Hall C EPICS
Slow Controls and Monitoring System

Detector Support Group

Peter Bonneau (Lead), Mary Ann Antonioli, Pablo Campero,
Brian Eng, Amanda Hoebel, and Tyler Lemon

Contents

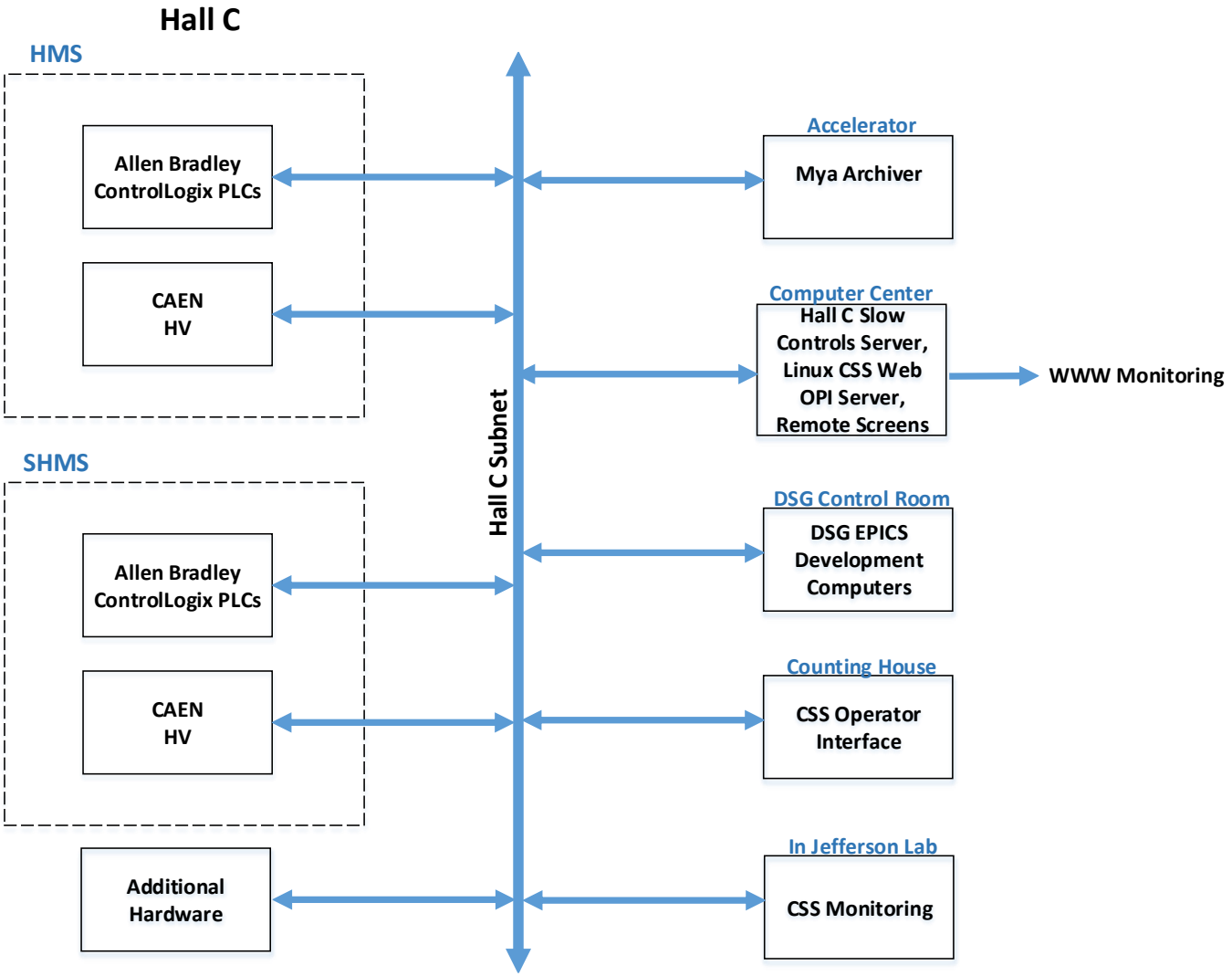
- Project Objectives
- System Architecture
- Control System Studio (CSS)
- Mya Archiver
- EPICS Framework
- Project Status
- Conclusion

Project Objectives

Develop integrated EPICS-based controls and monitoring systems for HMS and SHMS

- User interface screens
  - Clear and consistent
- Remote monitoring
  - Via web browser
- Integrated alarm handler system
  - Auto-emailing alarm notifications
  - Alarm logging
  - User guidance when alarms occur
- Archiving of signals
  - Easy user interface
- Use similar EPICS slow controls framework as in Halls B & D
  - To have consistency within Physics Division
  - Enables DSG expertise to support and maintain EPICS across Physics Division

Hall C Slow Controls Architecture



CS-Studio Overview

Control System Studio (CSS)

Open source Java-based collection of *integrated* software tools to monitor and operate large scale control systems

- CSS developed by Oak Ridge, Brookhaven, Lawrence Livermore, Michigan State, DESY
- CSS components being developed
  - BOY - User Interface screen
  - WebOPI - Remote monitoring
  - BEAST - Alarm handler system
- Other components under investigation
  - Data browser
  - Logbook
  - Diagnostic tools
- At JLab, used by Hall B and Hall D

CSS/BOY User Interface

- CS-BOY (Best OPI, Yet) features
  - An Operator Interface (OPI) development and runtime environment
  - Works like web browsers
  - .OPI file is a regular XML file that can be edited in OPI editor or text editor and run in OPI Runtime
  - Dynamic OPIs can be developed via PV-triggered scripts or rules
  - Comprehensive set of widgets

CSS/BOY Development Environment

All-In-One workbench for OPI editing

Navigator

- Browser shows all files in CSS workspace

Outline

- Overall view of screen under development

Toolbar

- CSS tools
  - Create file, arrange widgets, zoom

Editor

- Workspace where widgets are placed to develop screens

Console

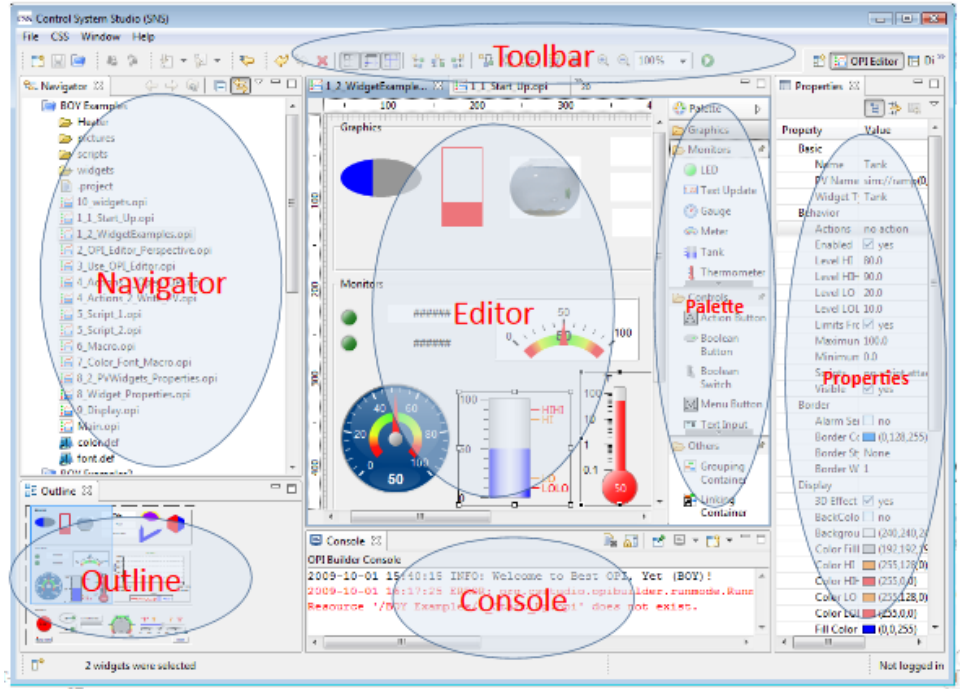
- Prints error and status messages during development and running

Palette

- List of widgets used to monitor, control, and organize screen

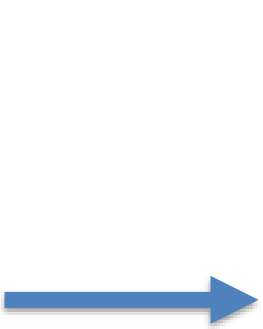
Properties

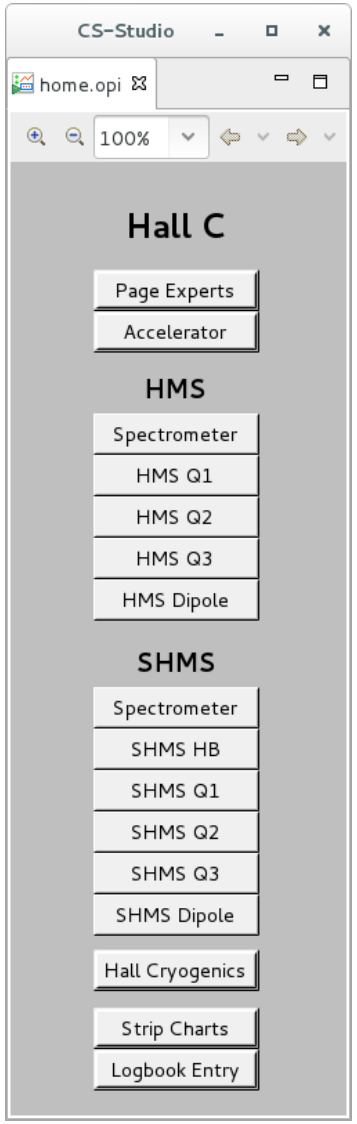
- All settings for a selected widget



CSS/BOY Run-Time Environment

- Run during operations by end users
- Run-time environment starting script
  - Specific for Hall C (*HallC-CSS*)
  - Runs on Hall C slow controls server
  - Generates temporary CSS Workspace for each CSS-BOY session
  - Provides consistent behavior and user experience
- Opens with a top-level menu window
  - Lists major systems with links to tree-structured sub-menus

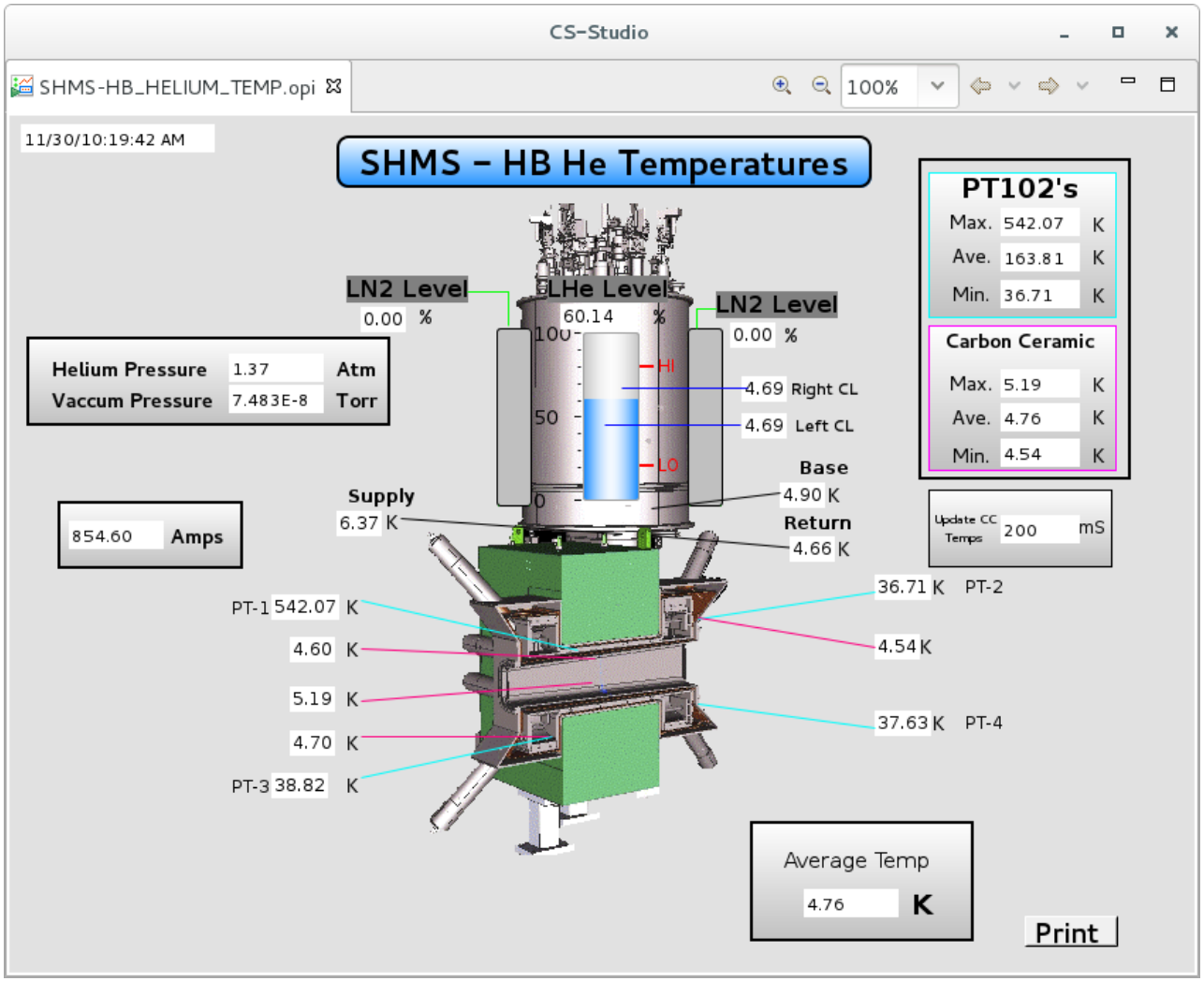




Steps to Develop and Install CSS/BOY

1. Export PV names from PLC HMI screens
   - 63 HMS HMI screens
   - 210 SHMS HMI screens
2. Add diagrams and background images to CSS screen
3. Add indicators to CSS screen for each PV
4. Assign PVs to indicators
5. Add rules and scripts to indicators
6. Develop CSS starting scripts
7. Integrate CSS-BOY screens into slow controls server
8. Update server's channel-access settings to point to appropriate IOC or gateway

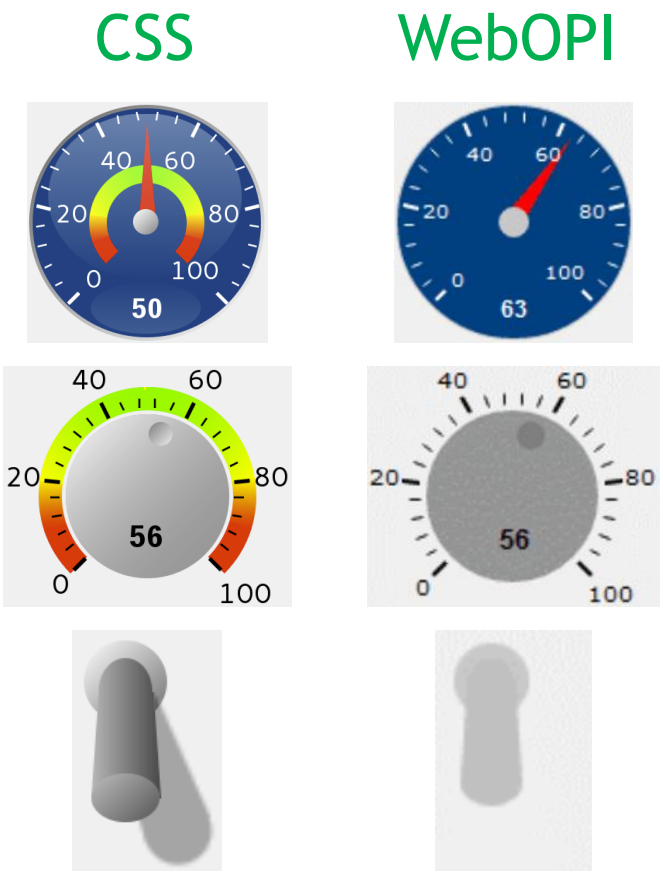
CSS/BOY User Interface



Working CSS-BOY OPI for SHMS Horizontal Bender magnet developed by DSG
Screen is replicated from current PLC HMI screen

CSS/WebOPI Remote Monitoring

- Allows CSS screens to be viewed in web browser
- Read-only EPICS channel access
  - No screen development capabilities
  - Requires CUE login for authentication
- Runs as application in a Java servlet
  - Tomcat recommended for servlet
- Existing CSS screens can be reused with adjustments
  - WebOPI does not support 3D effects, dashed lines, or color gradients
  - Some inconsistencies in text size between WebOPI and CSS



Comparison of widgets in CSS and WebOPI showing WebOPI's lack of 3D shading and color gradients.

Steps to Develop and Install WebOPI

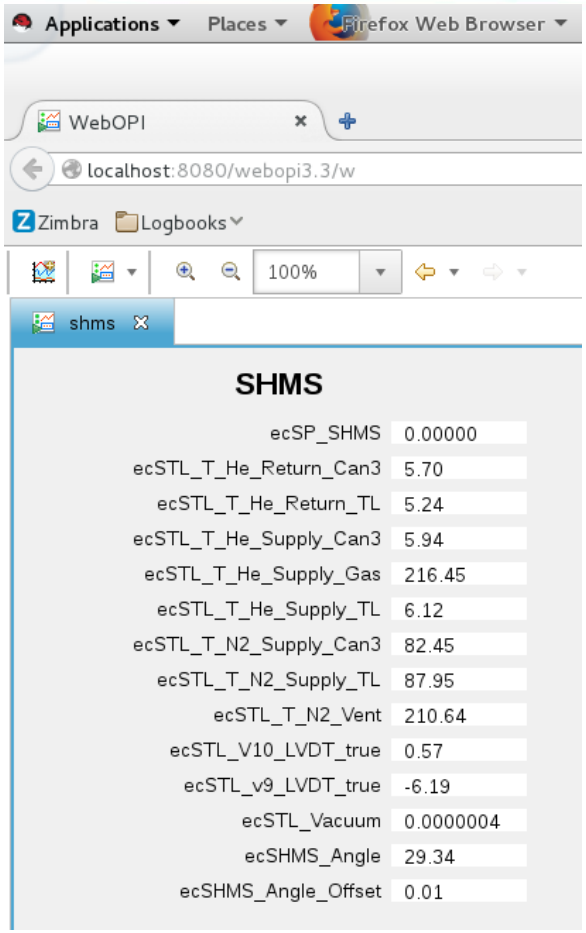
1. Develop channel-access gateway
   - **Status**:
     - Gateway set up by accelerator to read from Skylla7
     - To be replaced by DSG-developed server

2. Develop CSS screens
   - **Status**: In progress

3. Set up server in computer center
   - Install Java servlet that will run WebOPI on server
   - **Status**:
     - Investigating server requirements
     - Tomcat installed on DSG PC for initial development

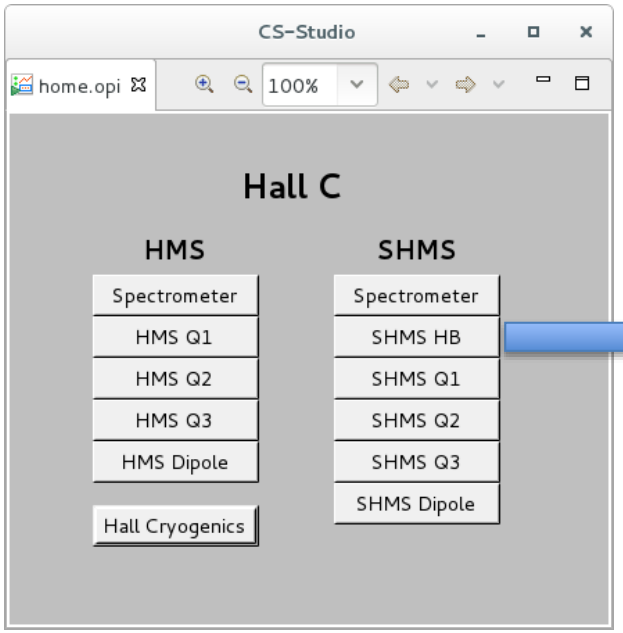
4. Configure and run servlet with WebOPI
   - Load WebOPI's web-archive (.war) file into servlet
   - Create configuration file for WebOPI
     - Set .opi files' storage path
     - Set EPICS channel access settings
   - **Status**:
     - Complete on DSG development PC
     - WebOPI set to run on servlet at URL localhost:8080/webopi3.3

5. Verify screen layouts on WebOPI
   - Correct text clipping and sizes caused by .opi font inconsistencies between hosts
   - **Status**: In progress

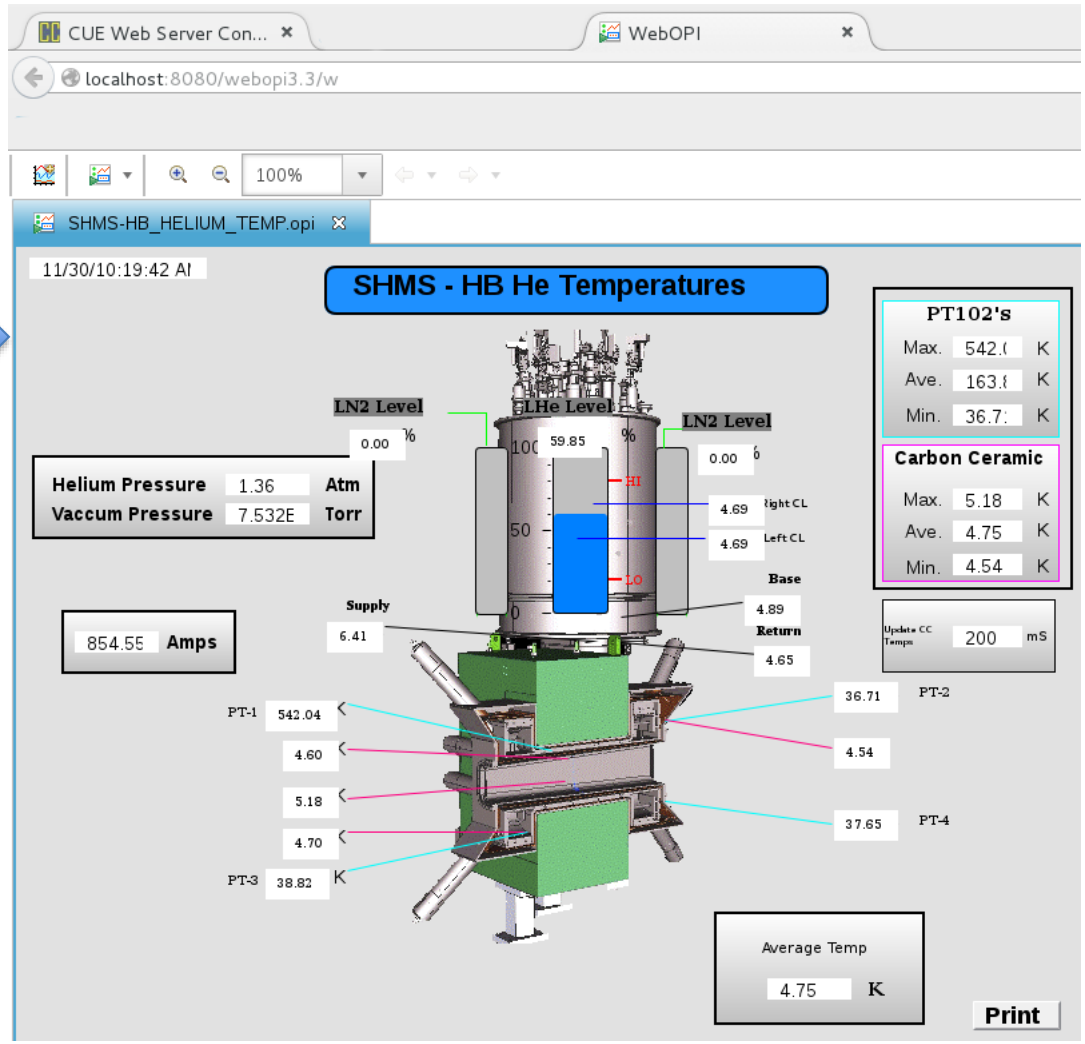


Preliminary CSS-BOY screen for SHMS running in WebOPI
WebOPI is running on Tomcat servlet on DSG PC at localhost URL

Operational WebOPI Screens



Hall C WebOPI Main Menu

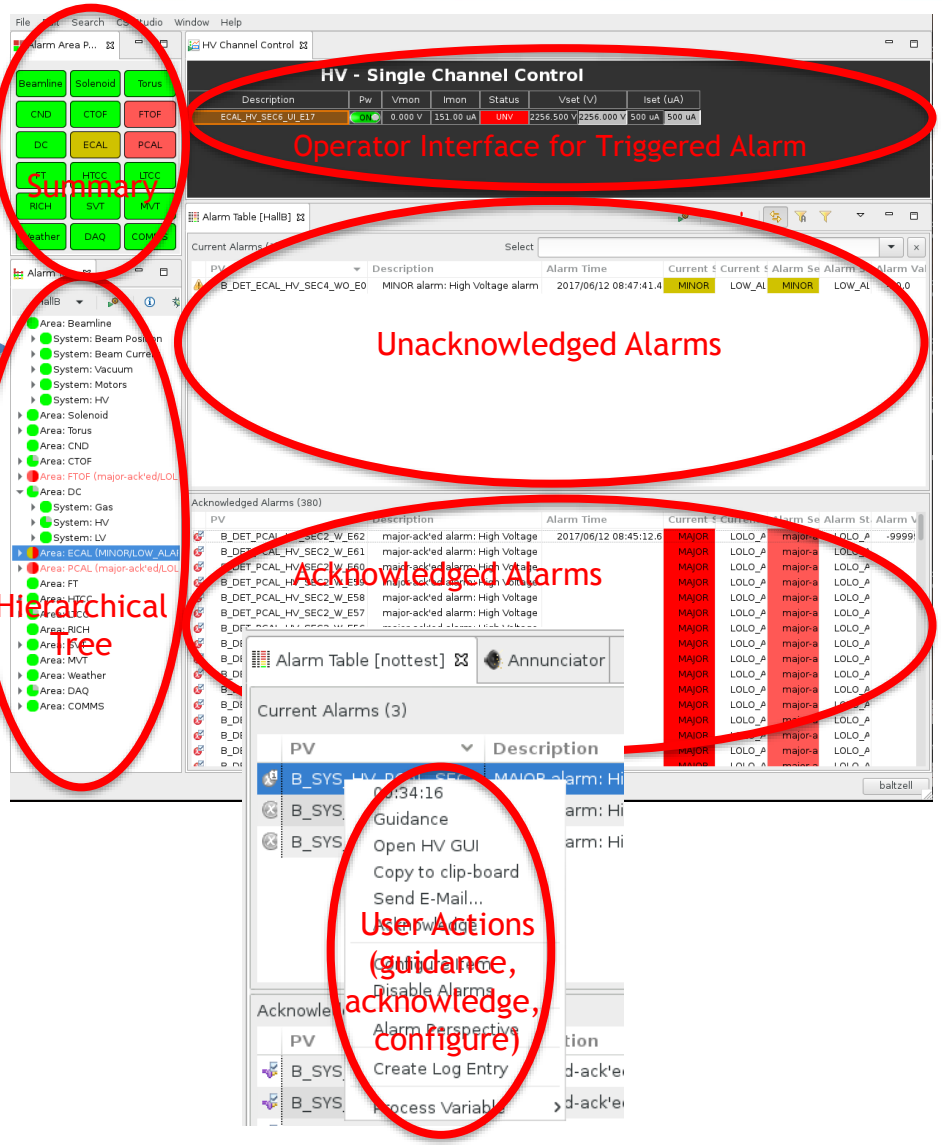


WebOPI screen for SHMS HB magnet (with live data)

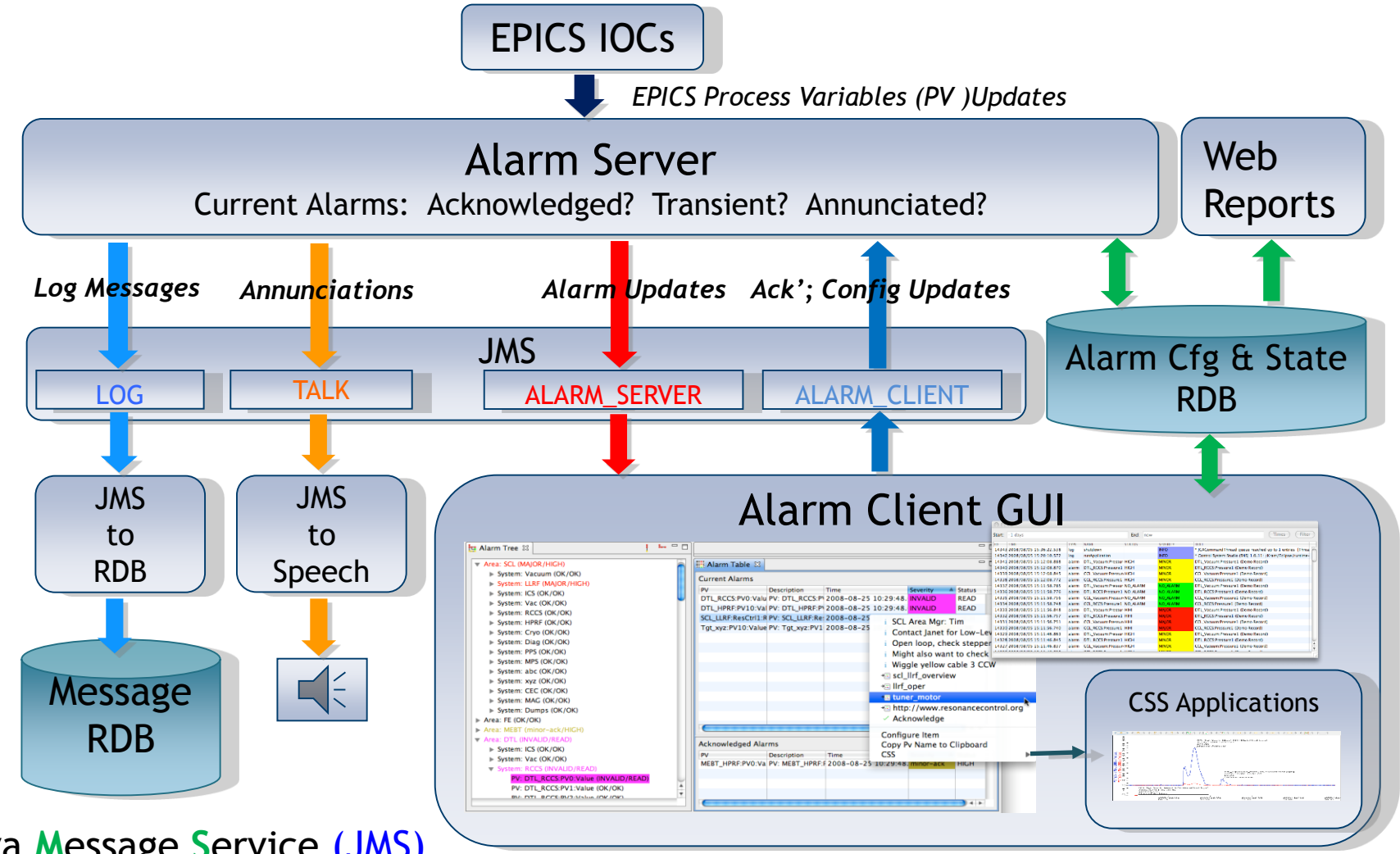
CSS/BEAST Alarm System

Best Ever Alarm System Toolkit

- Distributed alarm system
  - Alarm Server
    - Uses EPICS channel access to monitor alarm triggers in control system
  - CSS user interface
    - Views current alarms as table or hierarchical tree
  - Relational Database
    - Configures and logs
  - Web reports
    - Analyzes number and frequency of alarms, search alarm configurations
- User interface features for operators
  - Access guidance on how to handle specific alarms
  - Invoke links to related operator interfaces or other CSS tools for the alarm trigger PVs
  - Acknowledge alarms
  - Edit configuration



CSS/BEAST Alarm System



Java Message Service (JMS)

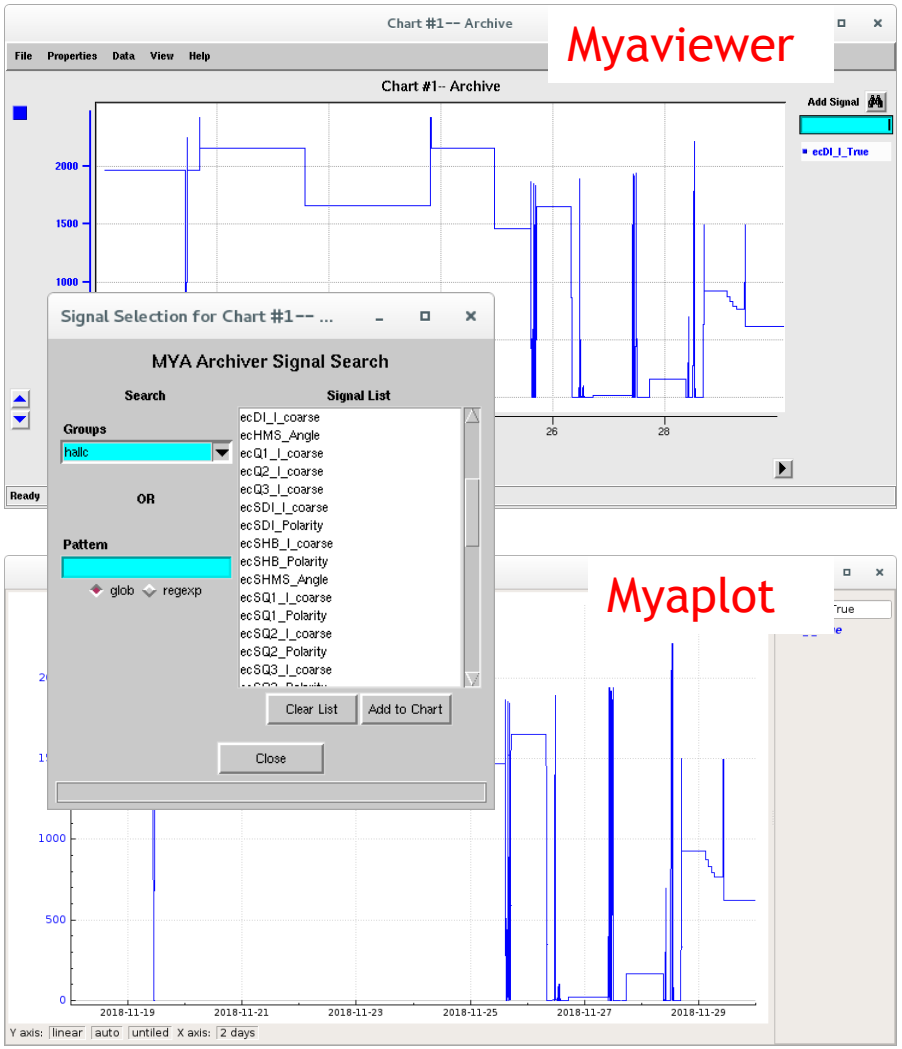
Relational Data Base (RDB)

Steps to Develop and Install CSS/BEAST

1. Generate CSV of all PVs for alarm handler
2. Develop Python script to convert CSV data to BEAST XML file
3. Develop JMS-to-RDB routine for message logging
4. Develop JMS-to-annunciator routine
5. Configure “alarm configuration and state” RDB to generate web reports
6. Develop CSS GUI for BEAST

Mya Archiver

- Hardware
  - Accelerator's servers
    - Maintained by Accelerator
- Software
  - JLab's Mya Archiver
    - Myaviewer & Myaplot user interfaces
    - PVs archived with "dead-bands"
    - Command line tools (myData, mySampler) to dump archive history to ASCII tables
    - Organized in "groups"



EPICS Framework

Hall C Slow Controls Server

- Slow control server cost ~$4800
  - Similar to Hall B slow control servers
- Will run EPICS base and support services
- JLab Computer Center will maintain
  - Computer hardware
  - Operating system and monitoring software
    - RedHat Enterprise Linux (RHEL7)
    - Monitored by Nagios with regular automatic checks
      - cpu/disk/memory usage
      - necessary software running (e.g. alarm server)
      - email notification

EPICS Framework

Hall C Slow Controls Server

EPICS and Supporting Software

Software	Function
EPICS Base	Main core of EPICS • Build system and tools • Common and OS-interface libraries • Channel Access client and server libraries • Static and run-time database access routines • Database processing code and standard record • Device and driver support
CSS-BOY	User Interface screen operations and starting scripts
CSS-WebOPI	Remote monitoring via Web
CSS-BEAST	Alarm System

EPICS Framework Cont.

Hall C Slow Controls Server

EPICS and Supporting Software

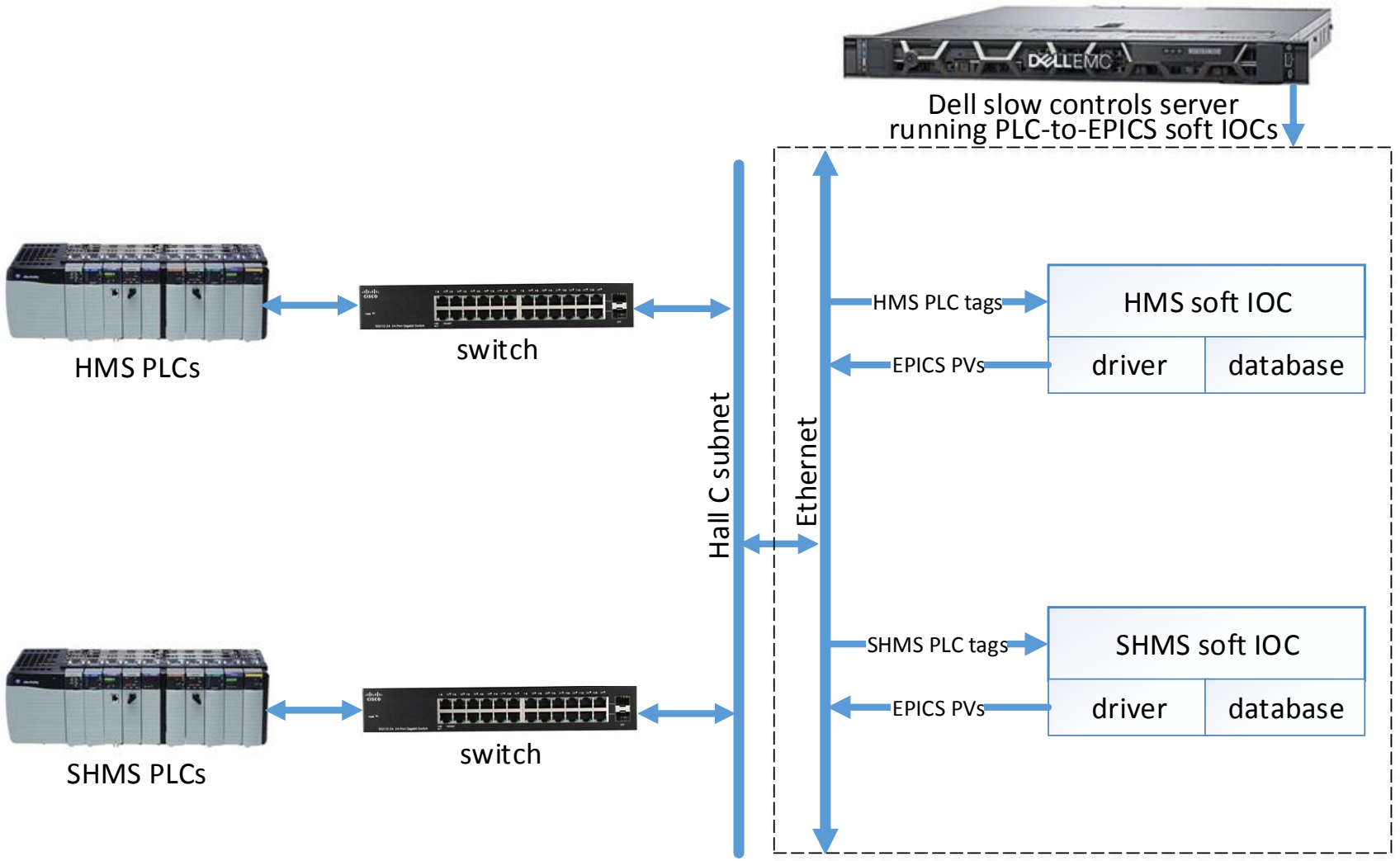
Software	Function
PLC to EPICS	Interfaces Allen Bradley PLCs to EPICS
SoftIOCs	Software-based EPICS input/output controllers
IOC Health Monitor	Monitors health of EPICS input/output controllers
BURT	Saves/restores EPICS PVs
Boot Server	For loading OS and databases on hardware IOCs
procServ	• Wrapper application that runs any process (e.g. a softIOC, a CA Gateway) in the background • Manual or automatic on start, crash, or exit • Logging supported

PLC to EPICS

PLC to EPICS SoftIOCs

- Software to convert PLC tags into EPICS PVs over Ethernet
  - *ether_ip* device EPICS support driver
    - A support module that interfaces Allen Bradley ControlLogix PLCs with EPICS via Ethernet
      - Used by Hall B and Hall D
  - Python configuration scripts
    - Generates the PLC EPICS databases
    - Separate script generates alarm handler configuration
  - PLC EPICS databases
    - Created from PLC tags and configuration parameters

PLC to EPICS


Dell slow controls server
running PLC-to-EPICS soft IOCs
HMS PLCs
switch
SHMS PLCs
switch
Hall C subnet
Ethernet
HMS PLC tags
EPICS PVs
HMS soft IOC
driver
database
SHMS PLC tags
EPICS PVs
SHMS soft IOC
driver
database

Steps to Develop and Install PLC to EPICS

1. Generate merged data list from PLC tags from HMS & SHMS in RSLogix 5000 (.ACD) files
   - Python scripts will use PLC network tags to generate EPICS databases
2. Determine optimized number of SoftIOCs for PLCs
   - Minimum of two – HMS & SHMS
3. Develop Python scripts to convert tag data list into EPICS databases
4. Determine EPICS process variable aliases for PLC tags
5. Use Python scripts to convert tag data list into EPICS databases
   - EPICS record fields auto-generated

Steps to Develop and Install PLC to EPICS (cont.)

6. Develop SoftIOCs initialization scripts using *ether_ip* EPICS support driver and PLC EPICS databases
7. Install and configure PLC to EPICS SoftIOCs on slow controls server
8. Configure slow controls server to auto-start the SoftIOCs
9. Configure slow controls server to monitor the SoftIOC processes

Project Status

Tasks Completed

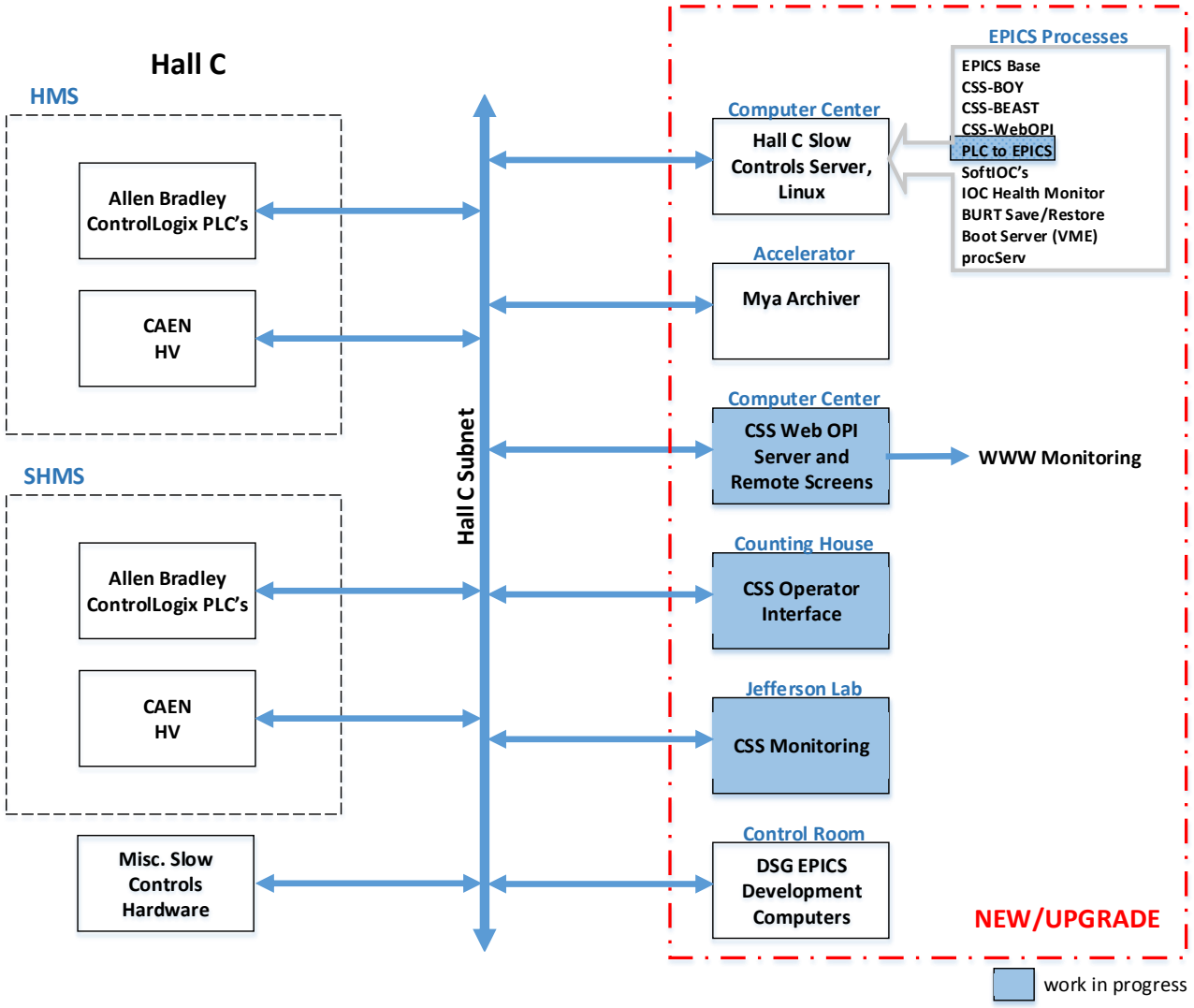
- Built development computers
  - Installed RedHat Enterprise Linux (RHEL) & CUE
  - Installed EPICS Base
  - Configured and installed CS-Studio
- Generated first working Hall C CS-Studio OPI screens
  - Hall C Main Menu
  - SHMS HB Magnet screen
  - SHMS overview displays key spectrometer PVs

Project Status

Work in Progress

- CS-Studio OPI screens
  - HMS & SHMS operator screen development
- CS-Studio WebOPI Remote Monitoring
  - Apache Tomcat Web server development
  - Remote OPI screens development
- EPICS framework
  - Linux-based PLC Tag to EPICS PV Interface

Project Status



Conclusion

DSG is developing an EPICS-based controls and monitoring system for HMS and SHMS

System Features

- CS-Studio BOY user interface screens
- CS-Studio WebOPI remote access monitoring
- CS-Studio BEAST alarm handler system
- MYA Archiver
- Similar to EPICS slow controls framework in Halls B & D

DSG Staff involved in this project

Mary Ann Antonioli, **Peter Bonneau (Lead)**, Pablo Campero, Brian Eng, Amanda Hoebel, and Tyler Lemon