

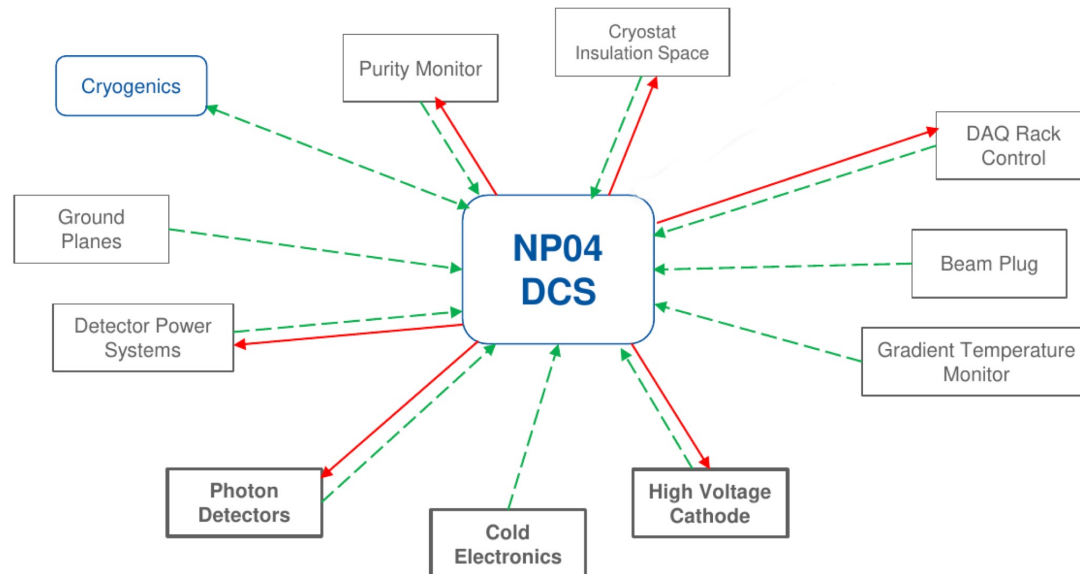
Slow Controls Conditions Data

Lino Gerlach, Paul Laycock

30.03.2022

Introduction / Reminder

- Working on protoDUNE conditions database
- Focus on data from Slow Controls (aka 'DCS')
 - E.g., LAr temp. & purity, high-voltage, ground impedance
 - Indexed by time stamp & stored in SC archive ('DCS-DB')
- Challenge: raw data written w/ very high granularity
 - Higher granularity than needed for offline processing



Current State of Slow Controls Data

- SCADA system writes raw data to DCS-DB (Oracle)
- Detector experts access that data via a website
 - <https://np04-slow-control.web.cern.ch/>
- DCS-DB is accessed via read-only rest API
 - <https://gitlab.cern.ch/ep-dt-di/dcsdb-rest>



Looking at the Database - 1

- Two tables / views are important
 - ELEMENTS: List of 29766 unique sensors
 - VEVENTSCREEN: Timestamped return values of all sensors

ELEMENTS

❖ COLUMN_NAME	❖ DATA_TYPE	❖ NULLABLE	DATA_DEFAULT	❖ COLUMN_ID	❖ COMMENTS
1 ELEMENT_ID	NUMBER(25,0)	No	(null)	1 (null)	
2 SYS_ID	NUMBER(20,0)	Yes	(null)	2 (null)	
3 EVENT	NUMBER(1,0)	No	0	3 (null)	
4 ALERT	NUMBER(1,0)	No	0	4 (null)	
5 ELEMENT_NAME	VARCHAR2(4000 BYTE)	Yes	(null)	5 (null)	
6 DPT_ID	NUMBER(20,0)	Yes	(null)	6 (null)	
7 DP_ID	NUMBER(20,0)	Yes	(null)	7 (null)	
8 DPE_ID	NUMBER(20,0)	Yes	(null)	8 (null)	
9 UNIT	VARCHAR2(4000 BYTE)	Yes	(null)	9 (null)	
10 ALIAS	VARCHAR2(4000 BYTE)	Yes	(null)	10 (null)	
11 GROUP_NAME	VARCHAR2(7 BYTE)	Yes	(null)	11 (null)	
12 COMMENT_	VARCHAR2(4000 BYTE)	Yes	(null)	12 (null)	
13 TYPE_	NUMBER(20,0)	Yes	(null)	13 (null)	

VEVENTSCREEN

❖ COLUMN_NAME	❖ DATA_TYPE	❖ NULLABLE	DATA_DEFAULT	❖ COLUMN_ID	❖ COMMENTS	❖ INSERTABLE	❖ UPDATABLE	❖ DELETABLE
1 ELEMENT_ID	NUMBER(25)	No	(null)	1 (null)	NO	NO	NO	
2 ELEMENT_NAME	VARCHAR2(4000)	Yes	(null)	2 (null)	NO	NO	NO	
3 ALIAS	VARCHAR2(4000)	Yes	(null)	3 (null)	NO	NO	NO	
4 TYPE_	NUMBER(20)	Yes	(null)	4 (null)	NO	NO	NO	
5 DPT_ID	NUMBER(20)	Yes	(null)	5 (null)	NO	NO	NO	
6 DP_ID	NUMBER(20)	Yes	(null)	6 (null)	NO	NO	NO	
7 DPE_ID	NUMBER(20)	Yes	(null)	7 (null)	NO	NO	NO	
8 SYS_ID	NUMBER(20)	Yes	(null)	8 (null)	NO	NO	NO	
9 TS	TIMESTAMP(9)	No	(null)	9 (null)	NO	NO	NO	
10 VALUE_NUMBER	BINARY_DOUBLE()	Yes	(null)	10 (null)	NO	NO	NO	
11 STATUS	NUMBER(20)	Yes	(null)	11 (null)	NO	NO	NO	
12 MANAGER	NUMBER(20)	Yes	(null)	12 (null)	NO	NO	NO	
13 USER_	NUMBER(5)	Yes	(null)	13 (null)	NO	NO	NO	
14 TEXT	VARCHAR2(4000)	Yes	(null)	14 (null)	NO	NO	NO	
15 VALUE_STRING	VARCHAR2(4000)	Yes	(null)	15 (null)	NO	NO	NO	
16 VALUE_TIMESTAMP	TIMESTAMP(9)	Yes	(null)	16 (null)	NO	NO	NO	
17 CORRVALUE_NUMBER	BINARY_DOUBLE()	Yes	(null)	17 (null)	NO	NO	NO	
18 OLVALUE_NUMBER	BINARY_DOUBLE()	Yes	(null)	18 (null)	NO	NO	NO	
19 CORRVALUE_STRING	VARCHAR2(4000)	Yes	(null)	19 (null)	NO	NO	NO	
20 OLVALUE_STRING	VARCHAR2(4000)	Yes	(null)	20 (null)	NO	NO	NO	
21 CORRVALUE_TIMESTAMP	TIMESTAMP(9)	Yes	(null)	21 (null)	NO	NO	NO	
22 OLVALUE_TIMESTAMP	TIMESTAMP(9)	Yes	(null)	22 (null)	NO	NO	NO	
23 POSITION	NUMBER(20)	Yes	(null)	23 (null)	NO	NO	NO	
24 VALUE_DYNNUMBER	BINARY_DOUBLE()	Yes	(null)	24 (null)	NO	NO	NO	
25 VALUE_DYNSTRING	VARCHAR2(4000)	Yes	(null)	25 (null)	NO	NO	NO	
26 VALUE_DYNTIMESTAMP	TIMESTAMP(9)	Yes	(null)	26 (null)	NO	NO	NO	
27 CORRVALUE_DYNNUMBER	BINARY_DOUBLE()	Yes	(null)	27 (null)	NO	NO	NO	
28 OLVALUE_DYNNUMBER	BINARY_DOUBLE()	Yes	(null)	28 (null)	NO	NO	NO	
29 CORRVALUE_DYNSTRING	VARCHAR2(4000)	Yes	(null)	29 (null)	NO	NO	NO	
30 OLVALUE_DYNSTRING	VARCHAR2(4000)	Yes	(null)	30 (null)	NO	NO	NO	
31 CORRVALUE_DYNTIMESTAMP	TIMESTAMP(9)	Yes	(null)	31 (null)	NO	NO	NO	
32 OLVALUE_DYNTIMESTAMP	TIMESTAMP(9)	Yes	(null)	32 (null)	NO	NO	NO	
33 VALTYPE	NUMBER(1)	Yes	(null)	33 (null)	NO	NO	NO	
34 OFFVALUE_NUMBER	BINARY_DOUBLE()	Yes	(null)	34 (null)	NO	NO	NO	
35 OFFVALUE_STRING	VARCHAR2(4000)	Yes	(null)	35 (null)	NO	NO	NO	
36 OFFVALUE_TIMESTAMP	TIMESTAMP(9)	Yes	(null)	36 (null)	NO	NO	NO	
37 OFFVALUE_DYNNUMBER	BINARY_DOUBLE()	Yes	(null)	37 (null)	NO	NO	NO	
38 OFFVALUE_DYNSTRING	VARCHAR2(4000)	Yes	(null)	38 (null)	NO	NO	NO	
39 OFFVALUE_DYNTIMESTAMP	TIMESTAMP(9)	Yes	(null)	39 (null)	NO	NO	NO	

Screenshots taken from
SQL Developer

Looking at the Database - 2

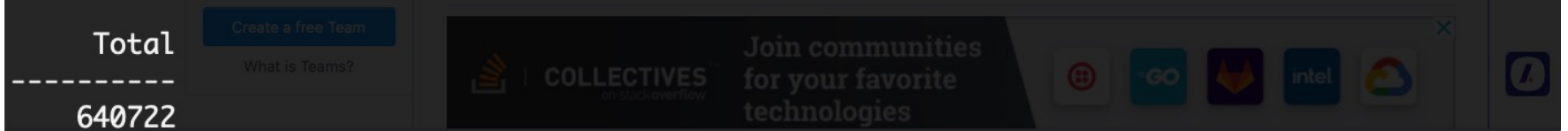
- How much data is in VEVENTSCREEN?
 - Simply counting number of all entries not feasible

```
SQL> SELECT COUNT(*) "Total" FROM NP04_DCS_01.VEVENTSCREEN;
```

- Count entries for random individual sensors

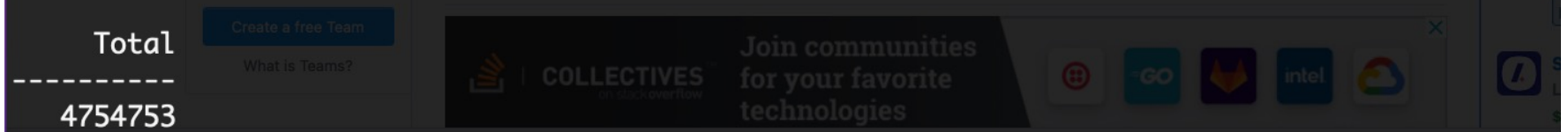
```
SQL> select COUNT(*) "Total" from NP04_DCS_01.VEVENTSCREEN where ELEMENT_ID = 47878785489690;
```

```
Total
-----
640722
```



```
SQL> select COUNT(*) "Total" from NP04_DCS_01.VEVENTSCREEN where ELEMENT_ID = 47878819044122;
```

```
Total
-----
4754753
```



- Some more random picks also showed $O(n) \sim 10^6$
 - Combined with ~ 30000 sensors $\rightarrow O(n_{tot}) \sim 10^{10}$
 - With 39 columns \rightarrow tons of data!

DCS-DB API (read-only & restful)

- Written by Roland Sipos
 - python using Flask & cx_Oracle
- Config files define groups of sensors that are read-out
 - 63 groups, 2170 individual sensors
- Set up personal CERN VM (vm-01.cern.ch)
 - Installed dependencies, got API to run
 - Can now access DCS-DB data through the API (from CERN)

```
[ligerlac@lxplus738 ~]$ curl http://vm-01.cern.ch:5000/current/beamplug  
[[{"Tue, 29 Mar 2022 13:42:26 GMT",23.705150604248047},{"Tue, 29 Mar 2022 13:42:45 GMT",22.39800453186035},{"Tue, 29 Mar 2022 13:42:27 GMT",958.7384033203125},{"Tue, 29 Mar 2022 13:41:41 GMT",7.999131679534912},{"Tue, 29 Mar 2022 13:41:41 GMT",4.557291507720947},{"Tue, 29 Mar 2022 12:51:41 GMT",22.0},{"Tue, 29 Mar 2022 13:11:41 GMT",21.100006103515625},{"Tue, 29 Mar 2022 12:43:17 GMT",998.0026245117188},{"Tue, 29 Mar 2022 12:43:17 GMT",0.4530164897441864},{"Tue, 29 Mar 2022 13:42:47 GMT",0.008425714448094368}]
```

- Not all resources work, yet
 - Some more debugging might be needed

Summary & Outlook

Summary

- Begin to understand structure & volume of raw SC data
- Set up CERN VM and got DCS-DB API to run
 - Can now access raw SC data through the API

Next Steps

- Continue debugging the API (with Roland's help)
- Talk to DRA group to better understand offline needs
- Long term: Copy necessary SC data into conditions DB

Thank you for your attention!

Questions?

Backup

DCS-DB has been copied before?

From the DUNE Core Computing Meeting (Jun 29, 2018):



There have been advances on a number of fronts. 

I have updated the DB wiki (<https://wiki.dunescience.org/wiki/Databases>) to make it more intuitive. There are now subpages for each DB, content will be added as it becomes available.

Slow Controls DB - Justin and Norm are close to having the necessary permission and access to enable writing the application that will copy the tables from the CERN Oracle DB to the FNAL PSQL DB. George has been very helpful.

Once we have the needed access (**NP04_DCS_01.VEVENTSCREEN** view) we can start developing a tool to copy the slow controls table(s) to FNAL. The application will need to run on NP04.

Dual Phase DB needs

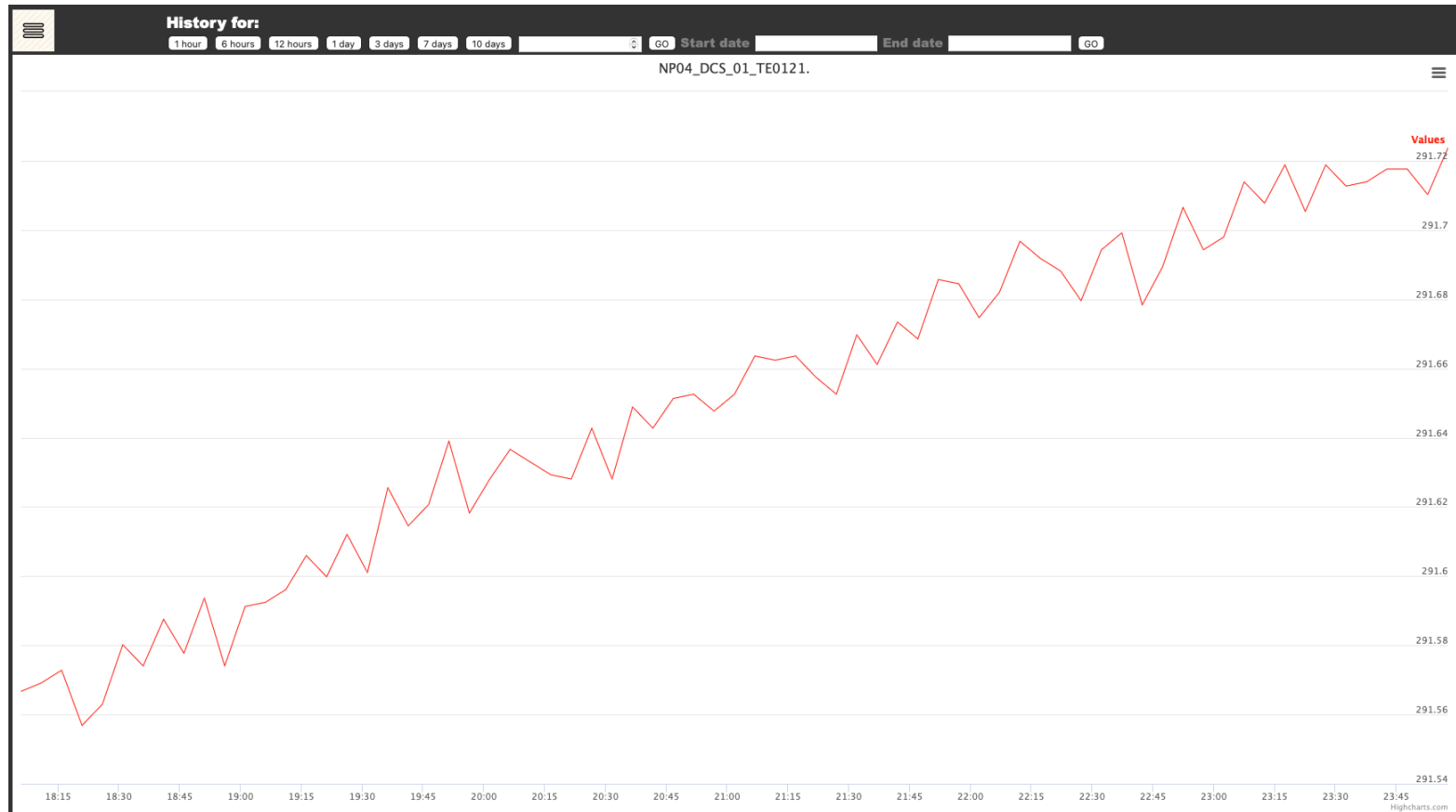
I have been in communication with Elisabetta. These are the needs for the DP group.

1) Slow control database

The slow controls DBs for SP and DP will be very similar. The DP group would like to copy their tables to the FNAL DB like is being done for the SP.

https://indico.fnal.gov/event/17494/?view=standard_inline_minutes

DCS Web App – Alternative View



Time evolution of individual sensor

Debugging the API

- Some resources try to modify sensors of 'heinzinger' category
 - Not defined in 'sensors.py' -> KeyError
- API assumes, that all sensors with 'Heinz' in their name are in group 'heinzinger'
 - Group 'heinzinger' does not exist
 - Neither in API config files, nor in actual DB

```
Traceback (most recent call last):
  File "/usr/local/lib/python3.6/site-packages/flask/app.py", line 1516, in full_dispatch_request
    rv = self.dispatch_request()
  File "/usr/local/lib/python3.6/site-packages/flask/app.py", line 1502, in dispatch_request
    return self.ensure_sync(self.view_functions[rule.endpoint])(**req.view_args)
  File "/usr/local/lib64/python3.6/site-packages/flask_restful/__init__.py", line 467, in wrapper
    resp = resource(*args, **kwargs)
  File "/usr/local/lib/python3.6/site-packages/flask/views.py", line 84, in view
    return current_app.ensure_sync(self.dispatch_request)(*args, **kwargs)
  File "/usr/local/lib64/python3.6/site-packages/flask_restful/__init__.py", line 582, in dispatch_request
    resp = meth(*args, **kwargs)
  File "/usr/local/lib/python3.6/site-packages/flask_caching/__init__.py", line 475, in decorated_function
    rv = f(*args, **kwargs)
  File "/root/dcsdb-rest/app/rest.py", line 182, in get
    if not sens.sensors['heinzinger'][s]['vals']:
KeyError: 'heinzinger'
```

It seems config files, python scripts, and DB are not synchronized (from different commits)

Some More DB Investigations

- Accessing the DB from Ixplus via 'sqlplus' command (pw needed):

```
$ sqlplus np04dbro@//pdbr-s.cern.ch:10121/pdbr.cern.ch
```

```
SQL> SELECT  
2 table_name  
3 FROM  
4 all_tables;
```

```
TABLE_NAME  
-----  
-----  
KU$_DATAPUMP_MASTER_12_2  
KU$_DATAPUMP_MASTER_12_0  
KU$_DATAPUMP_MASTER_11_2  
KU$_DATAPUMP_MASTER_11_1_0_7  
KU$_DATAPUMP_MASTER_11_1  
KU$_DATAPUMP_MASTER_10_1  
SPD_SCRATCH_TAB  
XDB$IMPORT_QN_INFO  
XDB$IMPORT_NM_INFO  
XDB$IMPORT_PT_INFO  
HELP  
...  
77 rows selected.
```

Verifying that sensors in 'ELEMENTS' are unique

```
SQL> select ELEMENT_ID, COUNT(ELEMENT_ID) from NP04_DCS_01.ELEMENTS group by ELEMENT_ID having COUNT(ELEMENT_ID)>1;  
no rows selected
```

```
SQL> select ELEMENT_NAME, COUNT(ELEMENT_NAME) from NP04_DCS_01.ELEMENTS group by ELEMENT_NAME having COUNT(ELEMENT_NAME)>1;  
no rows selected
```