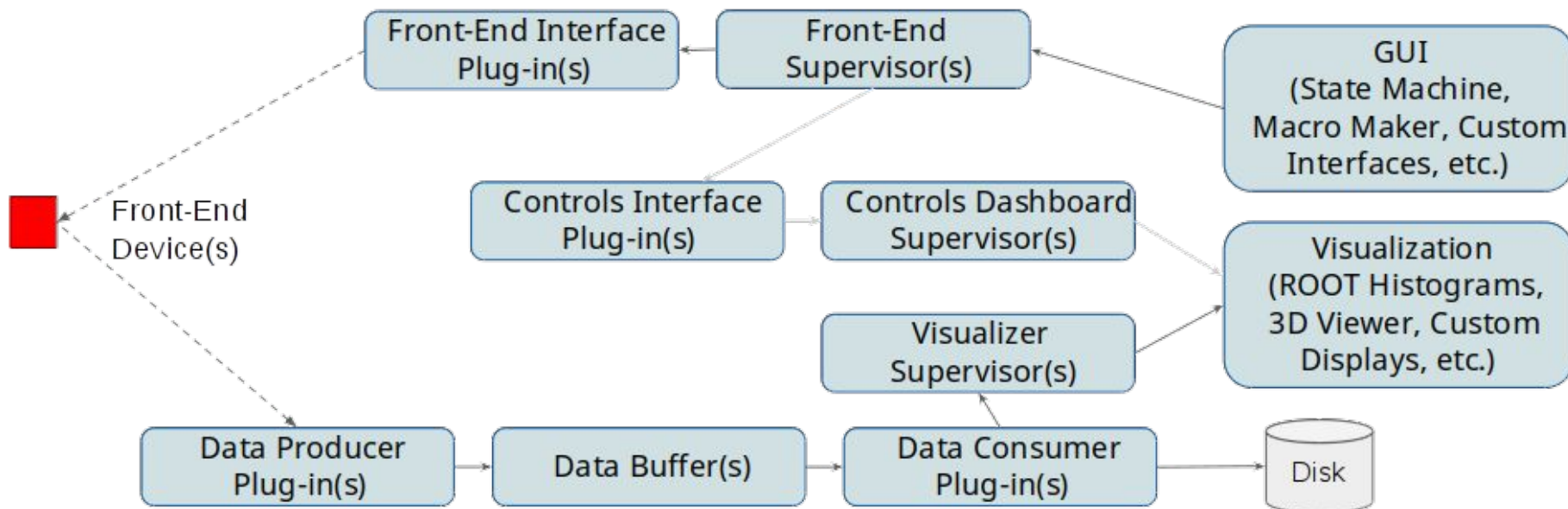
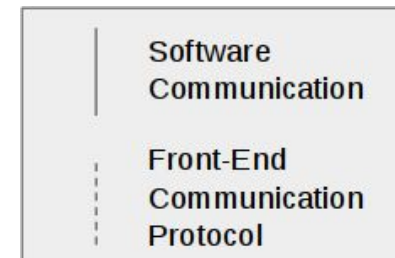
Last modified: 04/29/20 | email Fermilab

otsdaq overview



Data Flow Block Diagram

Server side is C++. User code is added through plugins (C++ classes inheriting from the appropriate class)

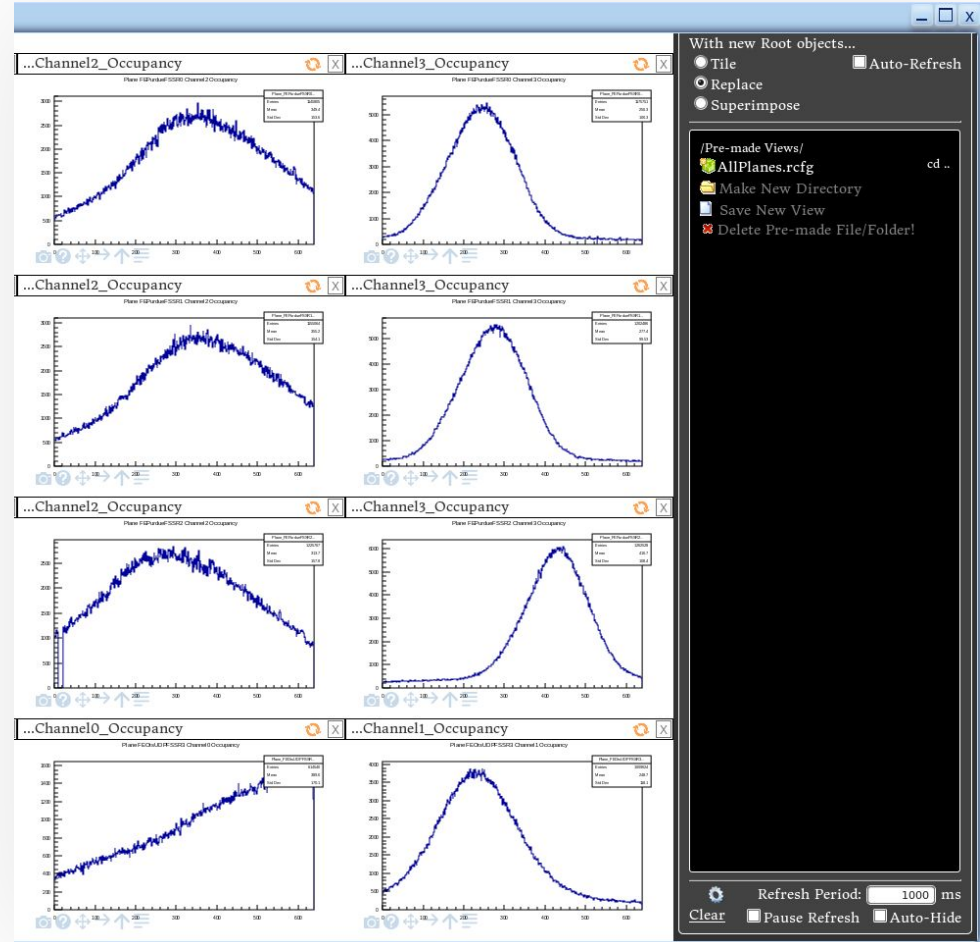


Web side is HTML and JavaScript. User code is added in the form of web-apps through .html files (including the appropriate .js and .css files)

otsdaq overview

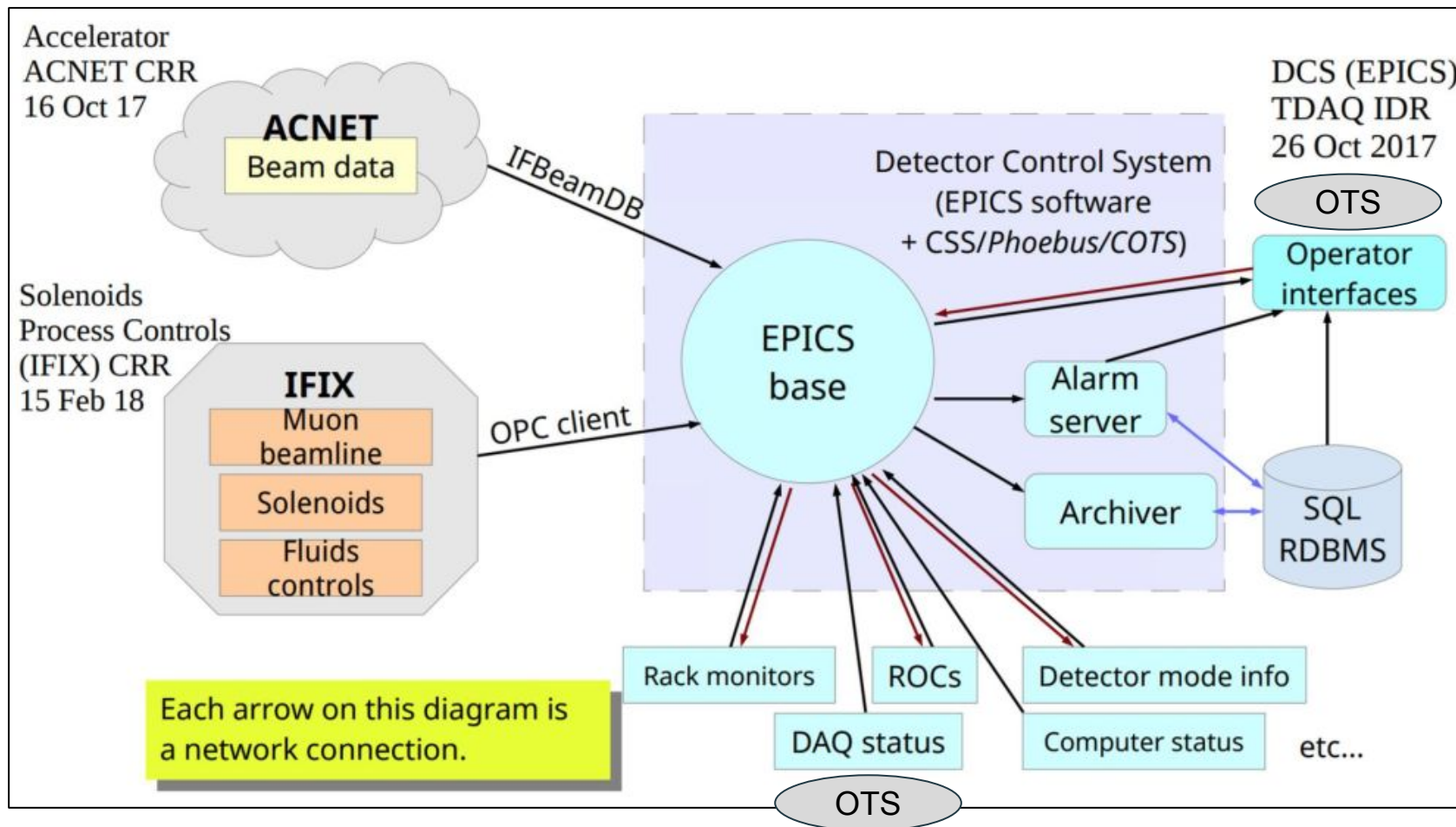
Data processing: Data Quality monitor GUI example

- Mu2e's event window data will be processed through artdaq modules
- Data processor and Data Quality Monitor **DQM** plugins are provided by otsdaq core
- **DQM** generates data products that are sent to an **artdaq Dispatcher**, which aggregates **DQM metrics** and presents them to a visualizer application



Slow Controls connection and **EPICS** plugin development in *otsdaq*

Experimental Physics and Industrial Control System



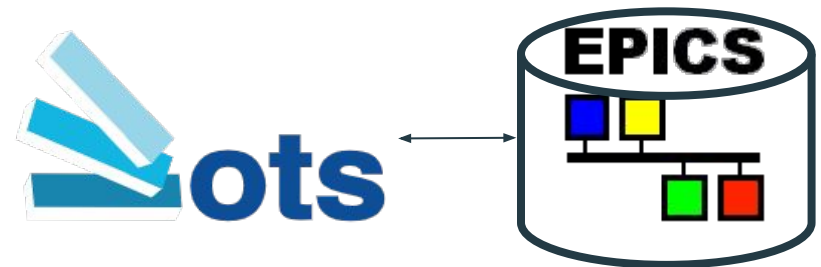
Slow Controls connection and **EPICS** plugin development in *otsdaq*

Channel subscription to **EPICS** (uses Input Output Controller **IOC**)

- Value
- Alarm (Status, Severity)
- Settings
 - *Process Variable Unit, Lower and Upper Warning Limits, Lower and Upper Alarm Limit, Lower and Upper Control Limits, Lower and Upper Display Limits*



- Channel history and alarms retrieving from EPICS Archiver Databases



- *dcx_archiver*
- *dcx_alarm*
- *dcx_log*

Slow Controls Monitoring in otsdaq

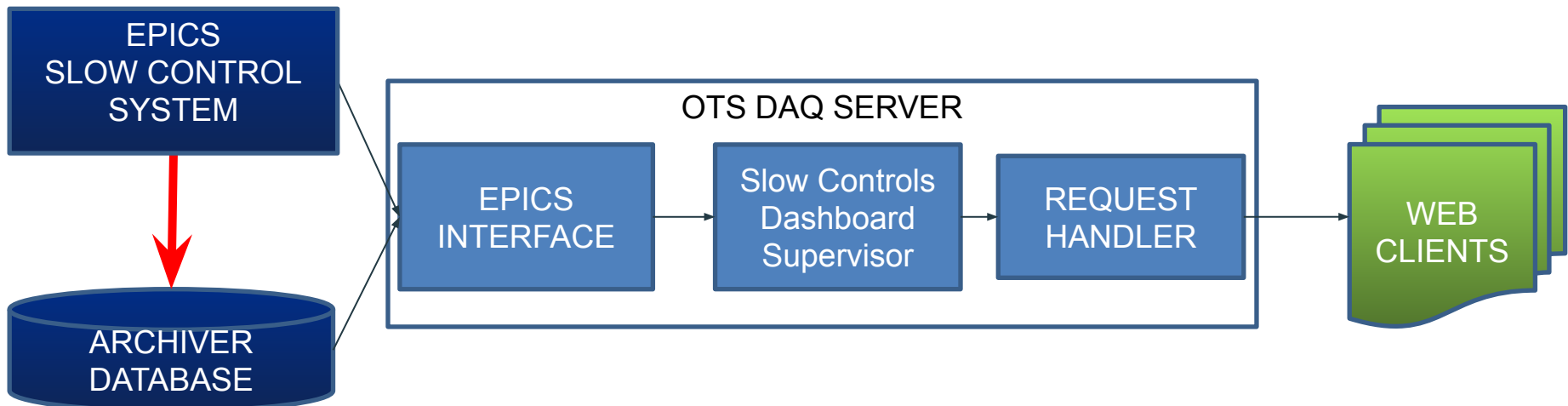
Slow Controls Software purpose

- Allow the user to monitor or interact with their own DAQ hardware. Able to see things such as:

Alarms, Warnings, Readouts, Timestamps, Status

- Interact through a web interface that is:

Lightweight, User-Friendly, Plug n' Play, Customizable

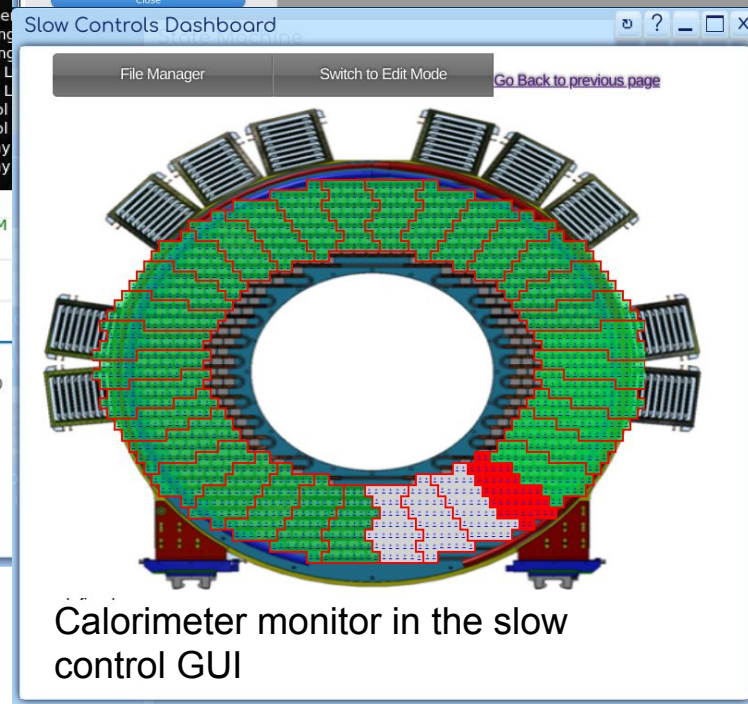
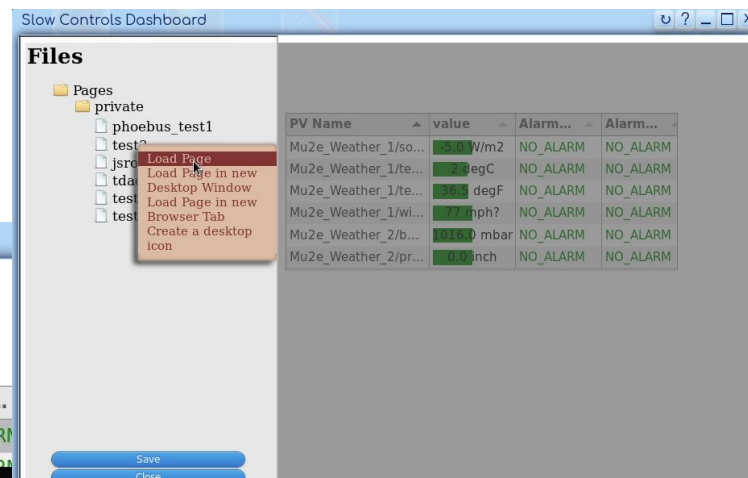
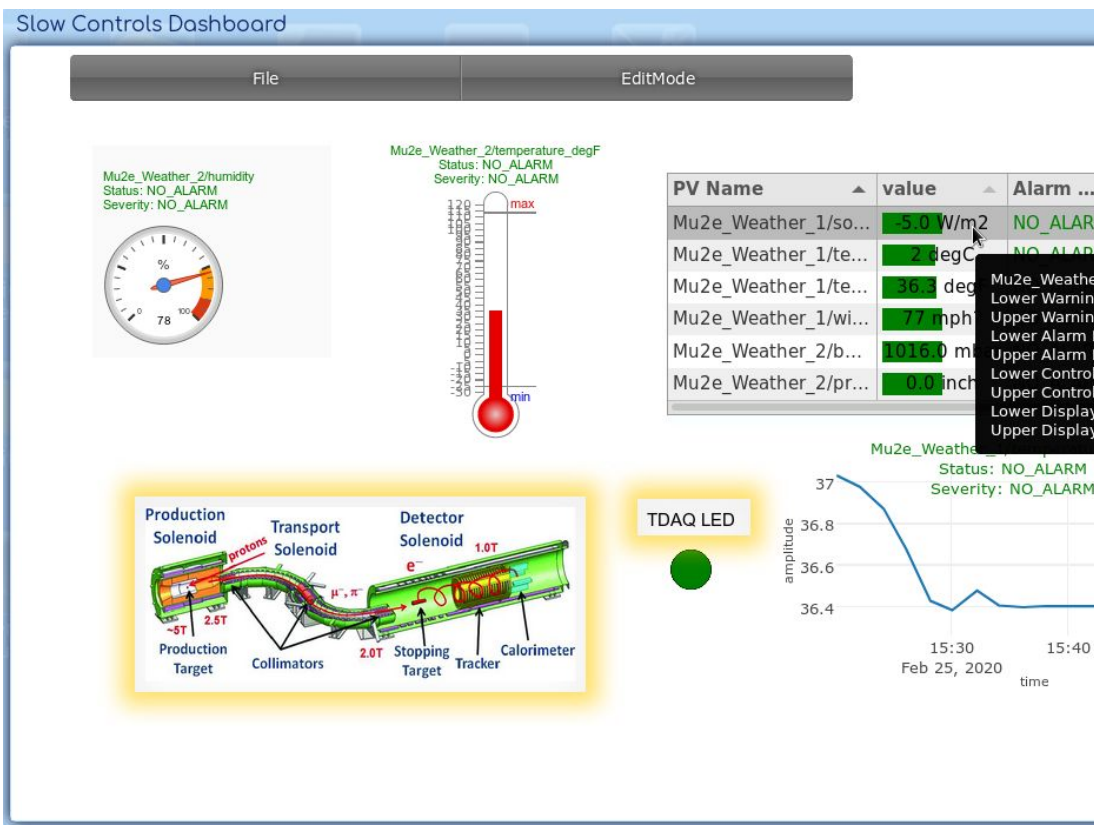


Slow Controls Monitoring GUI in otsdaq

Example of page loading

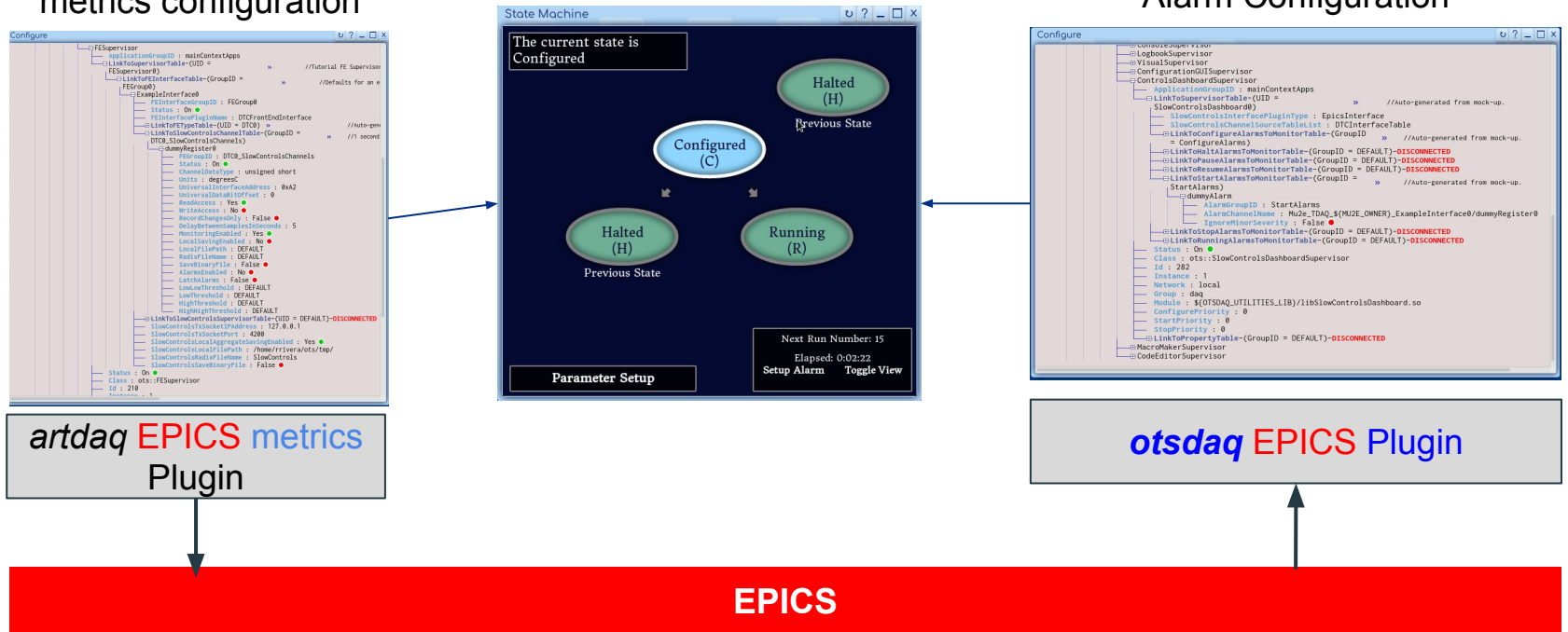
Examples

Example of loaded page



Integration with State Machine

- **State Machine** Configuration and data subscription to **EPICS**
 - Alarm propagation (from **EPICS**) and *otsdaq* State Machine handling
- DAQ HW, artdaq and DQM metrics configuration
- State Machine
- Alarm Configuration



Conclusions



- Mu2e Experiment is under construction at Fermilab and will be ready for data taking in two/three years
- Mu2e TDAQ and slow control are in large part developed according to the requirements (200K events/s for data taking) and hardware tests are going on
- Slow control integration in the online DAQ system, *otsdaq*, provides an advanced slow controls monitoring, an interface to send *otsdaq* front-end DAQ hardware, data processing, and DQM slow controls information to **EPICS**, and a real configuration and Integration with the *otsdaq* State Machine

This work was supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement no 734303, 822185, 858199, 101003460

Backup Slides

Slow Controls **WEB Monitoring GUI in otsdaq**

developed in JavaScript and HTML (client side) and C++ (server side)

Basic Widget Mechanism

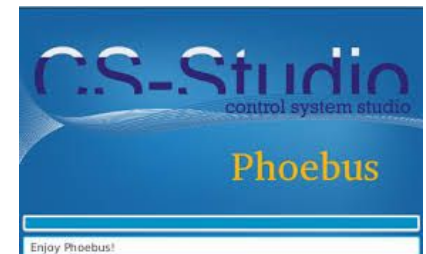
- All widgets have six required methods:
init(), getParameters(), setParameters(), setupPVs(), newWidget(), and newValue()

Widget properties

- Dynamic sizing
- Proper handling of setups
- Value error, warning and alarm handling
- Disconnection handling

Load and save dashboard page in XML

Cs-Studio Phoebus (EPICS GUI) compatible format



Mu2e is an experiment under construction at [Fermilab](#) to measure the charged-lepton flavour violating neutrinoless conversion of a negative muon into an electron in the field of an aluminum nucleus

Mu2e TDAQ Data Flow and High Level Trigger Software

Run Control and Data Flow summary

1. Experiment defined Run Plan is coordinated by CFO. The System Clock (40MHz) and Event Window markers originate at the Command Fan-Out Card (CFO) and are distributed to ROCs
2. CFO distributes System Clock and Event Windows to DTCs with fixed latency
3. DTCs distribute System Clock and Event Windows to ROCs with fixed latency
4. ROCs respond to Data Requests
5. A slice of the detector arrives at each DTC (6 ROCs for 1 DTC)
6. DTCs forward data slice through Event Building Switch to round-robin DTC destination
7. DTCs receive full events from multiple DTC sources, pre-process, and pass through PCIe to online processing
8. Trigger decision is made in online processing
9. Trigger accept causes readout of corresponding CRV data
10. Event data from all detectors are aggregated at Data Logger
11. Experiment data is transferred from Data Logger to persistent storage

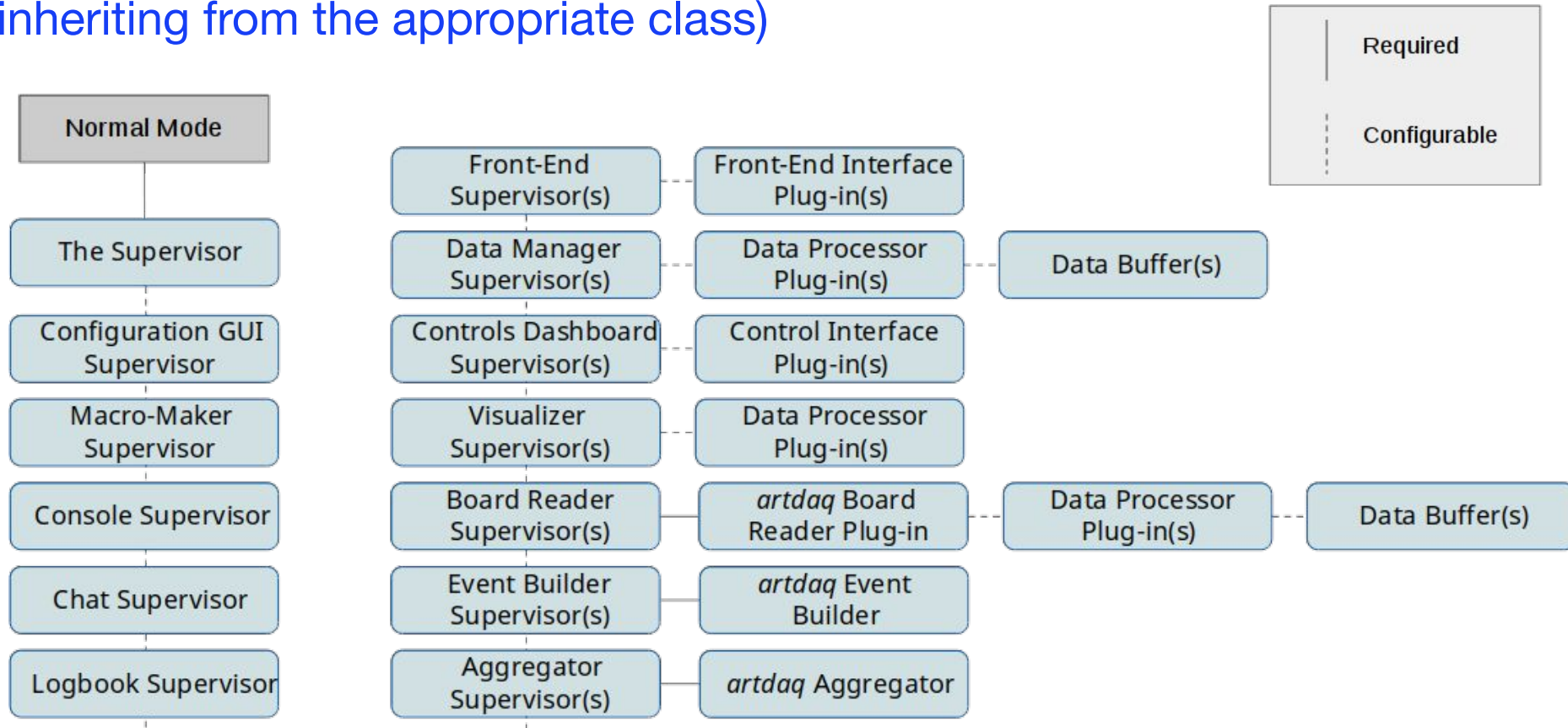
High Level Trigger Software



otsdaq overview



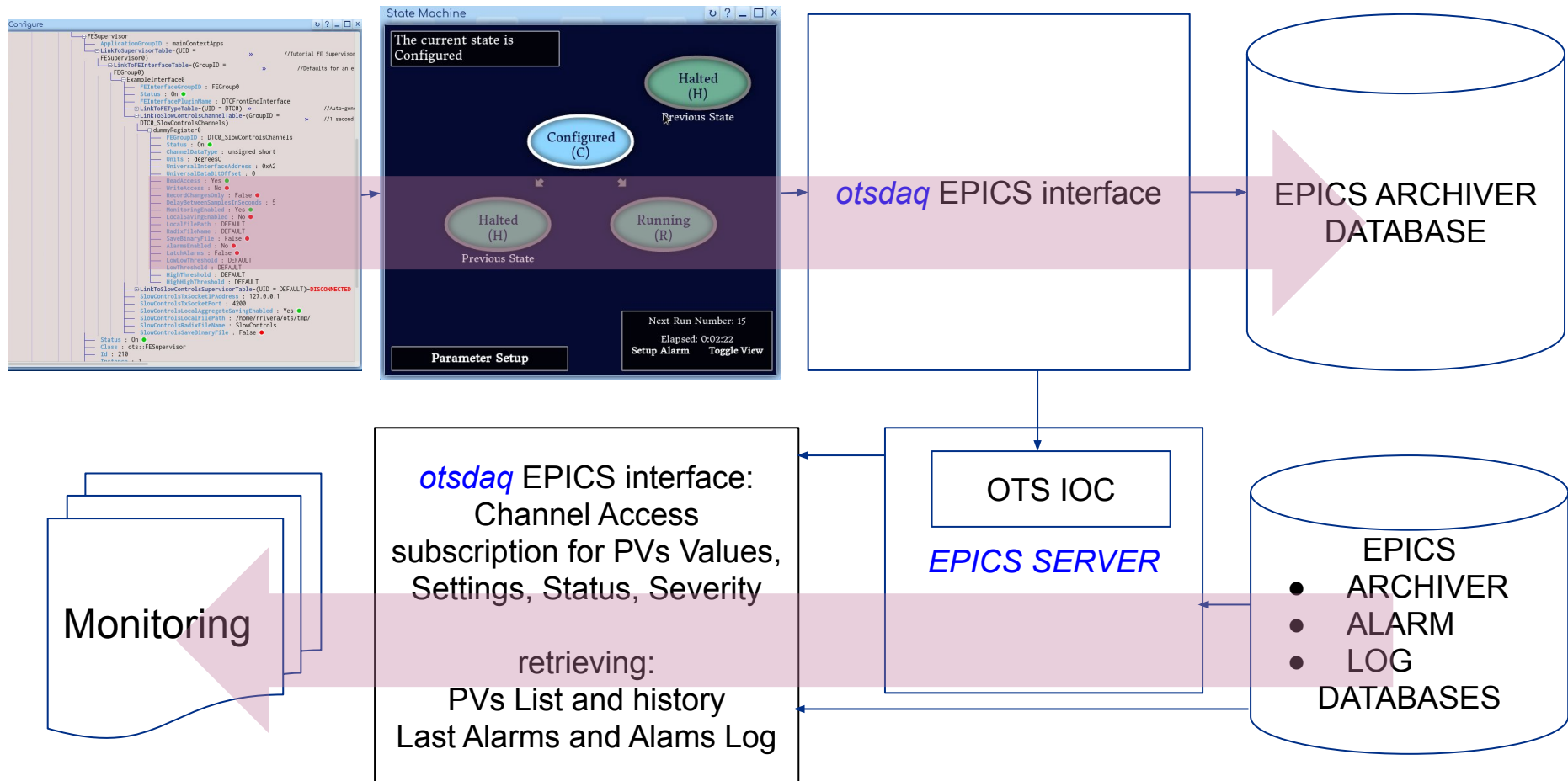
Server side is C++. User code is added through plugins (C++ classes inheriting from the appropriate class)



Web side is HTML and JavaScript. User code is added in the form of web-apps through .html files (including the appropriate .js and .css files)

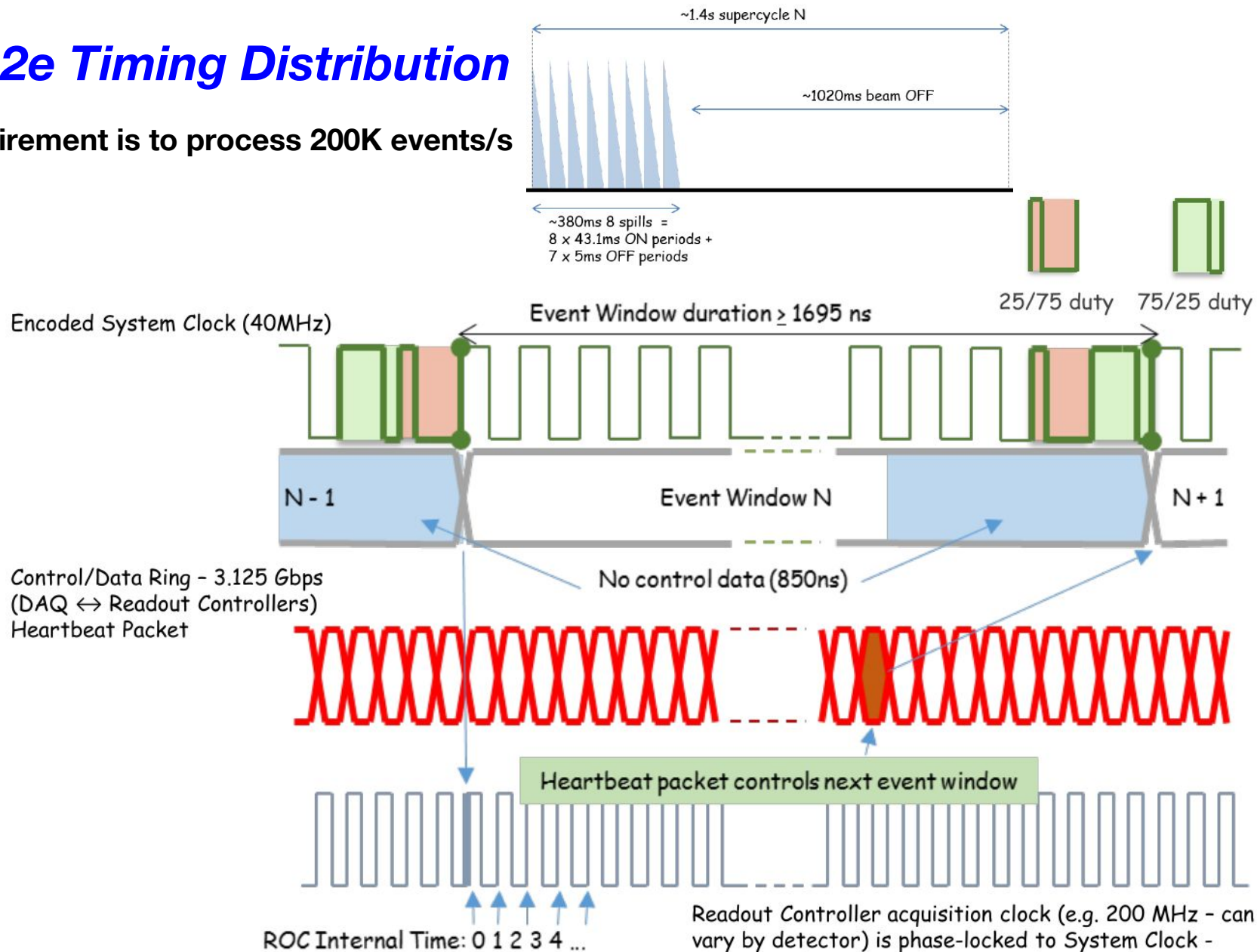
Integration with State Machine

- *otsdaq FE (DTC/ROC/CFO) / artdaq metric new channel or new slow control setting* → configuring State Machine → EPICS DBs and IOC configuration
- *otsdaq Interface* → *otsdaq CA subscription and DBs select* → Monitoring



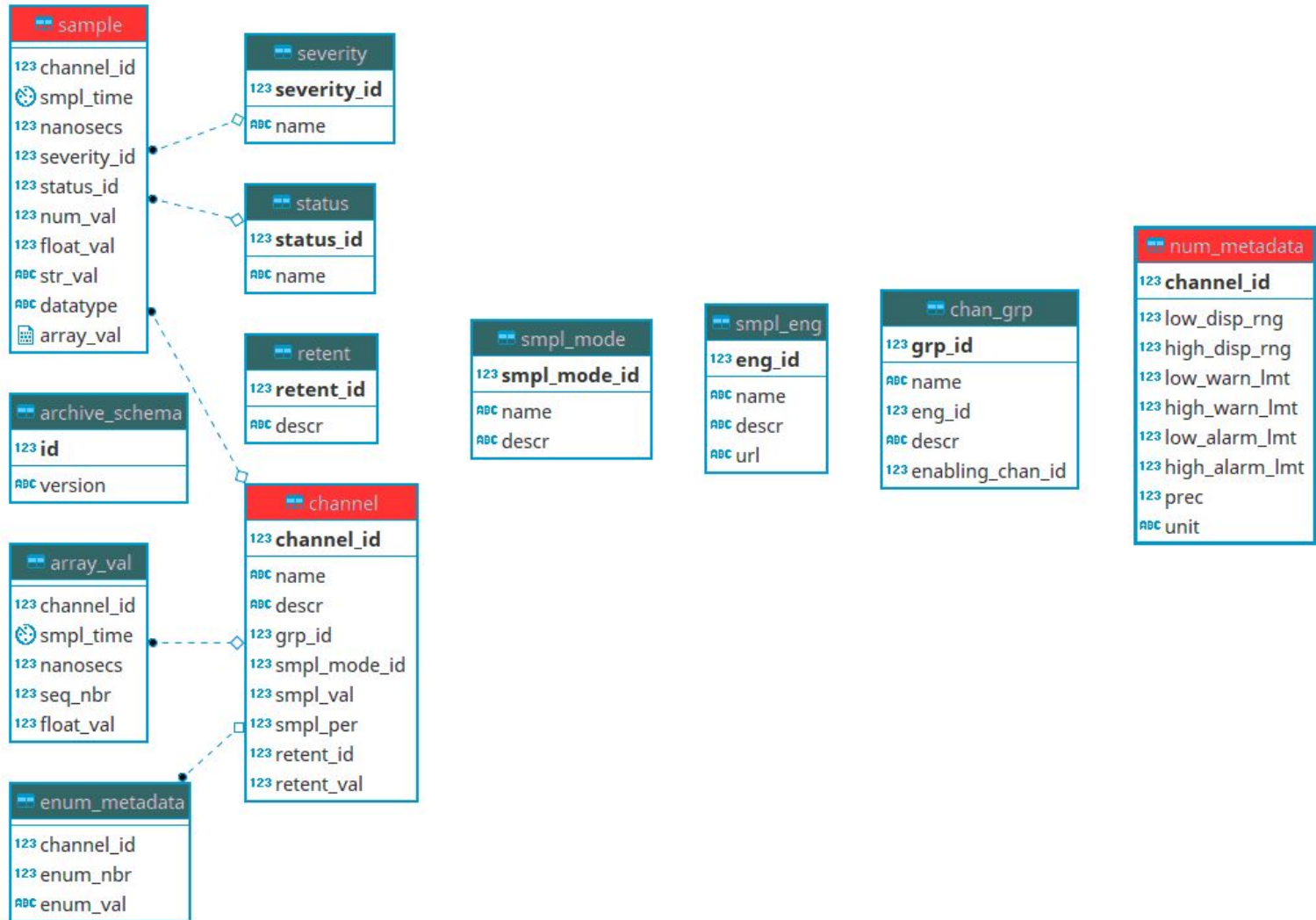
Mu2e Timing Distribution

Requirement is to process 200K events/s



EPICS Database

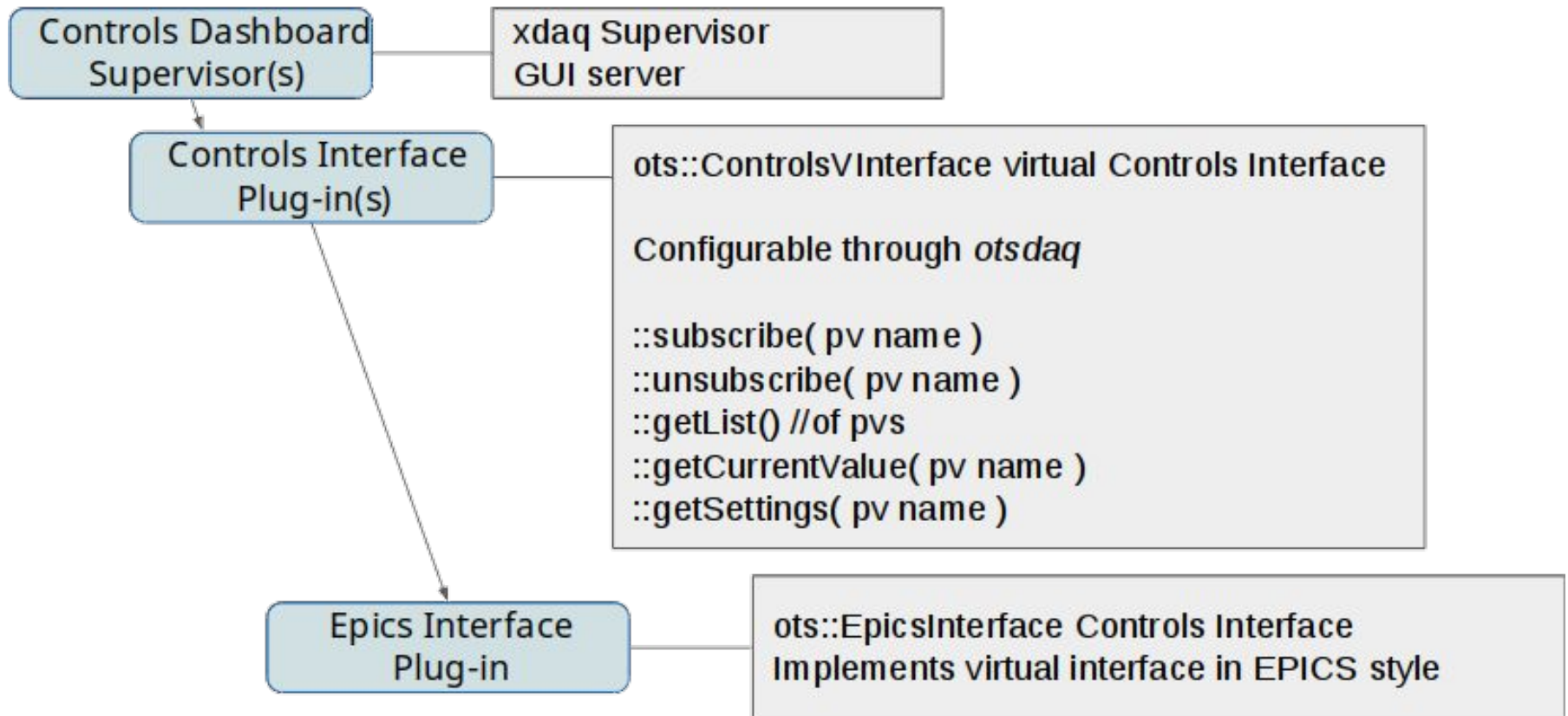
- Postgres DBMS



Slow Controls Monitoring in otsdaq



Slow Controls C++ Hierarchy

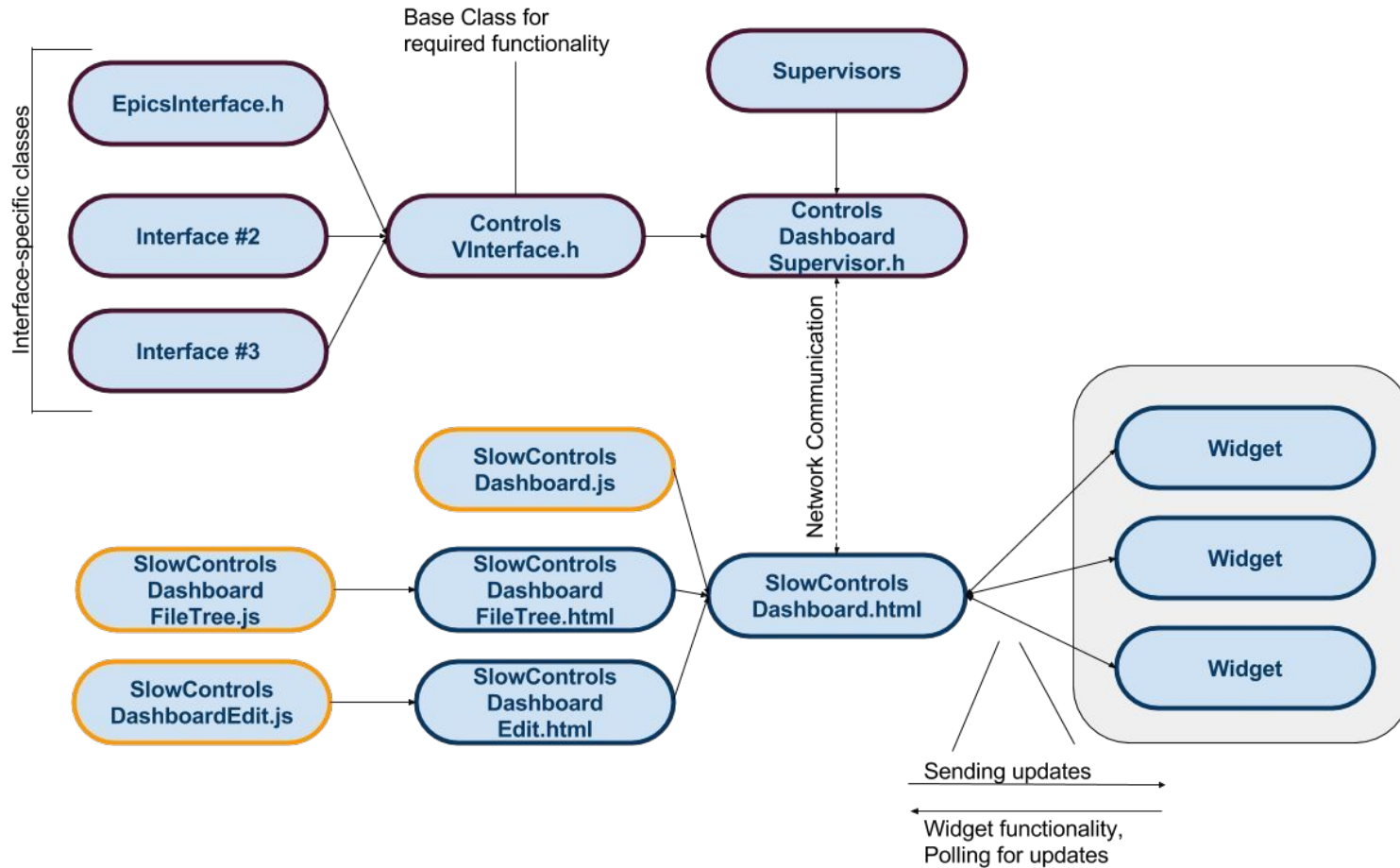


Slow Controls Monitoring in otsdaq



— C++
— HTML
— Javascript

Slow Controls GUI Hierarchy

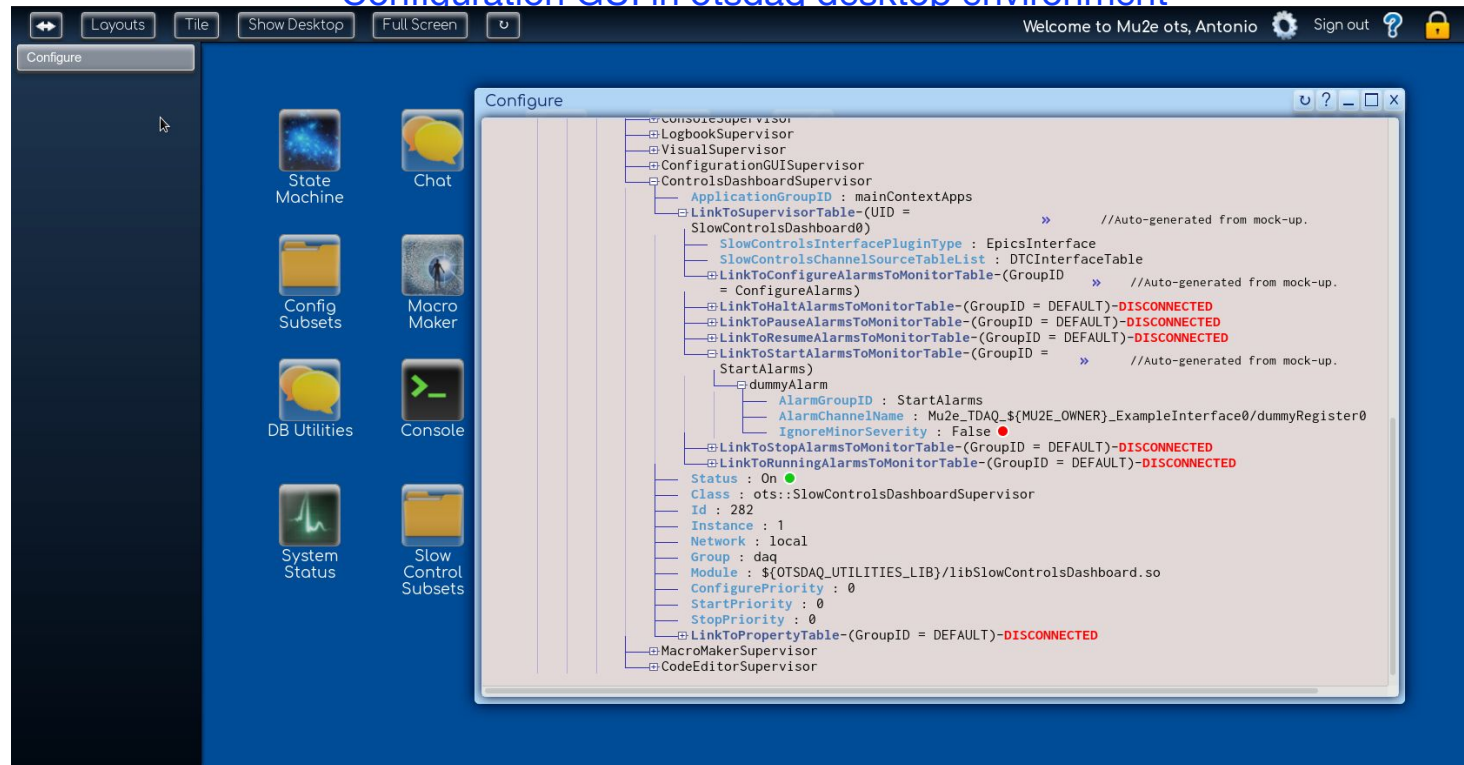


Slow Controls Monitoring in otsdaq

Configuring by specific tables in otsdaq

DesktopIconTable, XDAQApplicationPropertyTable, XDAQApplicationTable, XDAQContextTable

Configuration GUI in otsdaq desktop environment



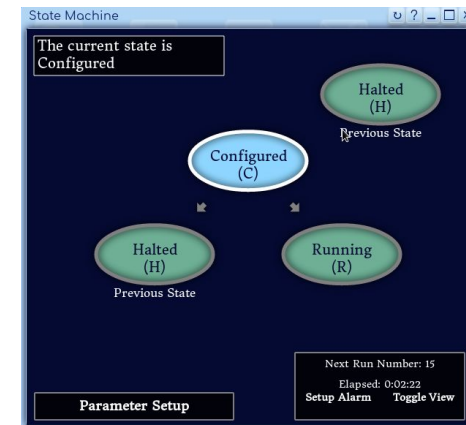
*Integration of **otsdaq** front-end DAQ hardware and **artdaq** metrics with **EPICS***

Actions designed and developed in *otsdaq*

1. *otsdaq* DCS channels Front End and tables configuration
2. *otsdaq* State Machine configuration implementation
3. add/update channels info for **IOC** and **Archiver** DB
4. software **IOC** restarting
5. **EPICS Archiver** restarting
6. new *otsdaq* epics_plugin channels subscriptions to EPICS
7. Sending configured channels values to **EPICS**:
*otsdaq DCS channels new values → artdaq Metric Manager
→ software **IOC** → **EPICS** → otsdaq DCS GUI*

Integration with State Machine

- **Alarm** propagation (from **EPICS**) and **otsdaq** state machine **handling** is available: needs just to identify which **PV alarms**, *status* and *severity* will be propagated
- *Tables and parameters designed for configuration*
 - SupervisorTable parameters:
 - *Slow Controls Interface Plugin Type*
 - *Slow Controls Channel Source Table List (HW list i.e. DTC Interface, CFO Interface)*
 - Alarms To Monitor Tables for transition to states:
 - *Configure*
 - *Halt*
 - *Pause*
 - *Resume*
 - *Start*



```
Configure
- ControlSupervisor
  - LogbookSupervisor
  - VisualSupervisor
  - ConfigurationSupervisor
  - ControlDashboardSupervisor
  - ApplicationGroup: mainContextApps
    - SlowControlDashboard
      - SlowControlInterfacePluginType: EpicsInterface
      - SlowControlChannelSourceTableList: DTCInterfaceTable
      - LinkToConfigureAlarmsToMonitorTable-GroupID = ... //Auto-generated from mock-up.
      - LinkToHaltAlarmsToMonitorTable-GroupID = DEFAULT-DISCONNECTED
      - LinkToPauseAlarmsToMonitorTable-GroupID = DEFAULT-DISCONNECTED
      - LinkToResumeAlarmsToMonitorTable-GroupID = DEFAULT-DISCONNECTED
      - LinkToStartAlarmsToMonitorTable-GroupID = ... //Auto-generated from mock-up.
    - StartAlarms
      - dummyAlarm
      - AlarmGroup: StartAlarms
      - AlarmChannelName: Mu2e_TDAQ_SMGE_OWNER.ExampleInterface/dummyRegister
      - IgnoreOnSeverity: False
      - LinkToTopAlarmsToMonitorTable-GroupID = DEFAULT-DISCONNECTED
      - LinkToBottomAlarmsToMonitorTable-GroupID = DEFAULT-DISCONNECTED
    - Status: On
    - Id: 282
    - Instance: 1
    - Network: local
    - Group: daq
    - Module: $OTSDAQ_UTILITIES_LIB/libSlowControlDashboard.so
    - ConfigPriority: 0
    - StartPriority: 0
    - StopPriority: 0
    - LinkToPropertyTable-GroupID = DEFAULT-DISCONNECTED
  - MacroMakerSupervisor
  - CodeEditorSupervisor
```

Integration with State Machine

- Alarm propagation (from EPICS) and *otsdaq* state machine handling: Example on “Start” transition

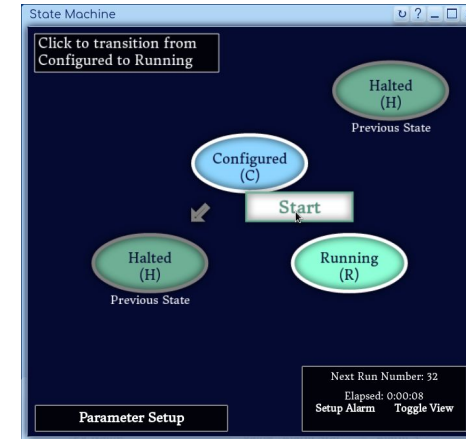
State Machine

Close Errors
Note: Newest messages are at the top.
(Press [ESC] to close and [SHIFT + ESC] to re-open)

```
:GatewaySupervisor:otsdaq/otsdaq/GatewaySupervisor/GatewaySupervisor.cc [1550]  
Received error from Supervisor instance =  
'ControlsDashboardSupervisor' [LID=282] in Context  
'mainContext' [URL=http://mu2edaq12.fnal.gov:3075].  
  
Error Message =  
:SlowControlsDashboardSupervisor:ControlsDashboardSuperv  
:otsdaq/otsdaq/CoreSupervisors/CoreSupervisorBase.cc [750]  
Error was caught while Starting:  
:EpicsInterface_slowcontrols.ccotsdaq_epics/otsdaq-epics/ControlsInterfacePlugins/EpicsInterface_slowcontrols.cc  
[1333]  
During 'start'... Alarms monitoring (count=1):  
dummyAlarm  
  
Found alarm for channel  
'Mu2e_TDAQ_shift_ExampleInterface0/dummyRegister0' =  
{time=1582678095, value=2020, status=HIHI, severity=MAJOR}!  
  
Total alarms found = 1
```

Slow Controls Dashboard

PV Name	value	Alarm Stat...	Alarm S...
Mu2e_TDAQ_shift_ExampleInte...	2020	HIHI	MAJOR



Configure

- LogbookSupervisor
- VisualSupervisor
- ConfigurationUISupervisor
- ControlsDashboardSupervisor
 - ApplicationGroupID: mainContextApps
 - SlowControlsDashboard
 - SlowControlsInterfacePluginType: EpicsInterface
 - SlowControlsChannelSourceTableList: DTInterfaceTable
 - LinkToConfigurableAlarmsMonitorTable-GroupID = //Auto-generated from mock-up.
 - LinkToAlarmsMonitorTable-GroupID = DEFAULT-DISCONNECTED
 - LinkToNewAlarmsMonitorTable-GroupID = DEFAULT-DISCONNECTED
 - LinkToStartAlarmsMonitorTable-GroupID = //Auto-generated from mock-up.
 - StartAlarms
 - dummyAlarm
 - AlarmGroupID: StartAlarms
 - AlarmChannelName: Mu2e_TDAQ_SHIFT_EXAMPLE_INTERFACE0/dummyRegister0
 - IgnoreOnSeverity: False
 - LinkToNewAlarmsMonitorTable-GroupID = DEFAULT-DISCONNECTED
 - LinkToStartAlarmsMonitorTable-GroupID = DEFAULT-DISCONNECTED
 - LinkToRunningAlarmsMonitorTable-GroupID = DEFAULT-DISCONNECTED

Class: ots:SlowControlsDashboardSupervisor

Id: 282

Instance: 1

Network: local

Group: daq

Module: \$OTSDAQ_UTILITIES_LIB/libSlowControlsDashboard.so

ConfigurePriority: 0

StartPriority: 0

StopPriority: 0

LinkToPropertyTable-GroupID = DEFAULT-DISCONNECTED

MacroMakerSupervisor

CodeEditorSupervisor

Slow Controls Monitoring in otsdaq

Examples

Editor

Slow Controls Dashboard

File EditMode

Mu2e_Weather_2/humi

Mu2e_Weather_2/temper...
Status: NO_ALARM
Severity: NO_ALA

PV Name	value	Alarm
Mu2e_Weather_1/so...	5.0 W/r	
Mu2e_Weather_1/te...	2 degC	
Mu2e_Weather_1/te...	36.3 de	
Mu2e_Weather_1/wi...	17 mph	
Mu2e_Weather_2/b...	1016.0	
Mu2e_Weather_2/pr...	0.0 inch	

Mu2e_Weather_1/temperatur...
Status: NO_ALARM
Severity: NO_ALARM

Editor Panel

Choose your widgets:

Grid Color
Background
Default Values

Name: Basic Root file viewer
Type: Root file

Example of widget settings window

Slow Controls Dashboard

widget-0

Parameter	Value
class	undefi
border	false
text	TDAQ
text_position	left
font	arial

Slow Controls Dashboard

widget-0

Chose PV names

Mu2e:TDAQ_hwdev_DTC0_BurstDataCount
Mu2e:TDAQ_shift:ExampleInterface0:dummyR
Mu2e:TDAQ_shift:ExampleInterface0:dummyR
Mu2e:TDAQ_shift:ExampleInterface0:dummyR
Mu2e:TDAQ_shift:ROC0:dummyRegister0

Add Remove

PV names chosen

Mu2e:TDAQ_shift:ExampleInterface0:dummyR
Mu2e:TDAQ_shift:ExampleInterface0:dummyR
Mu2e:TDAQ_shift:ExampleInterface0:dummyR

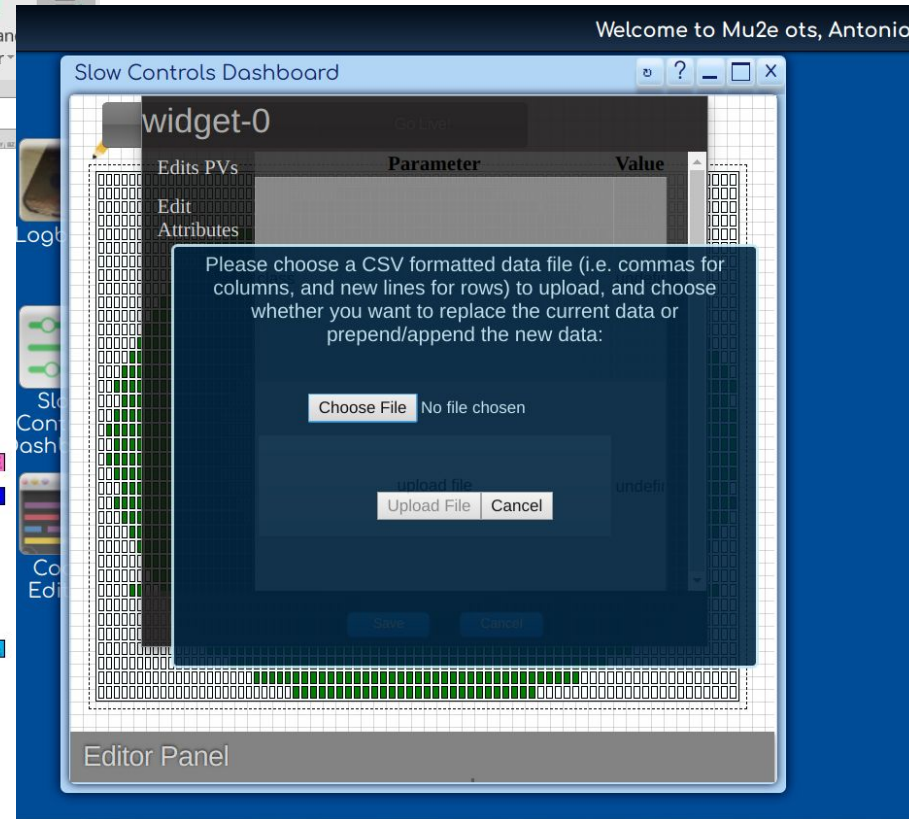
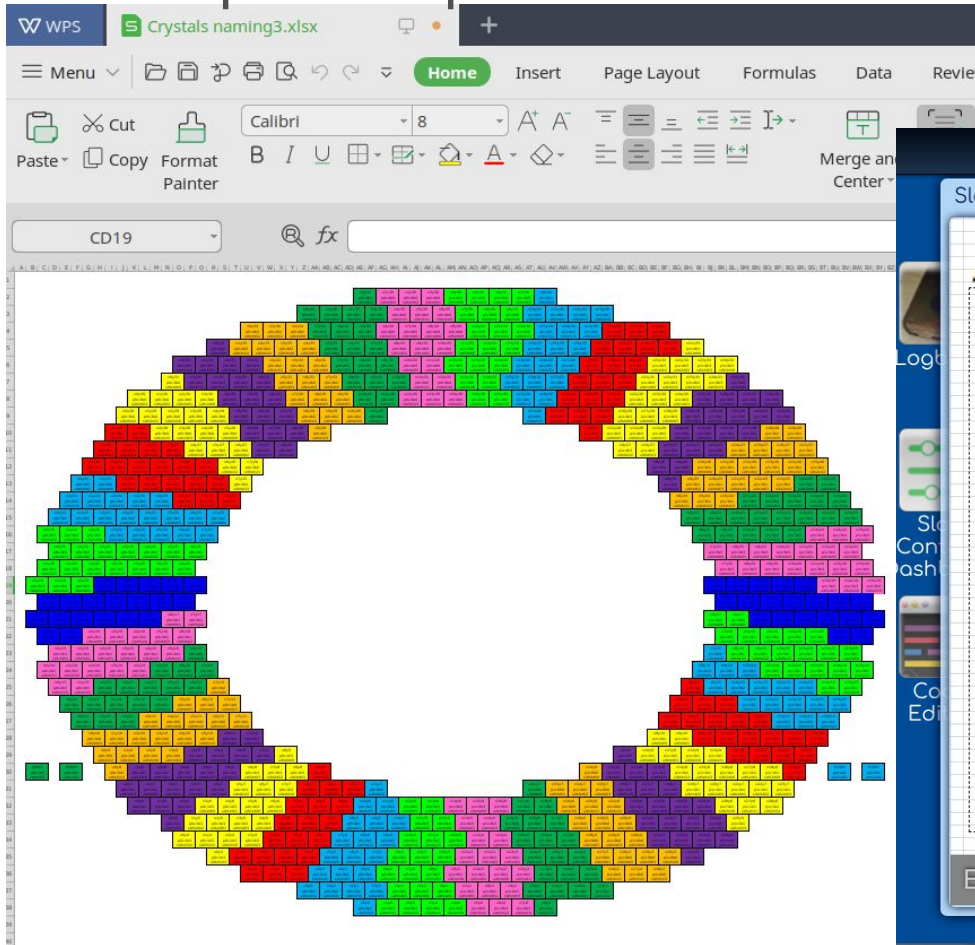
Save Cancel

Editor Panel

widget attributes editor

Calorimeter monitoring and the Slow Controls GUI

Examples: Import an xls file in a 2D-stop light widget



Slow Controls alarm notification by System Message

System message alarm notification example

The screenshot displays the Mu2e Slow Controls Dashboard interface. At the top, a navigation bar includes buttons for 'Layouts', 'Tile', 'Show Desktop', and 'Full Screen', along with a 'Welcome to Mu2e ots, Antonio' message and a 'Sign out' link. A system message notification is shown in a blue box on the left, stating: 'System Message Received at 20:01:19', 'Slow Control Alarm Notification: PV: Mu2e_TDAQ_shift_ExampleInterface0/dummyRegister0', 'at time: Mon Mar 30 13:01:07 2020 value: 1233 stouts: HHH severity: MAJOR'. A 'Dismiss' button is located at the bottom right of the notification box. The main dashboard area features a 'Slow Controls Dashboard' window with a 'File Manager' and 'Switch to Edit Mode' button. Below these buttons is a table with the following data:

PV Name	Alarm Status	Alarm Severity	Last Update
Mu2e_TDAQ_shift_ExampleInterface0/dummyRegister0	HHH	MAJOR	03/30/20

The dashboard also includes a sidebar with icons for 'Config Subsets', 'Macro Maker', 'DB Utilities', 'Console', 'System Status', and 'Slow Control'.

Slow Controls alarm notification by System Message

Configured by specific table in otsdaq:

