

# Operations with Configuration management for OKS

Marco Roda

CCM - 22 January 2025

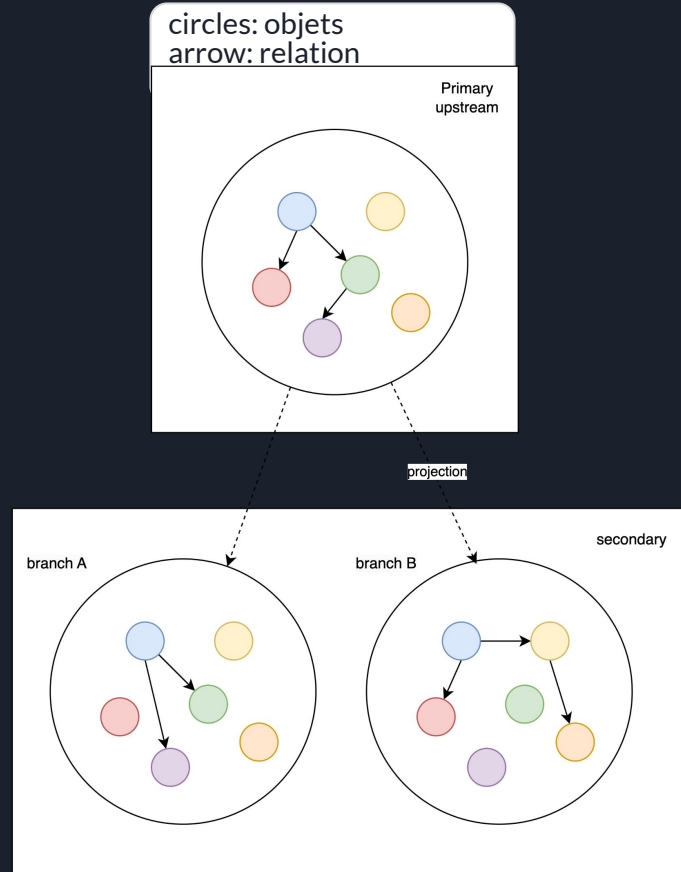


# Overview

- Last discussion was at the second day of the configuration workshop
- Since then we have an example of functionality that proves the feasibility
  - Based on NP02 - but there will be updates on this
- Today
  - quick overview and some definitions
  - Give entry points for code/documentation
  - Describe procedure for conf developers
  - Make some recommendations based on what we have learned so far

# Git model overview

- Detailed description of the concept is [here](#)
  - with comment privileges to highlight if something is not clear
  - I'll try to condense this in a few slides for you to follow the rest
- Basic assumption (yet to be fully realised)
  - We have a base repositories that stores the necessary items to create all the configurations: Configuration Object Dataset (COD)
  - all the configurations are created using the objects in the COD
    - simply linked to form the desired tree
    - the configurations live in a separate repository, called operation
  - This basic assumption is not always true, this is a problem of the schema
    - Reviews of the schema are encouraged
    - Some already happening - trigger menu





# The shifter workflow

- The configurations we store are not all the possible combinations
  - We only store all-enabled configurations
  - Shifters will have to
    - i. Retrieve the right configurations from operation
    - ii. through a dedicated interface, activate/deactivate elements
    - iii. Run with the xml files created by the interface
- The user interface is still under development, but it's a cider-like interface
  - Henry is working on this
- elements are all the things that can be activate/deactivated via the shifter interface
  - resources - or a subset of them
  - TPG (at some point)
  - ...

# Repos and interfaces

ConfPool
-Repo
-base: Remote
-operation: Remote
+constructor(dir, operation_url, base_url)
+get_cods() const
+get_daq_versions() const
+get_confs(release: regex, key: regex=.* ) const
+checkout_conf(release: string, conf: string): bool
+propagate_cod(base_ref, key: regex=.* )
+tag(base_ref, key: regex=.* )

- Base: <https://gitlab.cern.ch/dune-daq/online/np02-configs-base>
- Operation: <https://gitlab.cern.ch/dune-daq/online/np02-configs-operation>
- Scripts: <https://gitlab.cern.ch/dune-daq/online/config-management>
  
- Base and Operation are repositories that are remotes of the same git repositories
  - The difference between the two is in the branch and tag naming scheme
    - base has the same branching scheme as any DAQ repo
    - operation has a <daq-version>/<key> naming scheme for the branches and tags
  
- The scripts, config-management, has a dunedaq-like structure
  - But it lives on gitlab at the moment
  - For integration with the textual editor, I believe it should be moved on github
    - Opinions? Can this happen ASAP?
    - Note, it requires adding a python dependency: gitpython



# Why we think this will work?

- The base repository has the role of making sure that objects are always interpreted correctly
  - i.e. the objects respect the schema according to a particular version of the DAQ
- The configurations are a sort of default from which we specific configurations are added with a “small” tweaks
- We expect that thanks to the shifter operations we only support at most 30 configurations per detector
  - Far less for coldboxes
  - Of which just 2 or 3 are used daily
- This model will allow us to
  - Store all the configurations
  - allow the propagation of changes of the default using git features
  - Easily recreate configurations from the COD in case of failures due to heavy changes of the schema



# Configuration creation procedure

- Update of the COD
  - If the new configuration requires new objects first add the objects
  - The COD repo has the same tagging scheme as the other DAQ repos
    - develop
    - user\_name/<feature>
    - prep\_release/<version>
    - patch/<version>
- Once the COD is ready, new configuration(s) can be created in operation
  - For the key of the configuration, please contact me and/or Wes
  - user can develop their own configuration either on the COD repository (base) or on the operation
    - But things should be kept on separate branches:
      - One that simply adds the new object(s)
      - One that updates the links
- We will take care of propagating the changes in the COD to all the other configurations
  - We have tools for this operation



# Lessons learned

- What about NP04? And the coldboxes?
  - There are many possible ways to move forward and include everything in the same scheme
    - couple of repos (base and operation) for every detector (NP04, NP02, HD coldbox, VD coldbox)
      - Some redundancy but we will minimise the interference during operations
    - A global two repo (base and operation) for everything
      - Kind of extending the logic to the existing ehn1-config
      - Less redundancy but there will be interference between the repos
      - Will require a different naming scheme for branches to highlight the different setups
    - A single base repo with everything but separate operation repos
      - In this way the number of branches in the will be contained to each repo
  - This will be a trial and error
    - Since we noticed that with the current COD it is easy to separate the repo, we are going for
      - 1 Global base repo and separate operations
      - We might decide to separate later
- Schemas should be written in a way that high level objects DON'T have parameters, only relations
  - Ideally pushing all the parameters down to the lowest possible level, i.e. objects without relations

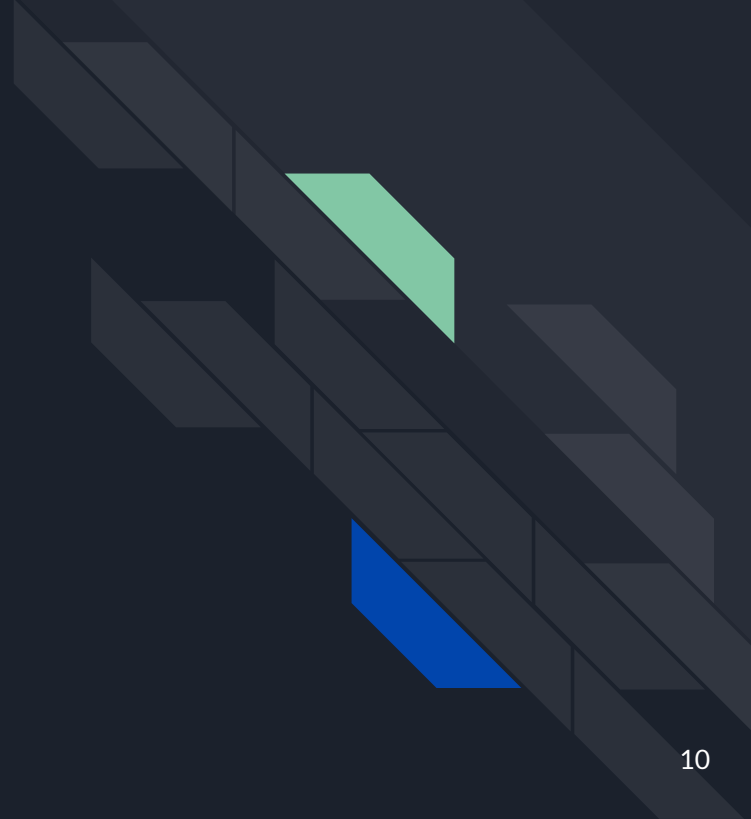




# Requests / next step

- It's time to make a decision: are we happy about what we can demonstrate?
  - What is needed to agree that this is the plan?
- Move the COD repo on github
  - In order to allow automatic operations/validations and to improve visibility
- Integration of the repo interface with textual editor
  - Mainly to highlight issues
- Keep working on the automatic features of the configuration management
  - COD propagation
  - batch tagging
- Schema updates
  - We need to move toward decoupling topology and settings in order to get the best out of this management system

Backup





# Database constraint

- The database contains about 700 objects
- We need to expose the different configurations in an easy way
- Reminder of the current system functionality
  - Structure: session -> segments -> application and other items -> ... -> small low level objects
  - When a change in the configuration happens at a low level:
    - Use a configuration object instead of another
  - all the chain up to the session level has to be duplicated if we want to select this as a session object
    - The objects must have unique ids
    - The combinatory will kill us: imagine to debug something that has tens of thousands of objects with names like “low-threshold-long-readout-window-beam-daphne-4-channel-27-conf”
  - additional objects are created by run control via the SmartDAQApps
    - Impossible to debug and to guarantee uniqueness and clarity of the names
- We need a system where the low level objects are multiple but **only some** are associated with the configuration tree
  - The others are just ready for the expert to add them in
    - The assumption is that if an object has been added, it's also tested
  - The high level objects will not be duplicated
    - sessions, segments, applications
  - This is a sort of **dropdown menu logic**
    - This is a guideline
    - It does not mean they should be disregarded lightly
    - Sometimes we need to create configuration quickly!
      - This is a way to make it possible




# The git model

- There will be 2 repositories
  - One used for development and tagged with the release
    - let's call it **config-base**, for the sake of this discussion
    - The configuration hosted here will be an "all-in" configuration that has to work
    - The configuration is the "best" default we can come up with every substem
      - Ideally a running configuration, not a calibration
      - Detector/subsystem experts should have a say in which configuration to store there
    - The role of this repo is to have a base for all the configurations that we are able to run
    - This repo is where detector experts will upload their configuration objects
      - If an object is in here it must have been tested
  - One used for storing the actual configurations so that they can be easily retrieved by shifters
    - let's call it **config-operation**
    - This is a repository aligned with config-base, config-base has to be an upstream repo
    - With branch names organised
      - main -> aligned with develop in config base
      - operation/<physical name>
      - <version>/<physical name> for past versions
      - <author>/<physical name> for expert to develop configuration
    - This is mainly managed by DAQ experts
    - The configurations here are not including the variation due to removing components
      - This is done via the shifter interface tool described by Wes in the previous talk
      - The branches here corresponds to the configurations listed in the tool
      - you can imagine easily mapping git commands onto the functions Wes described in his slides
- **Actions needs to be in place to have some level of automated validation**
  - Moving at least config-base to github seems best




# Workflow - configuration creation

- The idea here is that all configurations will be a default + few tweaks
  - This has been proved generally true in the past
- The tweak, *ideally*, is a **linking of the right objects** in the chain
  - Not the creation of a new object
- Operation
  - An expert, checks out locally the develop of config-base or the main of config-operation
    - (or the most similar branch among those already available)
  - Using DBE, the required objects are linked to the relevant objects of the chain
    - DBE here is preferred as it's type safe and well developed for batch changes
  - A branch is created with the name of the author in config-operation
  - Test
  - Creation of the final branch in operation/<physics name>



# Workflow - adding or removing configuration objects

- New objects at any level (sessions, segments, applications, conf objects, etc) should be added to the develop branch of config-base
  - Using the normal rules of the git repos
    - using prep-release and patch branches when necessary
- The addition has to be propagated to all the branches
  - As it's a simple addition the merge should be fast forward
  - Action: we need to develop scripts to do batch merges
    - Merges using regex should be supported
- Configurations that will need to rely on the new object should be updated to link to the new objects
  - At least one test branch is required
- develop can also be updated if necessary if a link is necessary
- Adding (and linking) new configuration objects is something that mostly detector expert will need to do
  - This happens often after calibration or as a result of firmware change
  - Sometime is necessary to remove objects as well in case they become deprecated
    - e.g. something not allowed anymore after a firmware update



# Workflow - release tag

- Config-base really needs to be tagged as frequently as the the code release
  - It's extremely tied to the code due to the object usage
    - Very different workflow with respect to v4 line
    - It was a big mistake that we didn't do it for v5.2.0
      - I hope a dedicate fix/discussion is happening
- config-operation can be tagged
  - With a new release
  - With changes in the connection/detector: a tag for config-operation will probably have 4 numbers
    - or 3 number and a date
  - Ideally the description of the detector would be independent from the version of the code
    - But our description is based on the schema objects so this is a necessary evil
- What we learned from NP04 is that some operations often require adaptation in the configuration code (we have O(10) patches for v4.4)
  - Whatever release we are going to use for NP02, patches driven by configuration are expected
- Following the tag of config-base, in config-operation
  - All the branches in main have to be tagged renamed with <version>/<physical name>
  - **Action: scripting for these are required**



# workflow - calibration

- Operation run by detector/subsystem expert
  - It clearly involves a loop
    - The loop logic is currently (NP04) managed by the expert
    - So it is expected for NP02
    - But in the long run we think DAQ should provide *some* support
- Operations
  - experts will start from a calibration configuration from config-operation
    - It can be created ad-hoc
    - Expert shall come forth with request to maintain this configuration on the repo
    - The configuration does not host all the different variations of the scan
      - it's just the starting point
  - Experts will maintain tools that will loop in different configuration
    - most likely these tools will call some of our scripts to update the configuration





# Workflow - calibration: comments

- When developing interfaces with detector experts this operation needs to be discussed
  - Tools to create configuration should accommodate this operation
  - Please take care of these tool
- Available tools
  - in the future we might want to consider available tools to generate configuration in multi-dimensional spaces to do these sort of operations
  - Plenty of material out there used for MC tuning as well
  - Some tools I would like to mention: [Professor](#), [Apprentice](#)
  - I think these workflows should be considered when developing tools to support calibration operation on DAQ side



# Roles of the scripts

- Scripts should be used by experts to create configuration to upload on config-base or config-operation
  - Not for shifters
  - Not for daily operations
- Notable use cases
  - Following an update in the schema an entire segment has to be recreated with all the links
    - Since there are a lot of objects a script is probably useful
    - Also useful in case of catastrophic loss of a configuration
  - Calibration
    - Used in the loop over the parameter space
    - These scripts (hopefully no more than one per subsystem) should be part of the contract with the detector experts
    - open question: where do we maintain them? My personal answer: detector specific repos
  - Adding new configuration objects following calibration
    - Mostly used by experts
    - They probably coincide with the calibration scripts
    - But they are followed up by some PRs on config-base
- Some comments
  - If an entire script had to be used to create a configuration out of config-base, the schema for the configuration would be poorly designed
  - Reminder: The creation of a schema should be a matter of linking objects



# Some comments on the model

- Expected number of branches
  - The number of combinations in the v4 database was insane: on top of the different configurations, we had to consider all the combinations of components as well
  - In this model as the shifter decides what is enabled (or rather disabled) this is not something we provide via the available configurations
    - just for the 4 APAs, we had 16 combinations
  - I don't think we will maintain more than 30 branches at anytime
    - Most likely only 5 will be used often
  - The combinatory that will still affect us is the one among multiple systems:
    - TPGThreshold x PDS thresholds
- Maybe it's time to start putting down a list?
  - I haven't, but that can be follow up item for Wes and I
  - We have some material already
- If the idea of the linking concept is well established creating a missing configuration is a matter of minutes
  - Creating a new configuration in the previous system was NOT a matter of minutes



# The Daphne example

- The daphne configuration is a very good example of all the operation we need to support:
  - It will have a number of configurations: cosmic run, calibration, data taking with laser
  - It requires calibration periodically
  - Firmware changes require code and configuration update
  - The board themselves have different hardware versions
- A bit of context
  - The application is a smart DAQ application, each one has an associated DapheConf object
  - Regardless of the number of applications and/or daphne used in the system, the DapheConf object is always the same
    - There will be one daphne object by daphne experts using scripts developed by DAQ people
  - Between DAQ and PDS consortia we have a “contract”
    - The configuration they produce is in a json format that we digest using the script
      - Essentially create a wrapper object around the json that we store in our database
- Configuration management
  - There will be a 1-to-1 relations between the available DapheConf object and the branches
    - When a new object is added, we have to create a new branch
    - When an old object becomes obsolete, we remove a branch and the object from every branch



# The daphne example - calibration

- The calibration should proceed more or less in this way
  - Experts will start from the calibration configuration, available among the branches
    - Still contains only one parameter value, but all the constant parameters will be in the configuration valid for the calibration procedure
  - They will have a loop to
    - generate the new contract json file
    - add the contract file to the calibration configuration using the available scripts
    - run with RC



# Some action items/follow ups

- <https://twiki.cern.ch/twiki/bin/view/CENF/DAQV5Operations>
  - Not linked to the main ProtoDUNE page yet
  - This should evolve to be a library of the workflows for the operations for shifters
- We have a working VNC maintained by Alec in np04
  - Very useful for experts to develop configurations
  - Please contact Alec to gain access
- Documentation
  - The only documentation on SmartDAQApplication is quite poor and oriented to low level details
  - Some of what has been discussed here should be added
  - I can volunteer but I'm not the designed of the SmartDAQApp, just a consumer. Is it enough?
- Feedback on errors
  - SmartDAQapplicaiton have a generate\_module which is written in C++ but it's called by a python
  - When everything is fine, there are no issues
  - When the C++ code throws exceptions we lose a lot of information because we only have a stack trace
    - ERS messages (if generated) are not propagated because the application itself is not on
    - But the controller is, so here there is room for improvement to make sure the ERS messages are not lost
- Feedback on enable disable mechanism
  - It's logically clear
  - But the tools to disable/enable are not ok
    - cider only works well for top level objects, for nested objects like stream it's not clear why it does not work
    - DBE seems to not allow enable/disable



# Conclusion

- I proposed a model for maintain the configurations
  - With examples from a new subsystem
  - Do we have database operations that are completely absent from this example that we expect to do?
- Missing work was **highlighted**
  - I don't think it's much work
  - One is urgent though as it's functionality it should have already been in place
- Unclear how this scales in the long term
  - I only highlighted that we should enhance support for calibration
  - Other than that I think we should use this idea for NP02 and decide after we gained experience there
- What are the next steps?
  - Clearly this is a provocation, I don't expect a full answer now
  - We probably need to digest a lot of this information