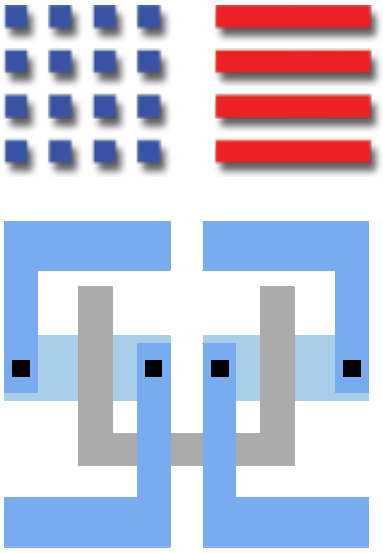
UNIVERSITÄT
HEIDELBERG

VXD Slow-Control Status



Michael Ritzert

michael.ritzert@ziti.uni-heidelberg.de

8th Belle II VXD Workshop

Trieste

09.09.2015

EPICS and CSS Builds

- For the CSS 4.1 branch, I changed the build process for CSS to only build a minimal code base locally.
- The rest is taken from the automatically updated upstream repositories.
- By now: No remaining user-visible changes expected (the only one was the missing VXD logo in the splash screen…).
- BUT: The automatic upgrade from the 4.0 branch fails. Reset your installation from the full distribution tar when CSS startup fails with „At most one workspace extension point, allowed, but found 2“.

- The EPICS RPM builds will switch to mock.
- No user-visible changes expected, but automatic stricter checking of dependencies.
- Still a bit of scripting to do before it can go live.

DHE (+ JTAG, …): libdevDHE.so

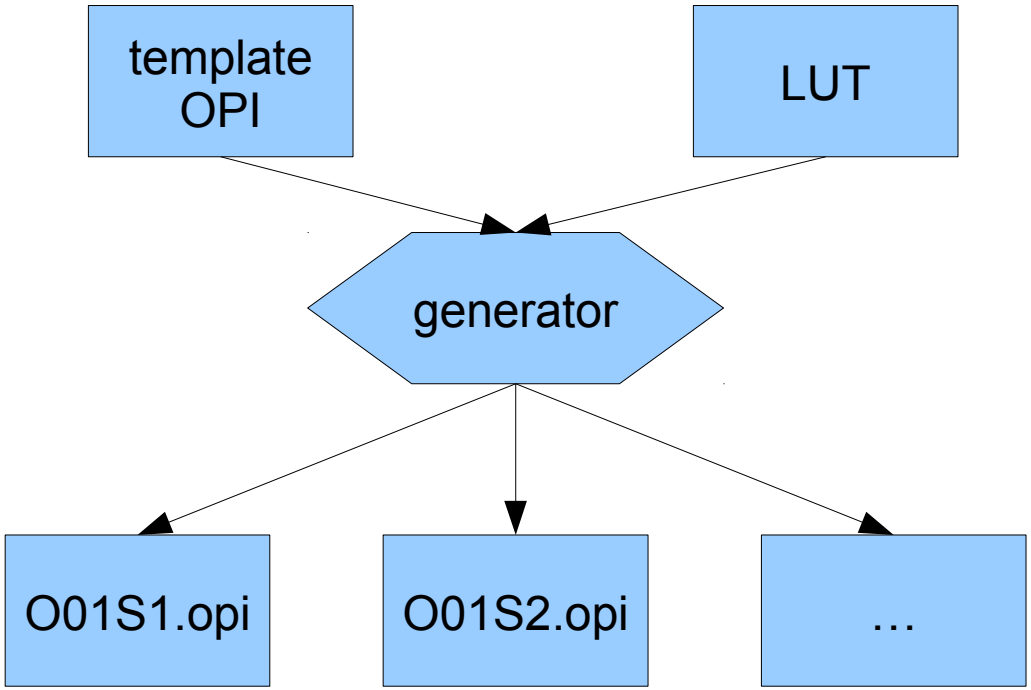
- Before: Lots of functionality provided by small standalone executables. Good for debugging, bad for rolling it out on a larger scale and non-expert use.
- To control the respective functionality through EPICS, everything has to be wrapped into an IOC.
- The actual functionality has been moved to a shared library linked from both the old executables and a new meta-IOC.
- The functionality is now also available through EPICS. For details see „DHE, Back-End Electronics“ tomorrow.

ONSEN: Struggles with the GUI

- The problem: The ONSEN GUI requires macros to set itself up, but macros cannot be changed during the runtime of an OPI.
- Bjoern's creative solution: Load the actual content into a linking container where the macros can be set beforehand.
  Works, but not easy to understand and therefore maintain (the above is simplified) and slow (resizes often).
- Easier solution requires changes to the CSS core – also see the discussion at https://github.com/ControlSystemStudio/cs-studio/issues/1280.
- The ideal solution to run macros early looks unrealistic to achieve in the near term.
- Providing a command to re-evaluate all macros should be possible. I'm „almost there“ for a while now, but especially updating the macros used in PV names is tricky…

ONSEN: Struggles with the GUI

- Proposal: Generate **static** OPI files from template + LUT
  (statically adding the macros to the OPI).
- Advantage: Also works when jumping in from alarm screen, etc.
- Quite easy to do: The OPIs are stores as XML files.
  ⇒ We can easily add a few lines to define the macros.

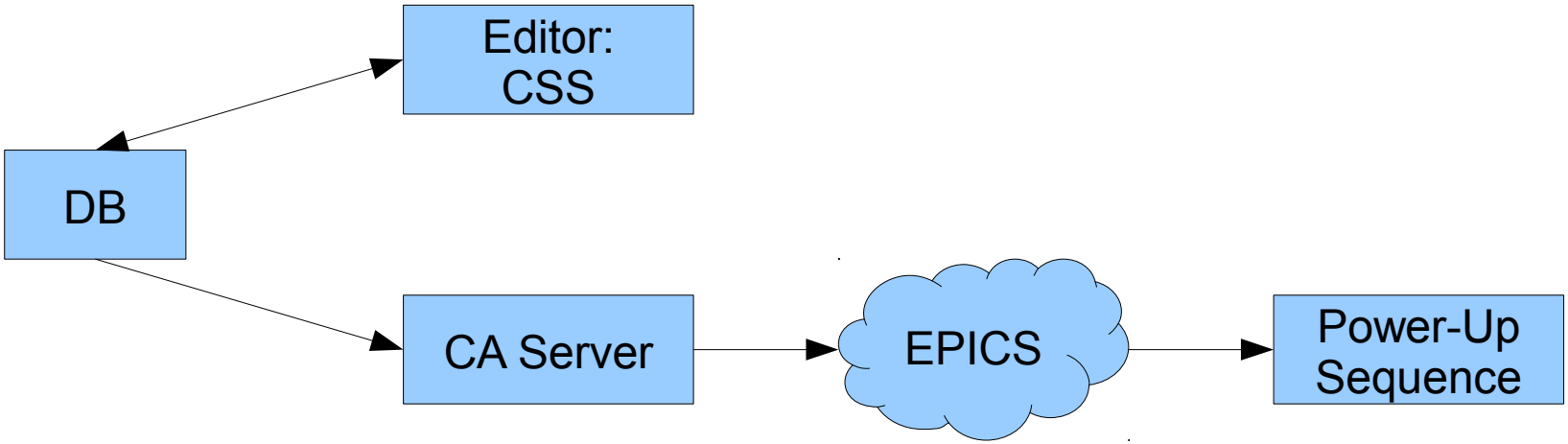


FOS: Including the Optical Multiplexer

- SM130 + Multiplexer: several fibres connected to one input of the interrogator. The mux position is changed automatically by the interrogator.
- When just asking the SM130 for a channel's data (not synchronized to the interrogator's clock), the multiplexer position is random ⇒ the current IOC would be unreliable.
- Solution: Put the SM130 in „streaming“ mode, sending data right as it is taken ⇒ all multiplexer positions always come by.
- Also offers the possibility to run an FFT on the incoming data (@ a few 100 Hz) for vibration frequency analysis.
  ⇒ to be published as a waveform record.
- In my tests with the hardware @ DESY, requesting spectra with streaming mode active on a second TCP connection worked fine.
- The IOC will be modified to get wavelength information from streaming mode and spectra via a second TCP connection.

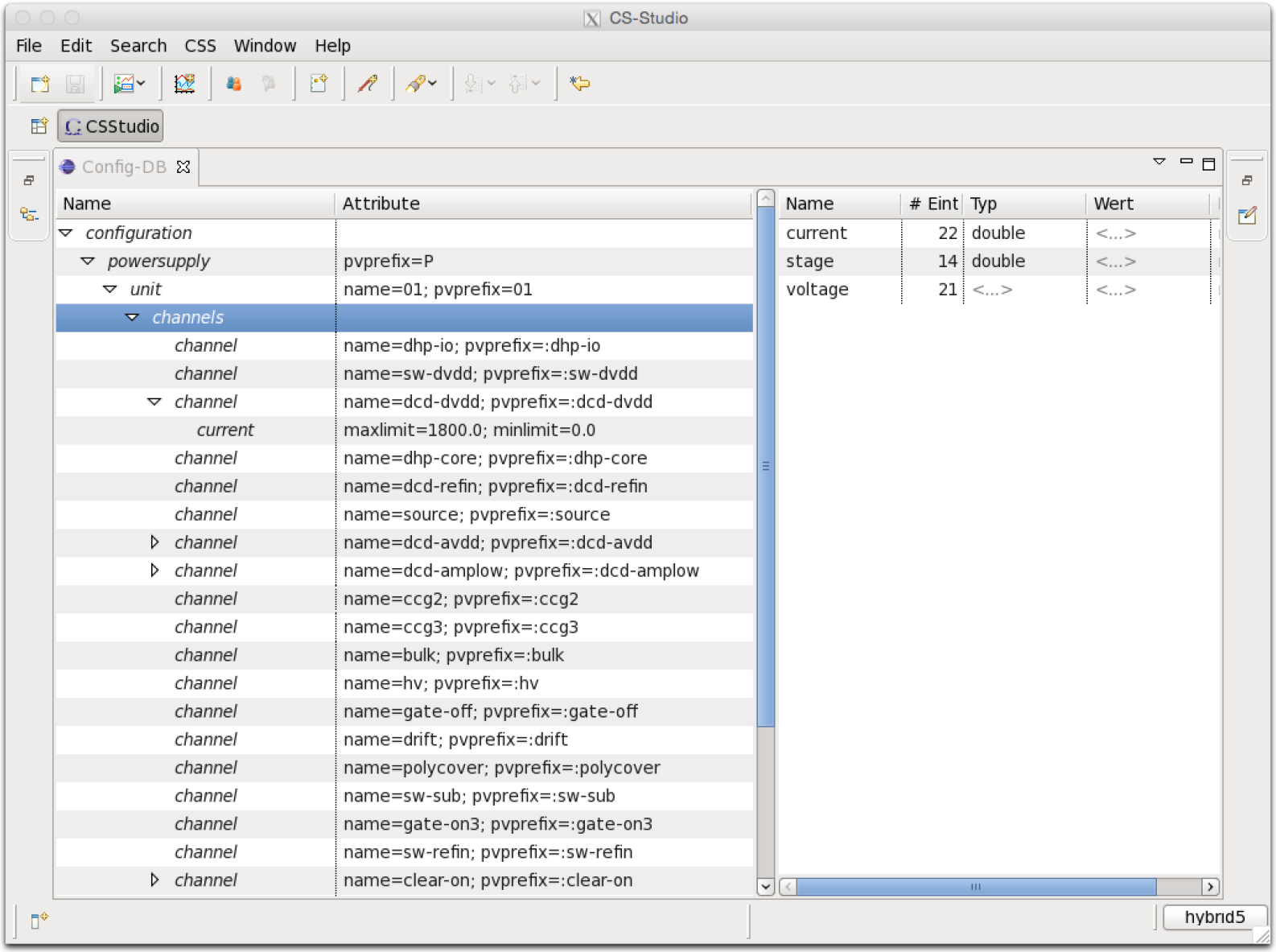
Configuration Database

- A configuration (set of PV values) is uploaded via the GUI included in CSS ⇒ Unique ID.
- Each possible Belle II run type is associated with a PXD configuration ID in a look-up table.
- On run start, the LUT is used to find the configuration to be used.
- The Config-DB CA server publishes the configured PVs.
- The power-up sequence uses these PV values as the values to upload to the system.
- During the run, the read-back from the hardware is compared to the configuration.



Configuration Database II

- PS configuration data loaded in the default Config DB GUI



Configuration Database III

- Alternative interface: The configuration data is accessible via PVs in CSS; a Jython interface to trigger actions („load“, „save“) is under development.
  ⇒ Build your own GUI!
- Available PVs:
  config://data:… The actual data.
  config://info:… Commit-ID, Commit-Text, Filename, etc.
  - branch_id
  - branch_name
  - *commit_author*
  - *commit_date*
  - commit_id
  - commit_text
  - file_description
  - file_id
  - file_name
- All data is run through validators before it can be commited.
  We can implement (in Java) custom validators for all systems.

„Injection Inhibit“ Flag

- Belle II-wide signal prohibiting injection to the empty ring (not continuous injection): NORMAL_INJECTION_ENABLE.
- PXD to disallow injection during first fill, when
  - any HV is on to prevent radiation damage to the detectors.
  - radiation monitoring is off
  - we want to do a local calibration run.

- This is in addition to the beam abort signal from the radiation monitoring.

Thank you!

Schedule, Milestone Discussion

For next testbeam @ DESY:

- Normal run start via the CSS GUI without intervention on the command line.
- Provide all required code to have it installed via RPM.
- Update the PV naming convention document.
- ONSEN integrated into RC.
- Environmental interlocks?
- Full archiving
- Setup:
  - Full EPICS/CSS infrastructure including CA Gateway.
  - Prepare performance tests with CA Nameserver, Channel Finder.