UNIVERSITÄT
HEIDELBERG

PXD SC Overview
A **Brief** Introduction Into Our Plans

Schaltungstechnik
und Simulation

Michael Ritzert
michael.ritzert@ziti.uni-heidelberg.de

10th VXD Belle II Workshop
Santander
14.10.2016

EPICS

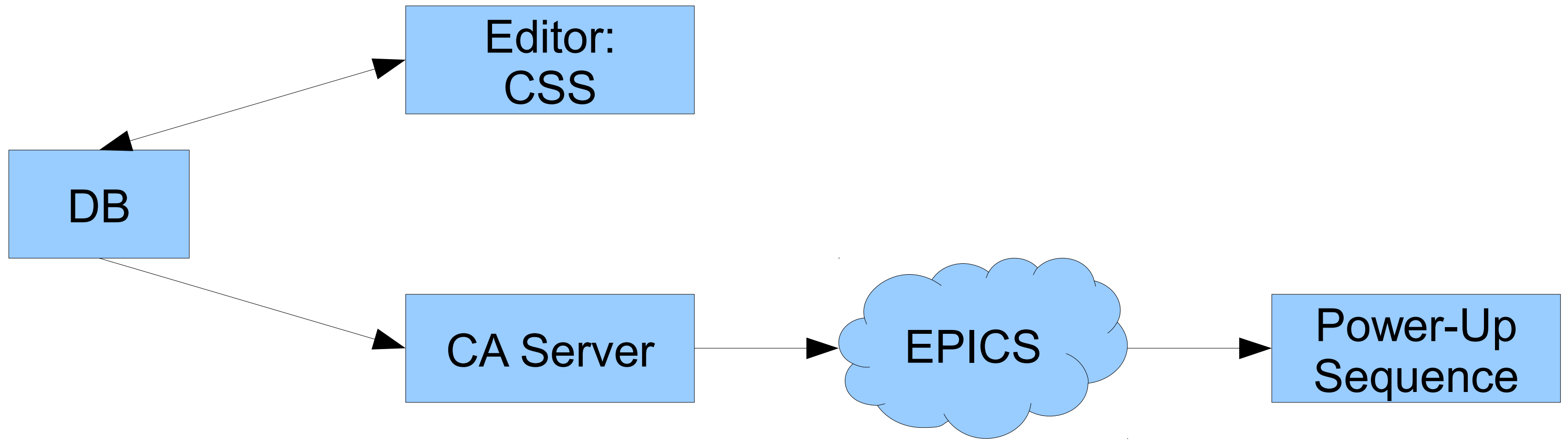
- Our OS for servers running IOCs is SL7 x86_64.
- EPICS base version 3.14 at the moment,
  3.16 (out soon) is under consideration.
- RPM repository with EPICS base, extension modules
  + own IOCs with automatic build from git repository.
  - Important rule for production servers: No software installations bypassing the RPM database, i.e. no pip install, etc.
  - „rpm -q -a“ should give a complete overview of the system.
    This is important when performing backups, or setting up replication servers.

CSS

- Custom build based on (currently) the **upstream branch 4.3**.
- Our own build includes:
  - Selection of common CSS modules.
  - Configuration database editor
  - NSM2 data source
  - DQM modules
  - Support for CERN UNICOS for IBBelle.
  - Default settings for archiver, alarm system, font and color definitions, etc.
  - Different installations (test system, DESY, KEK), use different configurations ⇒ the feature with the configuration changes.

Configuration Database

- Any information needed to start the run — mostly configuration parameters for the system — is taken from the configuration database ⇒ ensures reproducability.
- Data format: XML-like.
- Editor in CSS
- An IOC loads the active configuration and propagates it as PVs.
- The power-up sequence then applies the data to the hardware.
- When in physics mode, access to PVs in the hardware will by default be disallowed when the system is not idle.



PV Naming Conventions

- Goal: Define a scheme to encode as much information as possible in the PV name, considering EPICS limitations.
- The PV name should be short (EPICS length limit!), but easily legible
  - Needs some common definitions for structure, abbreviations, …
- Managed in the git repository.
- Main concepts:
  - Fixed structure of the PV name: device:datapoint:property[:function].
  - device: ONSEN, PS, DHH, … encoded as one letter, followed by device ID.
  - datapoint: defined individually for each device.
  - property: voltage, current, id, … with fixed encoding

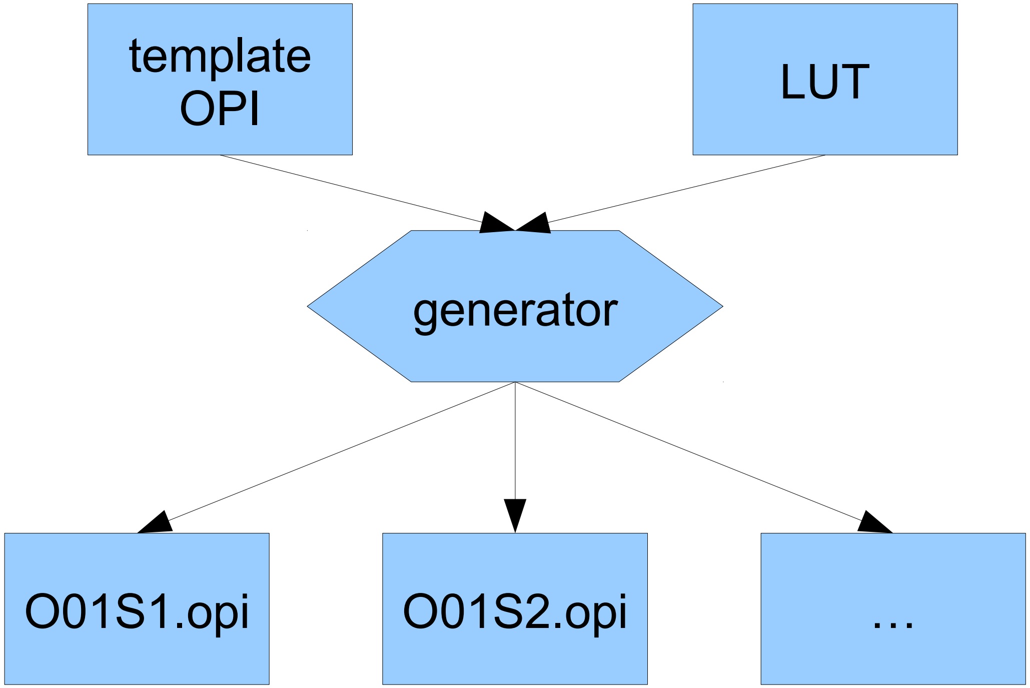
GUI Guidelines

- Document in git repository as the outcome of a dedicated meeting in April 2015.
- Covers many topics.
- But not really used in a realistic environment.
- At the meeting, we decided to pilot one implementation for a module, then re-convene to discuss changes to the guidelines based on actual experiences.
  - This has not happened so far.

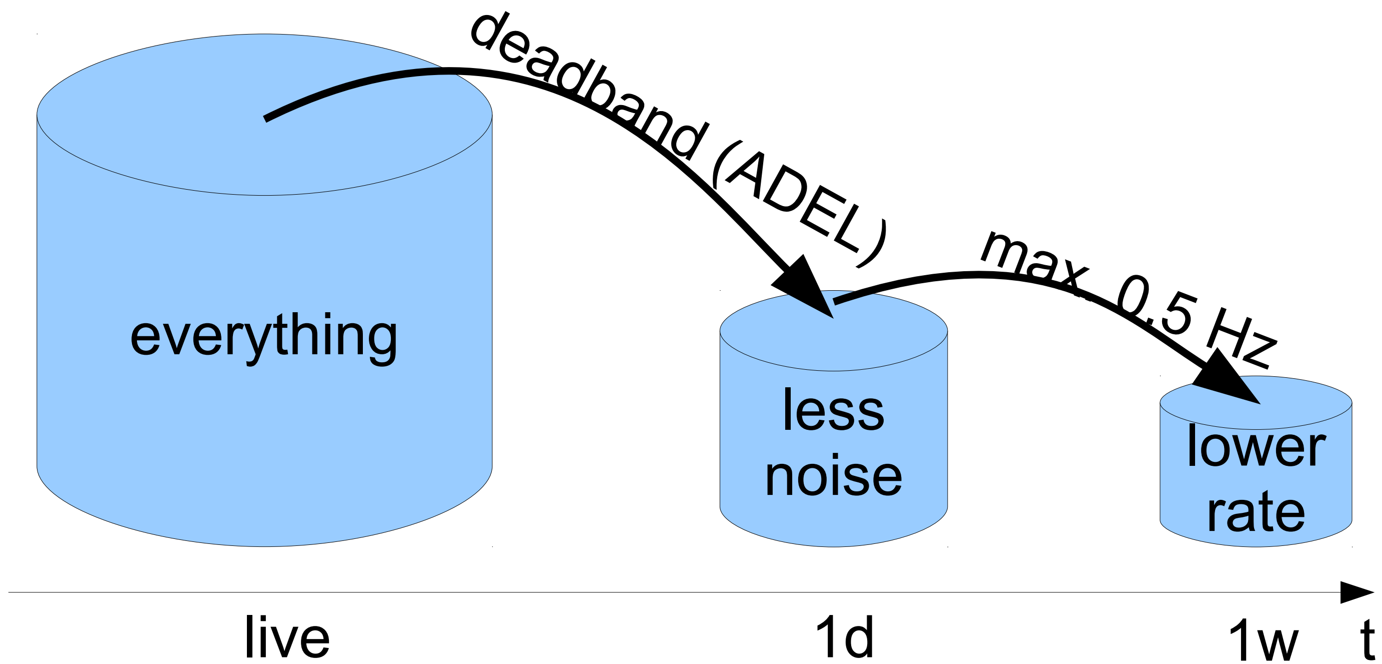
Contents

„Precompiled“ OPIs

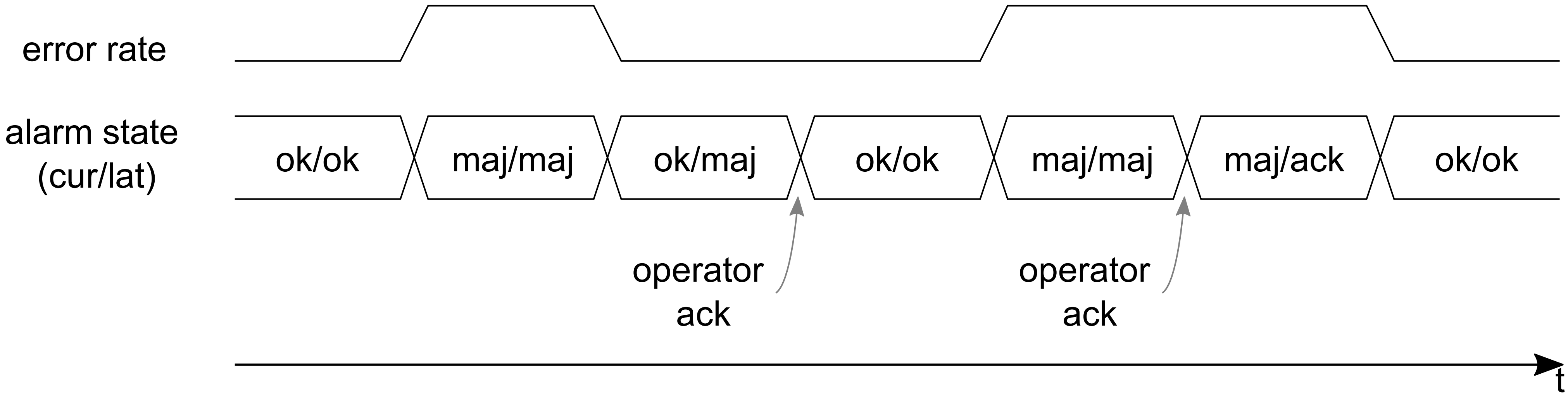
- Adding a sensible navigation (next module, previous module, …) to the OPIs proved difficult in CSS using only macros.
  ⇒ invented „precompiled OPIs“.
- A template OPI contains the actual contents, and a LUT (in a Python file) defines the „neighbors“ in the navigation.
- For each instance, a separate OPI is generated.
  ⇒ Also has advantages when jumping in from outside the hierarchy, e.g. from the alarm screen.



- We switched from the PosgreSQL-based archiver to the **Archiver Appliance**.
  - File-based instead of RDB-based ⇒ easier to backup.
  - Convenient HTTP interface returning data in CSV/JSON/… formats.
  - Multiple-tier storage allows us to reduce the data as they age: All data for a day, 1-sigma deadband for a week, max. rate of one sample every *x* seconds thereafter. Still *O*(20 TB) of data to expect during Belle II.

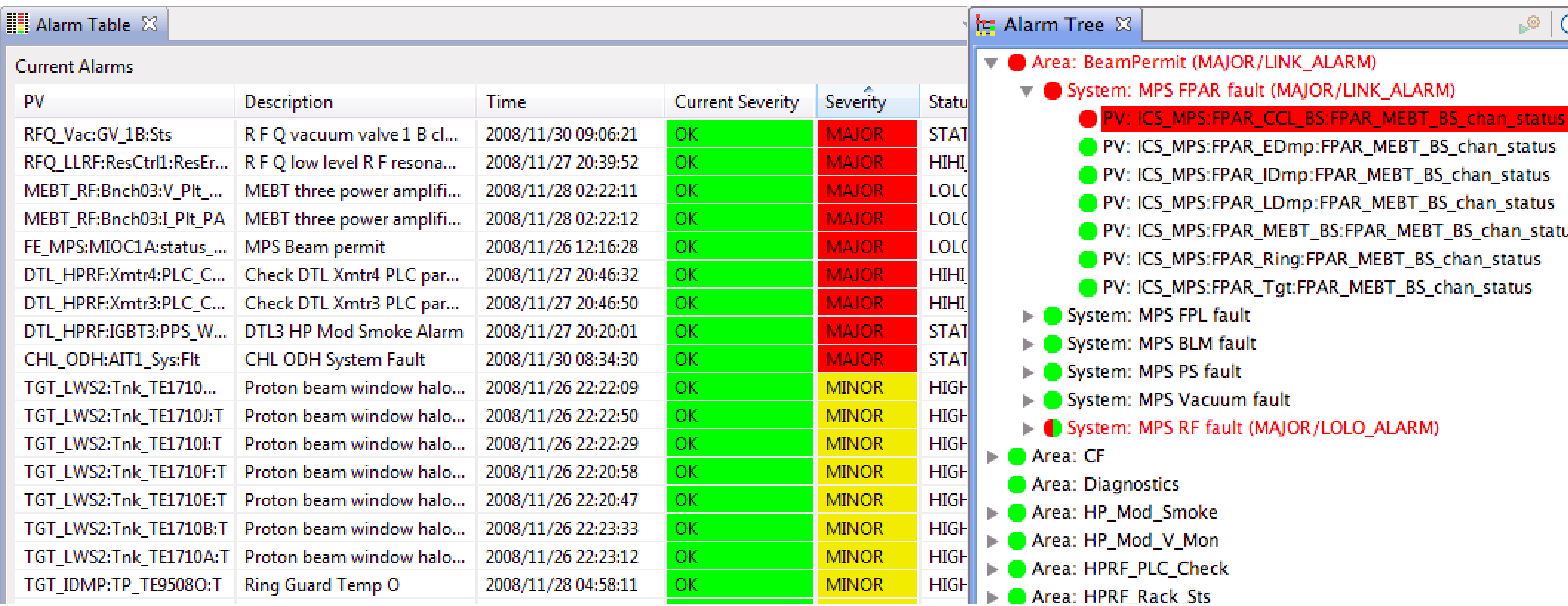


- The Best Ever Alarm System Toolkit, a.k.a. **BEAST** (← how could we use anything else? ☺).
- Manages alarm conditions in a tree-like structure, propagating alarms up to the root.
- Two states per alarm:
  - current condition, live from the PV
  - „latched“ condition: worst condition since last alarm acknowledge.
- Example alarm lifecycle:



Alarm System II

- The alarm server is a standalone executable
  ⇒ operates independently, backed by a PostgreSQL database.
- In CSS, the alarm module is used to display the alarm.
  – Displays as a tree, or as a table.
- Notifications by mail, sound in the control room are possible.



Alarm System III

- We collect alarms in an XML file converted to PDF.

2 IPMI

2.1 ONSEN

2.1.1 Errors reported by Shelf Manager

2.1.1.1 PSU failure	Power Unit failure	on device level: FRU, Voltage, Fuse	**A** shelf will switch off componentent **M** check remaining redundancy **M** if redundancy exists, mask error condition. **M** schedule replacement of the PSU	major
2.1.1.2 Fan failure	Fan Unit failure	on device level: FRU, Speed, Fuse	**A** shelf will increase other fan speeds **M** check remaining redundancy **M** if redundancy exists, mask error condition. **M** schedule replacement of the fan unit	major

- Columns so far:
  - Unique id, description, conditions,
  - actions: automatic and manual (shown to shifter on alarm condition)
- Clear definition of severities:
  - Major: system **is** broken
  - Minor: act **now**, or system **will** break.

Software Interlock

Inhibits unsave settings. E.g. condition: |clear-on – clear-off| ≤ 20 V

Off: OK

clear-on	0 mA	0 mA	50 mA	0 mV	0 mV	27000 mV
clear-off	0 mA	0 mA	50 mA	0 mV	0 mV	10000 mV

Difference 22V ⇒ inhibited

clear-on	0 mA	10 mA	50 mA	0 mV	22000 mV	27000 mV
clear-off	0 mA	10 mA	50 mA	0 mV	0 mV	10000 mV

Difference 19V ⇒ OK

clear-on	0 mA	10 mA	50 mA	0 mV	22000 mV	27000 mV
clear-off	0 mA	10 mA	50 mA	0 mV	3000 mV	10000 mV

- Implemented purely as EPICS DB with calc records.
  Cannot be circumvented.
- Still need to monitor the voltage.
  Hardware failures (shorts) are possible!

RC / PSC

- Commands from global run control / power supply control arrive via NSM2-to-EPICS gateway.
- PXD-intern: Fanout first to all subsystems, then to all instances, e.g. global → all PS → PS #14.
  - Any transition only completes, when all lower controls signal that it's complete.
  - Individual subsystems/devices can be put into „local“ mode to exclude then from global control.
    ⇒ They will not receive global commands and will not be considered during transitions.
- Implemented in IOCs using the EPICS sequencer.

RC / PSC II

- Sensors (PS + DHH): PSC
- DAQ (ONSEN + DATCON): RC

	OFF	STANDBY	PEAK
NOTREADY	DHH configured DAQ off	ASICs+DHH ready DEPFETs not powered DAQ off	Frontend sending data DAQ off
READY	DHH configured DAQ links established	ASICs+DHH ready DEPFETs not powered DAQ links established	Frontend sending data DAQ links established
RUNNING	DHH configured DAQ processing triggers, ROIs. **DHH sends dummy data**	ASICs+DHH ready DEPFETs not powered DAQ processing triggers, ROIs. **DHP sends dummy data**	Frontend sending data. DAQ processing triggers, ROIs.

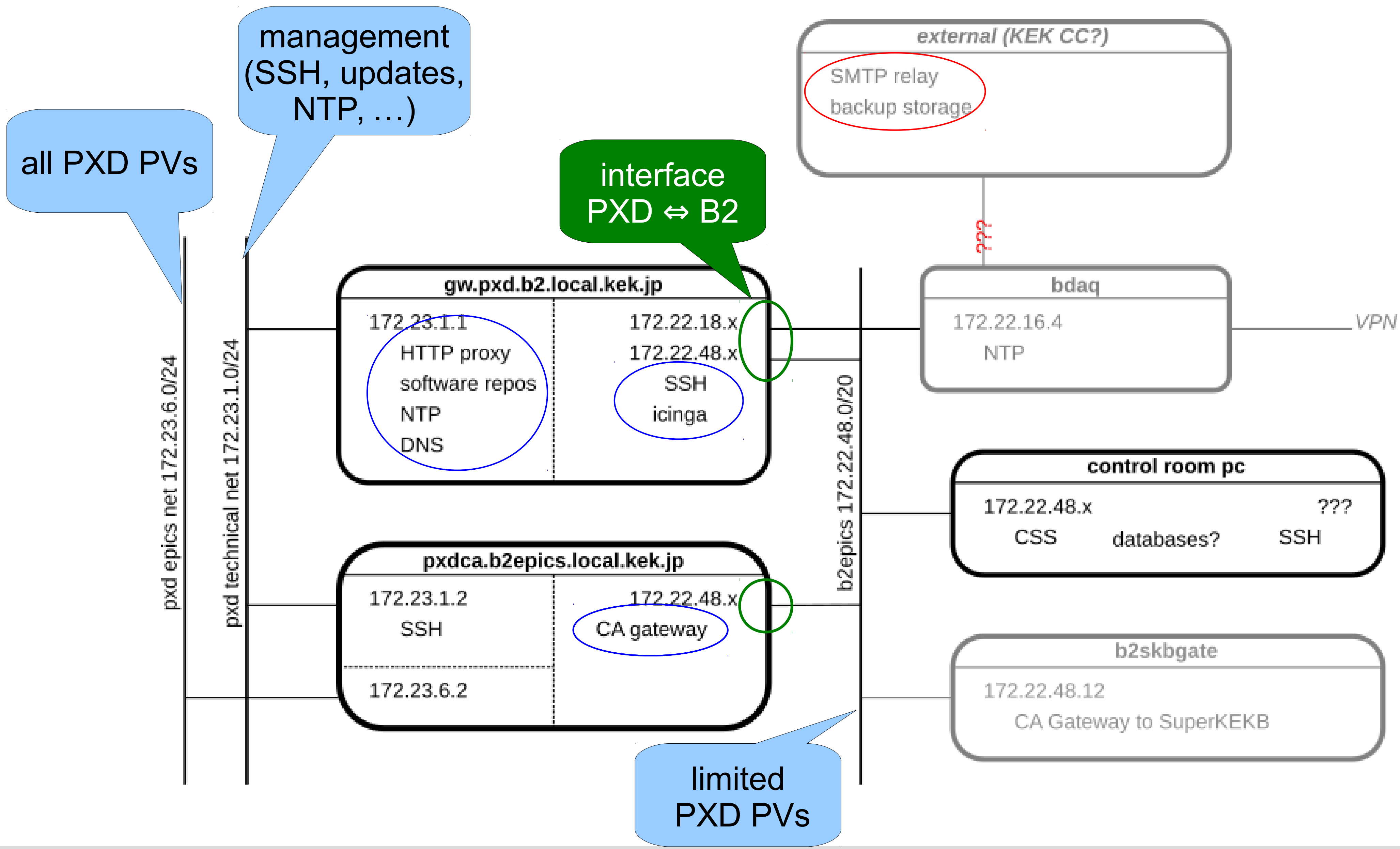
- Log messages are collected in a global database.
  - The applications send the messages to an ActiveMQ server.
  - JMS2RDB puts them into a (PostgreSQL) database.
  - CSS includes a viewer that reads from the database.
- We try to include as many applications as possible (presently, the archiver appliance is excluded)
  - Eclipse-based applications (CSS, alarm server) are compatible by default.
  - For our IOCs, we have a C++ library (see „Logfile“ in the RPM repository) that implements JMS-compatible logging.
    - Also includes automatic backtrace on any crash.

TIME	TEXT	NAME	SEVERITY	CREATETIME	USER	APPLICATION-I	CLASS
2015/01/05	Internal browser is not a	logging	WARNING	2015-01-05 20:06:39.04	ritzert	CSS	org.csstudio.logging.PluginLogListener
2015/01/05	WARNING: Prevented re	logging	SEVERE	2015-01-05 16:04:41.88	ritzert	CSS	org.csstudio.logging.PluginLogListener
2015/01/05	Operation details	logging	SEVERE	2015-01-05 15:53:53.50	ritzert	CSS	org.csstudio.logging.PluginLogListener
2015/01/05	Operation details	logging	SEVERE	2015-01-05 15:53:53.50	ritzert	CSS	org.csstudio.logging.PluginLogListener
2015/01/05	SIGNAL 11 received.	static void SuS::lo	FATAL	2015-01-05 15:50:59.37	fecmess	app	logger
2015/01/05	final	int main()	WARNING	2015-01-05 15:50:59.37	fecmess	app	example
2015/01/05	info: 9	int main()	INFO	2015-01-05 15:50:59.37	fecmess	app	example
2015/01/05	9	int main()	DEBUG	2015-01-05 15:50:59.37	fecmess	app	example

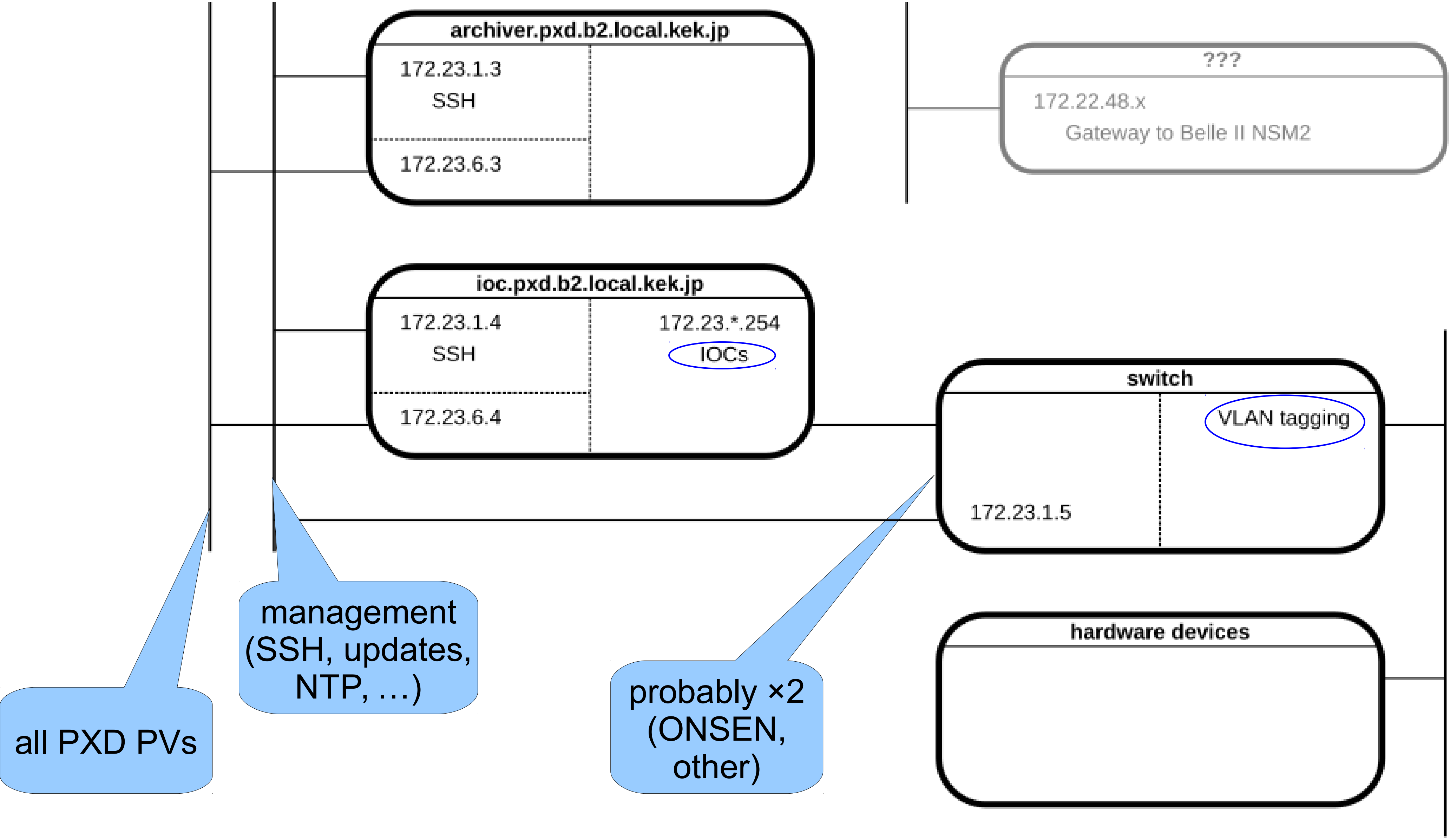
- When the SC server fails, Belle II stops.
  ⇒ redundancy is required.
  - We will use pacemaker on two identical servers for automatic switchover.

- We have many devices with small/embedded CPUs.
- When first testing DHH + PS, there was immediate interference between the systems, both using UDP.
  - This particular problem should be fixed by now, but it clearly indicates that we need to separate the systems for best reliability.
  - The data rates are low, so VLANs on top of a single 1GBit uplink will do.

Network Plan (I)



Network Plan (II)



***Note*: IP addresses are just placeholders**

Thank you!