



# **PANDA Detector Control System and Front-End Electronics: Interface Proposal**

**Tobias Triffterer**

Ruhr-Universität Bochum – Institut für Experimentalphysik I

- Questions came up during the last  $\overline{\text{PANDA}}$  CM:
  - ▶ How do we load ADC pedestals and other parameters when a run is started?
  - ▶ Where do we store the parameters that have been used?
  - ▶ How should the communication between DCS, FEE and run control work?
  - ▶ Who should be responsible for what?
- Proposal: Concept for the Interface between the  $\overline{\text{PANDA}}$  Detector Control System and the Front-End Electronics
- <https://panda-wiki.gsi.de/foswiki/pub/DCS/WebHome/dcs-fee-interface-draft-68bc99a4.pdf>

# Basic Idea

- DCS gives command “load configuration  $x$ ” via a dedicated EPICS server (Input/Output Controller  $\rightarrow$  IOC)
- DCS provides a central configuration database for DCS parameters (e. g. voltages) and DAQ/FEE parameters (e. g. ADC pedestals)
- Each configuration dataset in the database gets a unique identifier (integer, hash, etc.)
- Definition of configuration dataset: explained later
- This identifier is used to tell DAQ/FEE which dataset to use
- In addition, they also get a human-readable name

# EPICS IOC for Configuration Loading (1/2)

- Small and very incomplete EPICS crash course:
  - ▶ EPICS-based DCS consists of records
  - ▶ Records have unique name, value and further properties (alarm, archive deadbands, etc.)
  - ▶ Records can have “device support”: Glue between EPICS and hardware
  - ▶ Here: Custom device support connects to DAQ/FEE
- Start config loading process: Write unique ID to a record
- One config record per  $\overline{\text{P}}$ ANDA subdetector

## EPICS IOC for Configuration Loading (2/2)

- Connection to DAQ/FEE code with device support:
  - ▶ Method A: Shared library from DAQ/FEE used by device support code
  - ▶ Method B: Separate DAQ/FEE daemon contacted via inter-process communication or network
- Device support code forwards config dataset unique ID to DAQ/FEE code
- DAQ/FEE code uses unique ID to query all necessary data from config database
- DCS not involved in ADC programming etc.
- Distribution of responsibilities:
  - ▶ EPICS device support  $\Rightarrow$  DCS group
  - ▶ Shared library or daemon  $\Rightarrow$  DAQ/FEE group
  - ▶ API between them: Shared task, has to be agreed upon

## Feedback and Error Handling

- DAQ/FEE library/daemon shall return status code to IOC
- List of codes has to be agreed upon
- If any  $\bar{\text{PAND A}}$  subsystem not reports “successful” → abort starting of run
- No response within given time also treated as error
- Device support code sets error state on record
- ⇒ Error entered into DCS archive database
- ⇒ Experts informed via  $\bar{\text{PAND A}}$  DCS alarm handling

# Configuration Database

- Central database for all of  $\overline{\text{PANDA}}$
- Stores DCS parameters as well as DAQ/FEE data
- Storage separated into **configuration namespaces**
- One namespace per  $\overline{\text{PANDA}}$  subdetector
- ⇒ All  $\overline{\text{PANDA}}$  groups can manage their own system without harming or infringing others
- All parameter names local to each subsystem
- ⇒ Duplicates do not affect each other

# Key-value Store

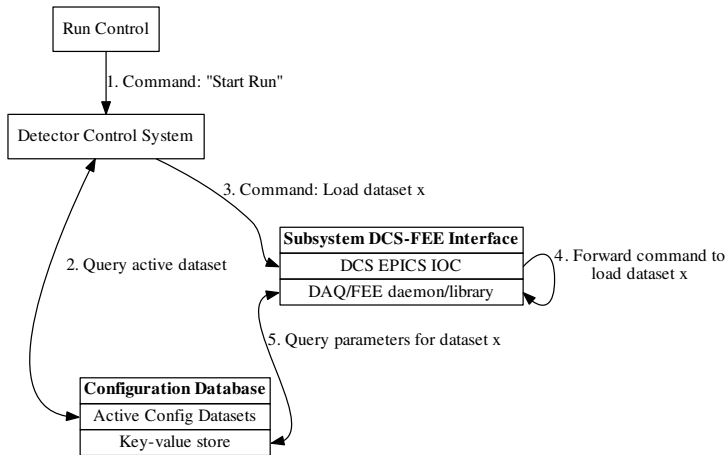
- Configuration database organized as key-value store
- Each subdetector group can freely assign keys (alias parameter names)
- Database supports standard data types for values (integer, double, string)
- Additional data types can be added on request
- Collection of key-value pairs: **configuration dataset**
- Configuration dataset must contain all required information to run the subdetector



# Configuration Dataset

- Each configuration dataset gets unique identifier
  - It also gets a human-readable name
  - All configuration datasets read-only after creation
- ⇒ Allow later reconstruction of the setup of a run
- Modification: Download dataset, modify, upload as new dataset ⇒ new unique identifier
  - Each subdetector group can designate one dataset as active per operation mode (→ explained later)
  - Run initialization procedure:
    1. DCS queries list of active datasets from config database
    2. DCS writes unique identifier of active dataset for a subsystem to EPICS config record of that subsystem

# Run Initialization Sequence



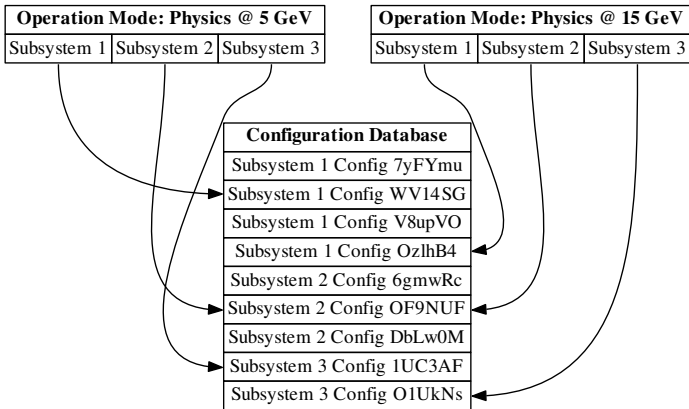
# Operation Modes

- Description of type of data-taking run
  - Global, i. e. not bound to namespace
  - There will be test runs, physics runs, and maybe more
  - Physics runs: Differentiate by beam energy
- ⇒ Adjust ADC parameters, sensor gain, etc. to expected particle energy
- ⇒ Use dynamic range of our DAQ efficiently
- Also: Run type “HESR down for time  $x$ ”
  - Automatically do some meaningful tasks (like light pulser runs for the EMC) in case of no beam
- ⇒ Calling in experts unnecessary, nevertheless maximal output from beamtime

# Linking Operation Modes and Config Datasets

- Each PANDA subsystem can make independent decision
- For each operation mode, each subsystem can designate one of their configuration datasets to be used
- One dataset can be used for several operation modes
- Old datasets are not assigned to any operation mode
- Selection can be changed at any time
- Unique identifiers of the configuration datasets used added to the metadata of a run
- Run (meta) data also read-only

# Operation Modes Schema



# Summary

- Central configuration database for all of  $\overline{\text{PANDA}}$
- Configuration datasets separated by  $\overline{\text{PANDA}}$  subsystem
- Configuration varies by operation mode
- DCS issues “load configuration” command to special IOC
- IOC forwards this to DAQ/FEE daemon/library
- Run started only if all subsystems report “successful”
  
- Questions, comments, suggestions?
  - ▶ Tell me right now...
  - ▶ Open an issue in the  $\overline{\text{PANDA}}$  GitLab  
<https://panda-repo.gsi.de/pandadcs/DcsTDR/issues/new>

Thank you for your attention!