GT datav6.3=2025V1

- Hiked global tag (GT) version to datav6.3=2025V1
- David provided alignment rootfile with input from
  - ▷ metrology (David)
  - ▷ cosmic tracks (Mikio)

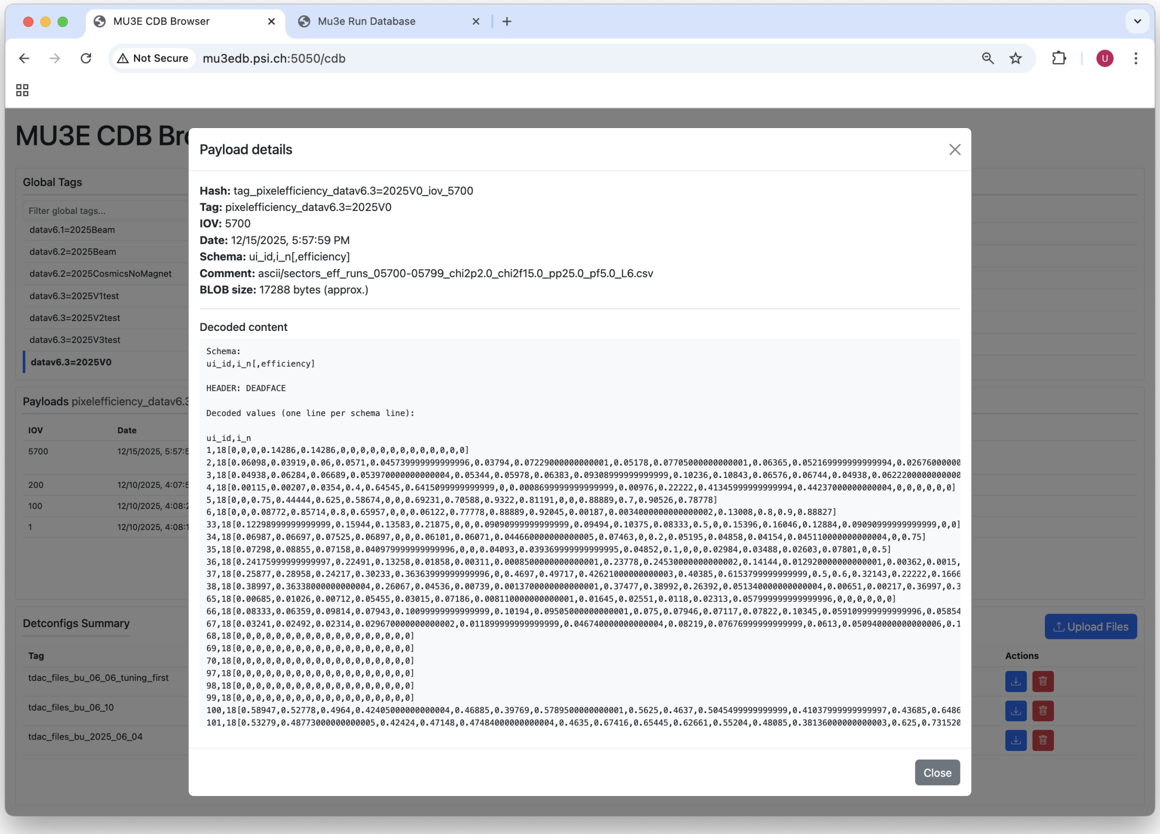
$\Rightarrow$ Contents of GT datav6.3=2025V1 now 11 tags

```
pixelalignment_datav6.3=2025V1
fibrealignment_datav6.3=2025V0
mppcalignment_datav6.3=2025V0
tilealignment_datav6.3=2025V0
pixelqualitylm_datav6.3=2025V0
detsetupv1_datav6.3=2025V0
eventstuffv1_datav6.3=2025V1
pixeltimecalibration_datav6.3=2025V0
tilequality_datav6.3=2025V0
pixelefficiency_datav6.3=2025V0
fibrequality_datav6.3=2025V0
```

- ▷ Note: eventstuffv1 not yet with proper payloads
- ▷ Note: pixelefficiency with real payload for runs > 5700 (see below)

calPixelEfficiency

- Thomas S provided CSV with percentages for runs 5700-5799
  - ▷ Mikio had something similar a bit earlier (but not identical)
- cdbWritePayload
  - ▷ translates input into $0 < \varepsilon < 1$ (if necessary)
- New payload
  - ▷ _iov_5700
- Missing?
  - ▷ efficiency errors
- Access



IOV	Date
5700	12/15/2025, 5:57:5
200	12/10/2025, 4:07:5
100	12/10/2025, 4:08:2
1	12/10/2025, 4:08:1

Tag
tdac_files_bu_06_06_tuning_first
tdac_files_bu_06_10
tdac_files_bu_2025_06_04

Payload details

Hash: tag_pixelefficiency_datav6.3=2025V0_iov_5700
Tag: pixelefficiency_datav6.3=2025V0
IOV: 5700
Date: 12/15/2025, 5:57:59 PM
Schema: ui_id,i_n[,efficiency]
Comment: ascii/sectors_eff_runs_05700-05799_chi2p2.0_chi2f15.0_pp25.0_pf5.0_L6.csv
BLOB size: 17288 bytes (approx.)

Decoded content

```
Schema:
ui_id,i_n[,efficiency]

HEADER: DEADFACE

Decoded values (one line per schema line):

ui_id,i_n
1,18[0,0,0,0.14286,0.14286,0,0,0,0,0,0,0,0,0,0,0,0,0]
2,18[0.06098,0.03919,0.06,0.0571,0.045739999999999996,0.03794,0.07229000000000001,0.05178,0.07705000000000001,0.06365,0.052169999999999994,0.02676000000
3,18[0.04938,0.06284,0.06689,0.053970000000000004,0.05344,0.05978,0.06383,0.09308999999999999,0.10236,0.10843,0.06576,0.06744,0.04938,0.0622200000000000
4,18[0.00115,0.00207,0.0354,0.4,0.64545,0.6415099999999999,0,0.0008699999999999999,0.00976,0.22222,0.41345999999999994,0.44237000000000004,0,0,0,0,0,0]
5,18[0,0,0.75,0.44444,0.625,0.58674,0,0,0.69231,0.70588,0.9322,0.81191,0,0,0.88889,0.7,0.90526,0.78778]
6,18[0,0,0.08772,0.85714,0.8,0.65957,0,0,0.06122,0.77778,0.88889,0.92045,0.00187,0.0034000000000000002,0.13008,0.8,0.9,0.88827]
33,18[0.12298999999999999,0.15944,0.13583,0.21875,0,0,0.09090999999999999,0.09494,0.10375,0.08333,0.5,0,0.15396,0.16046,0.12884,0.09090999999999999,0,0]
34,18[0.06987,0.06697,0.07525,0.06897,0,0,0.06101,0.06071,0.044660000000000005,0.07463,0,0.2,0.05195,0.04858,0.04154,0.045110000000000004,0,0.75]
35,18[0.07298,0.08855,0.07158,0.040979999999999996,0,0,0.04093,0.039369999999999995,0.04852,0.1,0,0,0.02984,0.03488,0.02603,0.07801,0,0.5]
36,18[0.24175999999999997,0.22491,0.13258,0.01858,0.00311,0.0008500000000000001,0.23778,0.24530000000000002,0.14144,0.012920000000000001,0.00362,0.0015,
37,18[0.25877,0.28958,0.24217,0.30233,0.36363999999999996,0,0.4697,0.49717,0.42621000000000003,0.40385,0.6153799999999999,0.5,0.6,0.32143,0.22222,0.1666
38,18[0.38997,0.36338000000000004,0.26067,0.04536,0.00739,0.0013700000000000001,0.37477,0.38992,0.26392,0.051340000000000004,0.00651,0.00217,0.36997,0.3
65,18[0.00685,0.01026,0.00712,0.05455,0.03015,0.07186,0.008110000000000001,0.01645,0.02551,0.0118,0.02313,0.057999999999999996,0,0,0,0,0,0]
66,18[0.08333,0.06359,0.09814,0.07943,0.10099999999999999,0.10194,0.09505000000000001,0.075,0.07946,0.07117,0.07822,0.10345,0.059109999999999996,0.05854
67,18[0.03241,0.02492,0.02314,0.029670000000000002,0.011899999999999999,0.046740000000000004,0.08219,0.07676999999999999,0.0613,0.050940000000000006,0.1
68,18[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
69,18[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
70,18[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
97,18[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
98,18[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
99,18[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
100,18[0.58947,0.52778,0.4964,0.42405000000000004,0.46885,0.39769,0.5789500000000001,0.5625,0.4637,0.5045499999999999,0.41037999999999997,0.43685,0.6486
101,18[0.53279,0.48773000000000005,0.42424,0.47148,0.47484000000000004,0.4635,0.67416,0.65445,0.62661,0.55204,0.48085,0.38136000000000003,0.625,0.731520
```

```
#include "mu3e/tools/cdb.h"
#include "mu3e/cdb/calPixelEfficiency.hh"

calPixelEfficiency *cpe = mu3e::cdbCal<calPixelEfficiency>("pixelefficiency_");
int chipID, sector;
// snip
double eff = cpe->efficiency(chipID, sector); // NOTE: YOU must know what sector is
```

calEventStuff

- Not all pixel data in a run should be analyzed
  - ▷ pixel not ready
  - ▷ de-synchronization
- calEventStuff to contain such information
  - ▷ Marius added two fields to MIDAS meta data
    - end_frame_good_pixel_data
    - start_frame_good_pixel_data
  - ▷ $\approx$ time span with 10 FEDs delivering data(?)

$\Rightarrow$ maybe other aspects will be added (Correlations?)

- Access to information

```
#include "mu3e/tools/cdb.h"
#include "mu3e/cdb/calEventStuffV1.hh"

calEventStuffV1* ces = mu3e::cdbCal<calEventStuffV1>("eventstuffv1_");
uint64_t goodStartTime(0), goodEndTime(0xFFFFFFFF);
if (ces != 0) {
  goodStartTime = ces->startFrameGoodPixelData();
  goodEndTime = ces->endFrameGoodPixelData();
}
```

- Note: Reproduce all meta data before real payloads available!

CDB access without (mu3e) C++ code

- The CDB (in essence) is very simple (GT, tags, payloads)
  - ▷ JSON, either from mongoDB/REST (http) or directly from files
  - → interface to CDB can be literally from anything/anywhere
  - ▷ GT and tags are simple JSON documents with (basically) one array
  - ▷ payloads: metadata and (base64-encoded) BLOBs (binary large objects)
- De-coding the BLOB is the only non-trivial aspect
  - ▷ "Schema" information in payload, e.g. "ui_id,i_n[,efficiency]"
  - ▷ CDB browser does it in JS
  - ▷ Python API example
  - → "cursor" can do that
- Access to CDB contents
  - ▷ raw information
    e.g. get the numbers
  - ▷ interpretation of information
    e.g. combine column information to check pixel status
  - ▷ synchronization between the two aspects
  - → this is the "tricky" part

Python API to CDB

- Setup

```
git clone git@github.com:ursl/mu3eanca
cd mu3eanca/mu3e-cdb-python/
pip install -e .
python examples/basic_usage.py
```

- Example code (examples/basic_usage.py)

```
db = JsonDatabase("/data/project/mu3e/cdb")  # This is for merlin7. Replace with your actual path
conditions = Mu3eConditions.instance("datav6.3=2025V1", db)

# Enable verbose output to see what's happening
conditions.set_verbosity(1)

# Debug: Print available tags
print(f"Global tag: {conditions.get_global_tag()}")
tags = conditions.get_tags()
print(f"Available tags ({len(tags)}): {tags[:10]}...")  # Print first 10 tags

# Set run number
conditions.set_run_number(3823)
print(f"Set run number: {conditions.get_run_number()}")

# Get pixel quality calibration
print("\nTrying to get pixelqualitylm_ calibration...")
try:
    pixel_quality = conditions.get_calibration("pixelqualitylm_")
    if pixel_quality:
        print(f"Successfully got pixel_quality calibration")
        print(f"  Hash: {pixel_quality.hash}")
        print(f"  Schema: {pixel_quality.get_schema()}")

        CHIP = 1313
        # Check pixel status
        status = pixel_quality.get_status(chip_id=CHIP, col=10, row=20)
```

  - ▷ **Very preliminary:** example, not the "proper" place, not properly validated