# CDB$^{(*)}$ Tutorial

Urs Langenegger

2024/10/31

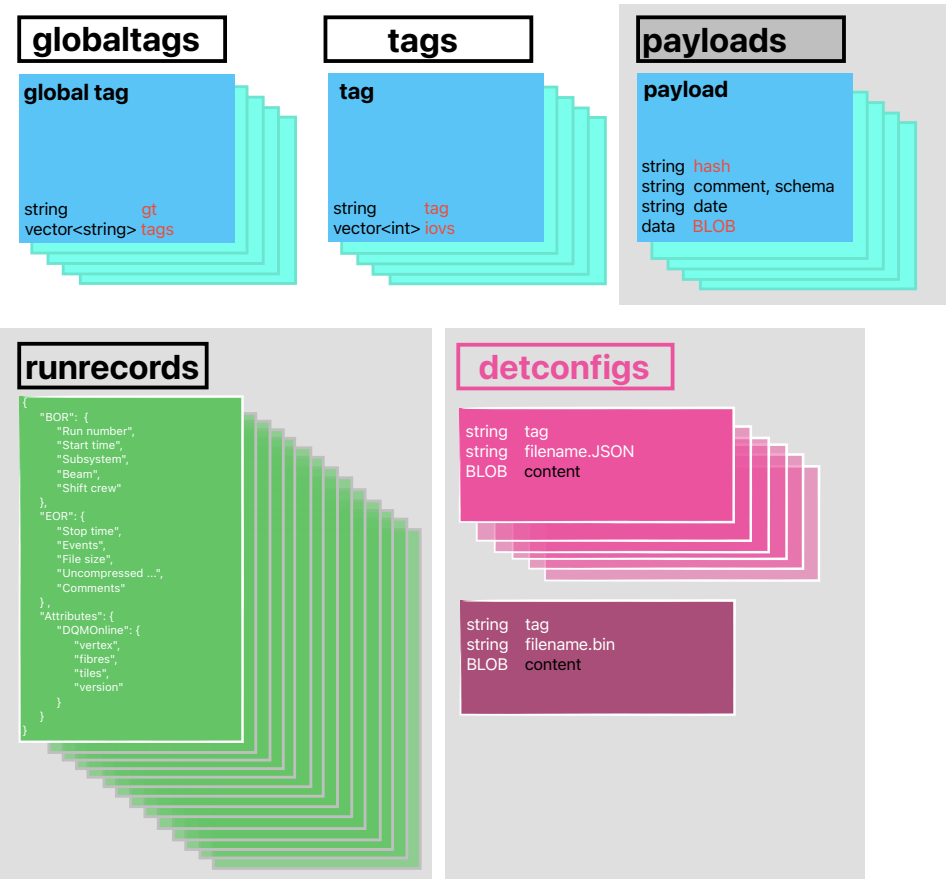
- Introduction
- Run Database
- detConfigs
- Conditions (offline)

$^{(*)}$ CDB $\neq$ Conditions DataBase

Note: http://localhost:5050/ $\approx$ http://pc11740/

# Introduction

- CDB has three "domains" (so far)
- RDB - Run Database
  - information about (all) Mu3e runs (time, duration, comments, DQM, . . . )
  - records can be inserted/updated via curl
  - web user interface (alternatively)
- detConfigs
  - can contain "anything"
    - mask/TDAC files
    - JSON dumps from ODB/MIDAS
    - . . .
  - records inserted/extracted via curl
- conditions for reco/vtx/ana (offline)
  - alignment constants, chip status and numbering, . . .
  - retrieved in offline code
- curl - "command line tool and library for transferring data with URLs"



# RDB: run record

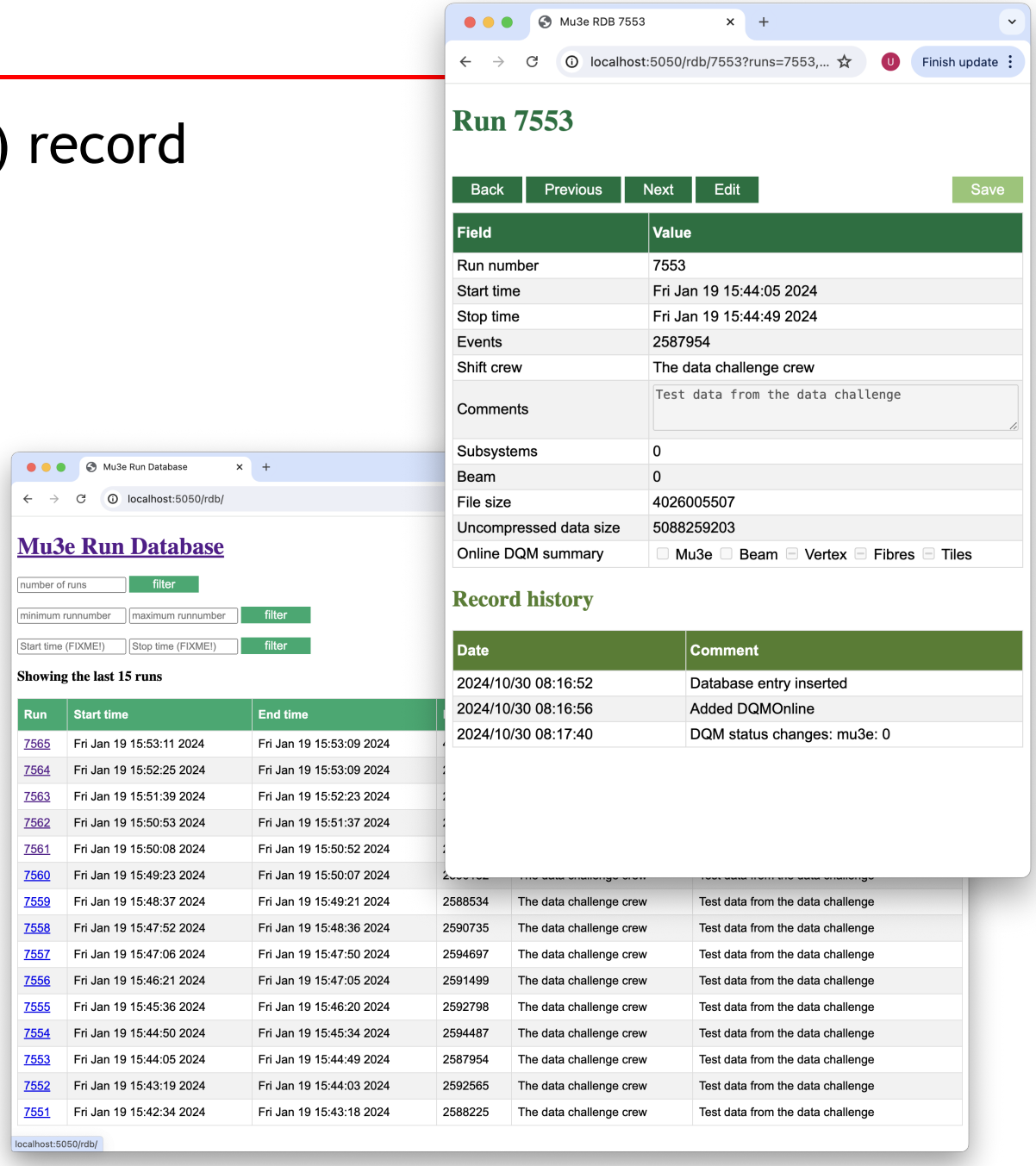
http://pc11740/rdb/ (within PSI network)

- MIDAS writes the (basic) record
  - currently:

```
moor>cat runlog_007565.json
{
  "BOR": {
    "Run number" : 7565,
    "Start time" : "Fri Jan 19 15:53:11 2024",
    "Subsystems" : 0,
    "Beam" : 0,
    "Shift crew" : "The data challenge crew"
  },
  "EOR": {
    "Stop time" : "Fri Jan 19 15:53:09 2024",
    "Events" : 4403,
    "File size" : 5840820,
    "Uncompressed data size" : 8692397,
    "Comments" : "Test data from the data challenge"
  }
}
```

  - more information should be included
    - active subsystems
    - run type: cosmics commissioning physics
    - $B$ field strength
    - . . .

⇒ automate upload



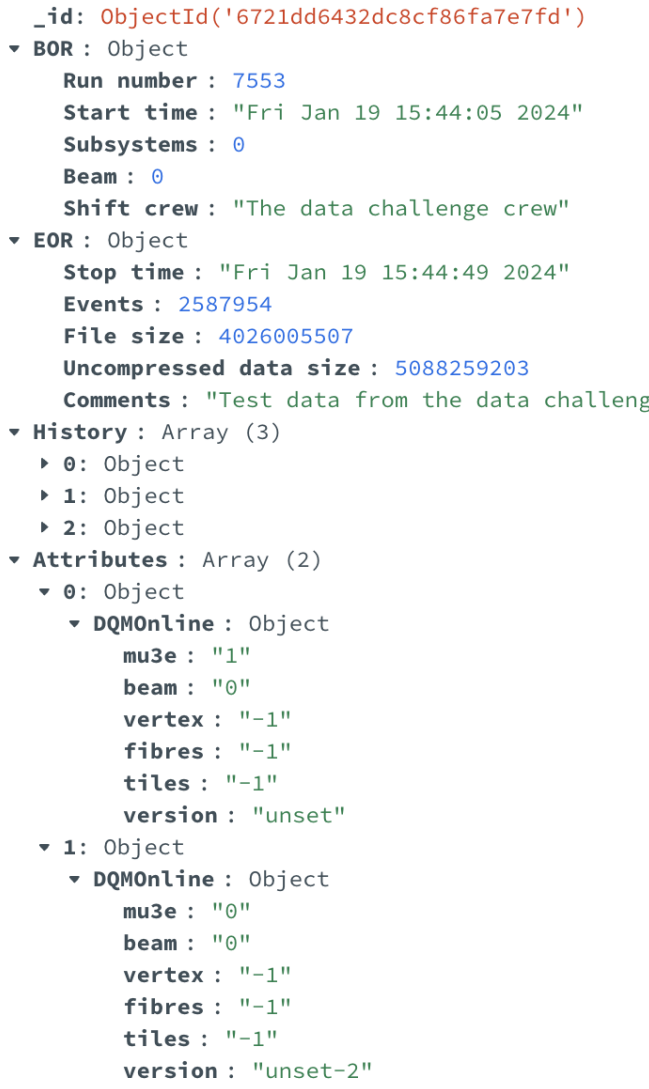

curl -X PUT -H "Content-Type: application/json" --data-binary @/Users/ursl/data/mu3e/json13/runrecords/runlog_007553.json http://pc11740/rdb/runrecords

http://pc11740/rdb/7553 (within PSI network)

# RDB: DQM information

- Data Quality Monitoring (DQM) information
  - "online" minalyzer running in hut?
    - must be fast enough (harvesting)
  - "delayed" harvester running in hut?
  - "prompt-reco" running on merlin
    - full statistics
    - delayed, should keep up with data-taking
  - pushed into "Attributes" array
- Currently:

```
moor>cat dqm10uuu.json
{
    "DQMOnline": {
        "mu3e": "1",
        "beam": "0",
        "vertex": "-1",
        "fibres": "-1",
        "tiles": "-1",
        "version": "unset"
    }
}
```



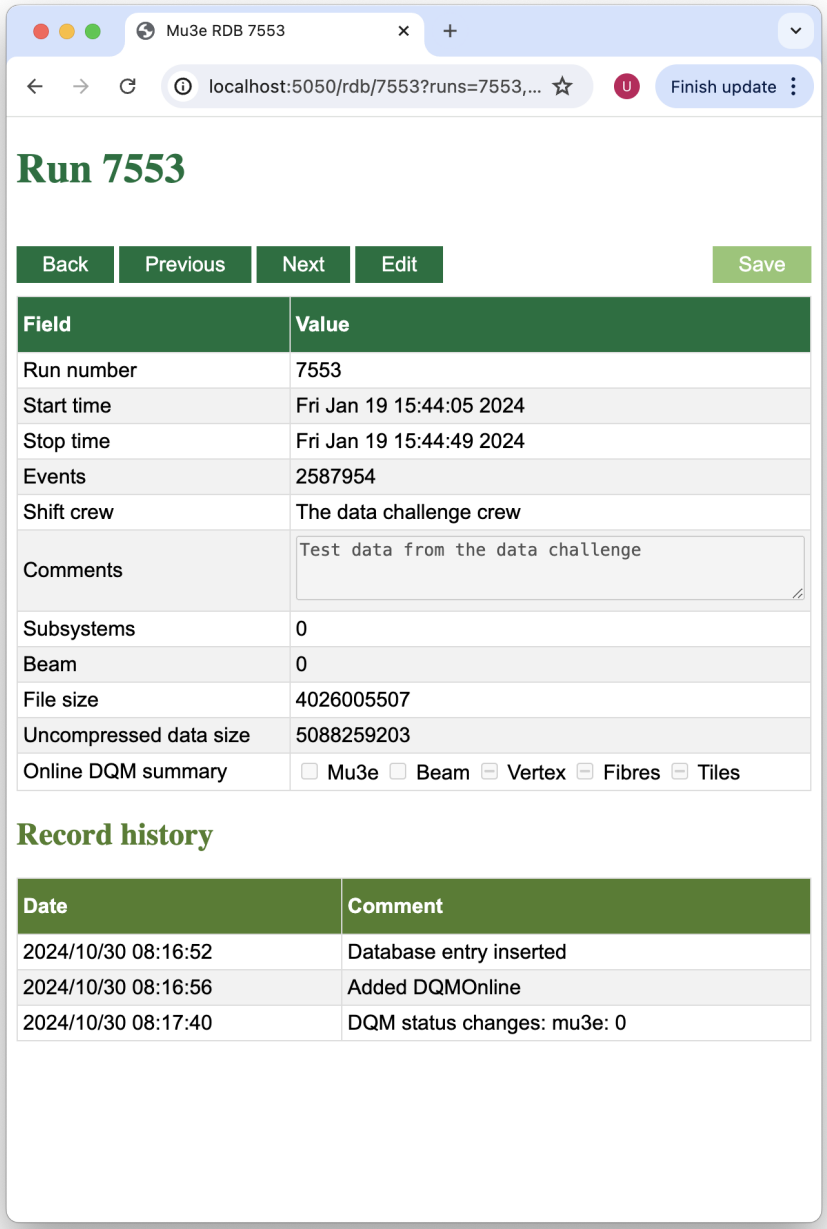

  - of course, this is a template and should be computed for each run

⇒ automate upload (after online and/or prompt reco)

curl -X PUT -H "Content-Type: application/json" --data-binary @/Users/ursl/tmp/maskfiles/dqm10uuu.json http://pc11740/rdb/addAttribute/7553

# RDB: Web user interface

- Single run display
  - ▷ shows all information available
  - ▷ allows updating of record
  - → hit "Edit" to allow updating records
  - → hit "Save" to store changes to RDB
- Note
  - ▷ you cannot add contents
    - without DQM attribute, you cannot add it
    - that should be done via curl
  - ▷ if you change a DQM attribute
    - status change will be in history
    - you could/should simultaneously edit "Comments"



# detConfigs: many (binary) files

- Possibility to obtain "versioned/keyed" config files (≡ detConfigs)
  - ▷ mask/TDAC files
  - ▷ JSON dumps from ODB/MIDAS
  - → if you want it
    not replacement of provenance tracking
- No CDB-based versioning (so far)
  - ▷ chose what you want: mask_408, mu3eqc3setup, singleChip4, . . .
- Usage examples:
  - ▷ Upload

```
moor>foreach file (mask_408_1_*.bin)
 foreach? echo $file
foreach? curl -v -F "file=@$file" -F "tag=mask_408" http://localhost:5050/cdb/upload
foreach? end
```

```
moor>curl -v -F "file=@mask_408_1_9_DS_chip6.bin" -F "file=@mask_408_1_9_DS_chip5.bin" -F "tag=mask_408bis" http://local
host:5050/cdb/uploadMany
```

  - ▷ Download

```
moor>curl -O -J "http://localhost:5050/cdb/downloadTag?tag=mask_408"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  169k    0  169k    0     0   194k      0 --:--:-- --:--:-- --:--:--  194k
moor>unzip
unzip    unzipsfx
moor>unzip mask_408.zip
Archive:  mask_408.zip
  inflating: mask_408_1_11_DS_chip4.bin
  inflating: mask_408_1_11_DS_chip5.bin
  inflating: mask_408_1_11_DS_chip6.bin
  inflating: mask_408_1_11_US_chip1.bin
  inflating: mask_408_1_11_US_chip2.bin
```

# detConfigs: (single) JSON files

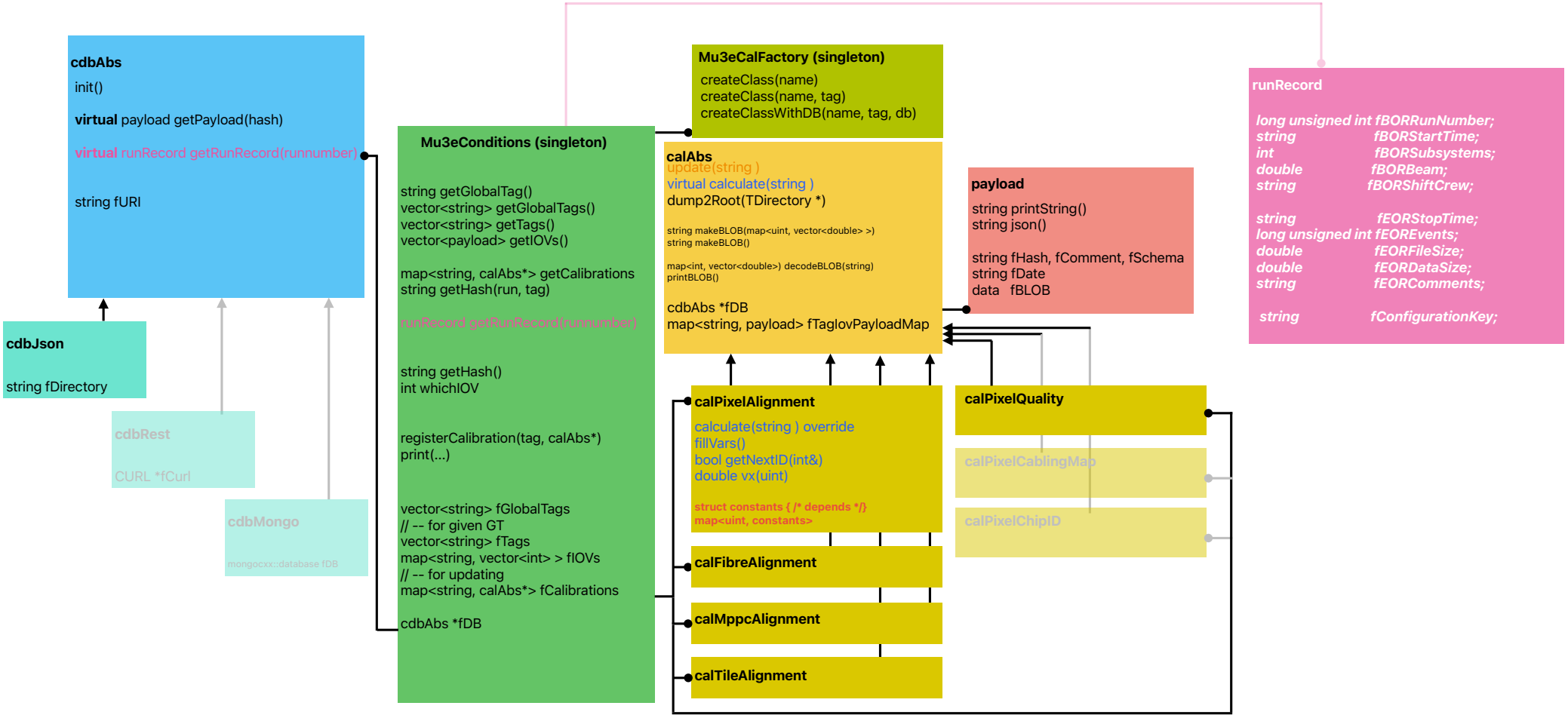
- Usage examples:

```
[moor>ls -l someODBdump.json
-rw-r--r--@ 1 ursl  staff  4077410 Oct 30 13:12 someODBdump.json
[moor>curl -X POST -F "tag=someODBdump" -F "filename=someODBdump.json" -F "file=@someODBdump.json" http://localhost:5050/cdb/uploadJSON
[File uploaded successfullymoor>
[moor>curl http://localhost:5050/cdb/downloadJSON/someODBdump -o odb.json
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 1727k  100 1727k    0     0  27.4M      0 --:--:-- --:--:-- --:--:-- 27.6M
[moor>jq --indent 3 . odb.json > odb-formatted.json
[moor>diff someODBdump.json odb-formatted.json
124420c124420
< }
\ No newline at end of file
---
> }
```

  - ▷ format issue to be solved (and also the filename/file duplicate argument)
  - ▷ uploading with same tag will not delete previous entry
    you'll get back the ''last'' uploaded version in case of multiple entries

- If reasonable use case develops, will provide "keyed" access
  - ▷ possibly with "global key"
    - containing multiple "key" for each subsystem
  - ▷ Note "key" somewhat similar to "tag" (for conditions)
  - ▷ Note difference between "tag" and "key"
    - IOVs are part of tag, but key changes for different setup conditions
  - → will do this when needed/desired

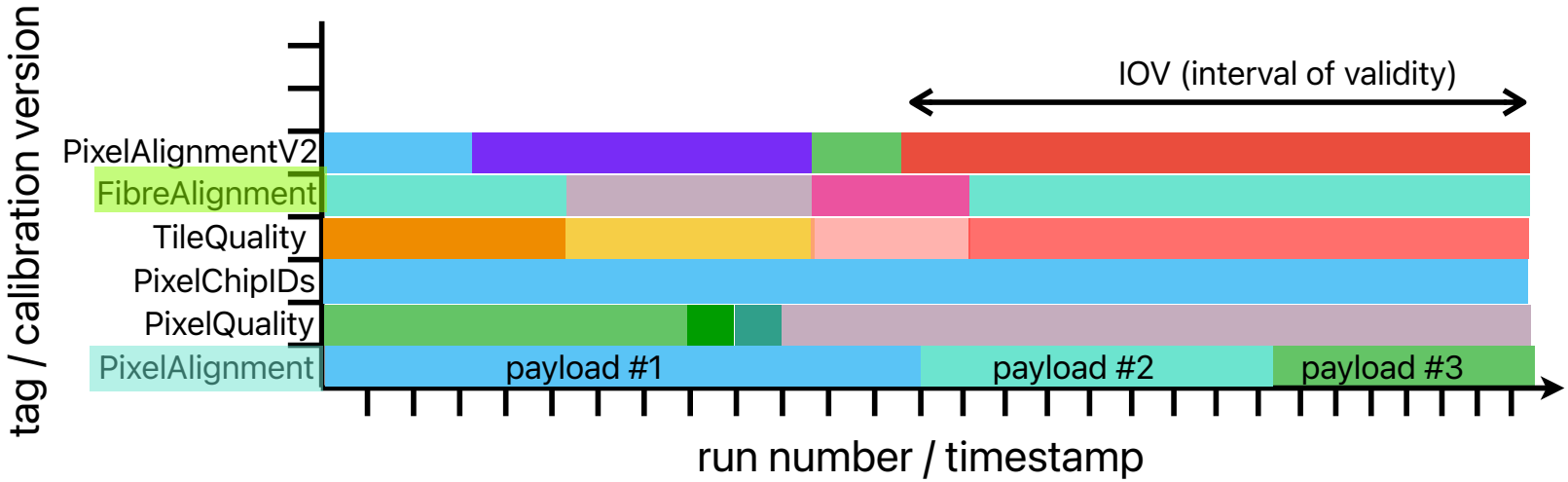
# Conditions: code structure




- All coded in terms of abstract classes
  - DB backend can be replaced w/o problems
  - Constant specifics in one (concrete) place
- One central class (Mu3eConditions) to gain conditions access

# Conditions: Tags, global tags, . . .

**Tags combine conditions/calibrations with intervals of validity**




**Global tag collects consistent set of tags**

| | | | | |
|---|---|---|---|---|
| mcidealv5.1 | pixelalignment_mcidealv5.1 | fibrealignment_mcidealv5.1 | mppcalignment_mcidealv5.0 | tilealignment_mcidealv5.1 |
| mcidealv5.0 | pixelalignment_mcidealv5.0 | fibrealignment_mcidealv5.0 | mppcalignment_mcidealv5.0 | tilealignment_mcidealv5.0 |

- Notes
  - minimal coupling between software and conditions
  - payload name contains class and tag
  - record/payload/tag naming scheme not cast in stone

# Conditions: usage

```
// get and build mu3e, including make install!
// similar for minalyzer

// simulate and sort a run

../_build/mu3eTrirec/mu3eTrirec \
    --cdb.dbconn=/Users/ursl/mu3e/software/241031-tutorial/mu3e/install/cdb/ \
   directory/mu3e_sorted_000779.root directory/wcdb-mu3e_trirec_000779.root

../_build/mu3eVertex/mu3eVertexFit \
    --cdb.dbconn=/Users/ursl/mu3e/software/241031-tutorial/mu3e/install/cdb/ \
    --input=directory/wcdb-mu3e_trirec_000779.root --output=directory/wcdb-mu3e_vertex_000779.roo

cd ../../minalyzer
_build/analyzer/minalyzer -e10 /Users/ursl/data/mu3e/run2024/run05378.mid.lz4 \
  -- offline cdb.dbconn=/Users/ursl/mu3e/software/241031-tutorial/mu3e/install/cdb/ \
  cdb.gt=mcidealv5.1
```

- Note
  - inside PSI network you could use the REST interface: --cdb.dbconn=rest
    this will connect you to pc11740
  - you could specify another server (but no other server running so far)
  - ASCII/file-based CDB is built in make install

# Conditions: contents

- Conditions CDB content
  - ▷ mu3e/mu3e> make install will create JSON and payload(!) files
    - based on mu3eUtil/cdb/ascii

```
moor>ls -r /Users/ursl/mu3e/software/241031-tutorial/mu3e/install/cdb/*
/Users/ursl/mu3e/software/241031-tutorial/mu3e/install/cdb/tags:
tilealignment_mcidealv5.1       pixelalignment_mcidealv5.1      mppcalignment_mcidealv5.0       detconfv1_mcidealv5.1
tilealignment_mcidealv5.0       pixelalignment_mcidealv5.0      fibrealignment_mcidealv5.1
pixelalignment_qc2024v1.0       mppcalignment_mcidealv5.1       fibrealignment_mcidealv5.0

/Users/ursl/mu3e/software/241031-tutorial/mu3e/install/cdb/runrecords:
runlog_004001.json

/Users/ursl/mu3e/software/241031-tutorial/mu3e/install/cdb/payloads:
tag_tilealignment_mcidealv5.1_iov_1     tag_pixelalignment_mcidealv5.0_iov_1    tag_fibrealignment_mcidealv5.0_iov_1
tag_tilealignment_mcidealv5.0_iov_1     tag_mppcalignment_mcidealv5.1_iov_1     tag_detconfv1_mcidealv5.1_iov_1
tag_pixelalignment_qc2024v1.0_iov_1     tag_mppcalignment_mcidealv5.0_iov_1
tag_pixelalignment_mcidealv5.1_iov_1    tag_fibrealignment_mcidealv5.1_iov_1

/Users/ursl/mu3e/software/241031-tutorial/mu3e/install/cdb/globaltags:
qc2024v1.0      mcidealv5.1     mcidealv5.0

/Users/ursl/mu3e/software/241031-tutorial/mu3e/install/cdb/configs:
```

- Various tools exist for creating/viewing payload files
  - ▷ mu3eUtil/cdb/test
- so far, no (user) web interface to conditions CDB

# Summary

- CDB has three domains (so far)
  - RDB - Run Database
  - detConfigs - detector configuration data
  - conditions - for offline reco/vtx/ana
- Access to CDB
  - conditions
    - ASCII files created during make install (in mu3e/mu3e or mu3e/mu3eUtil)
    - cdb.dbconn=/Users/ursl/mu3e/software/241031-tutorial/mu3e/install/cdb/
      beware of differences (--) between minalyzer and offline executables
    - cdb.dbconn=rest
      REST api works witin PSI network (VPN)
  - RDB with curl or browser http://pc11740/rdb
  - detConfigs with curl (only)
- CDB is work in progress
  - automate RDB updates in the next week(s)
  - more use cases (better summary, add DQM plots/detConfig key browsing?)
  - authentication, keys, backups, outside PSI, better server h/w, . . .
  - storage scaling issues? (detConfigs)