Contents

1 Introduction 5
1.1 The DMS system 5
1.2 Flag states 5
1.3 Diagram block 5
1.4 Channels data are written to downstream JSON-formatted files. 6
1.4.1 DMS data providers connected to a shared memory 6
1.4.2 DMS data providers not connected to a shared memory 6
1.5 Data JSON files 7
1.6 The DMS server 7
1.7 Alarm suppression 7
1.7.1 Shelving Flags 7
1.8 Alarm filtering for notification 7
1.8.1 Persistence 7
1.8.2 Muting Flags 7
1.9 Alarm filtering for displaying 7
1.9.1 Visualization delay 7
1.9.2 Grouping Flags 8
1.10 Alert notifications 8
1.11 Snapshot JSON file 8
1.12 Visualization of Flags and associated information via a WUI 8
1.13 The DMS publisher 9
1.13.1 Archive snapshot 9
1.13.2 Archive trend 9
2 Moni processes 9
2.1 Moni process configuration file 9
2.2 Moni processes mathematical functions 10
3 ComputingDMS 10
3.1 ComputingDMS configuration file 10
3.2 Available information 11
3.3 ComputingDMS mathematical functions 11
4 Data JSON files details 12
4.1 Payload 12
5 The DMSserver in detail 13
5.1 DMSserver configuration file 14
6 Snapshot JSON payload 17
6.1 metatron_info 18
6.1.1 state_info 18

6.1.2 metadata 18
6.2 subsystem 19
6.3 active_comments 19
6.4 active_muting 19
6.5 group_flags 19
6.6 dms_info 20
6.7 active_shelving 21
6.8 providers 21
6.9 active_alerts 22
6.9.1 flags 22
6.9.2 DMS 22
6.9.3 condition_flags 22
6.9.4 group_flags 23
6.9.5 providers 23
6.10 configuration_info 23
6.11 flags 24
6.12 cond_flags_conflicts 25
6.13 Notes on the Snapshot JSON payload 25
7 The DMSpublisher in detail 26
7.1 Compression of Snapshot JSON files 26
7.2 Reading raw data and writing it to Trend 26
7.3 Compression of Trend JSON files 26
8 The Web User Interface 26
8.1 The homepage 26
8.1.1 Subsystem states 28
8.1.2 The individual-flag-information section 28
8.1.3 The Dashboard 30
8.1.4 The Alerts section 30
8.1.5 The Shelving section 31
8.1.6 The Muting section 32
8.1.7 Associated Condition Flag plots 32
8.1.8 Responsiveness 33
8.2 The DMS event monitor WUI 33
8.3 The DMS playback WUI 34
8.4 The DMS archive WUI 36
8.5 The DMS currently shelved condition-flags WUI 36
9 DMS workflow 37
9.1 Moni process configuration 37
9.2 Moni process output: JSON payload 37

9.3 DMSserver configuration 38
9.4 DMSserver output: snapshot JSON payload 38
9.5 Homepage 39
9.6 DMSpublisher: reading raw data and writing it to Trend 39
9.7 DMSpublisher: compression JSON snapshot 39
9.8 DMS event monitor WUI 40
9.9 DMS playback WUI 40
9.10 DMS archive WUI 41
10 How-to 41
10.1 How-to for the Moni processes 41
10.1.1 How to configure a generic channel in the Moni process 41
10.1.2 How to configure a channel computed with the mean() mathematical function 42
10.1.3 How to configure a channel computed with the rms() mathematical function 42
10.1.4 How to configure a channel computed with the brms() mathematical function 42
10.1.5 How to configure a channel computed with the delta() mathematical function 42
10.1.6 How to configure a channel computed with the exist() mathematical function 42
10.1.7 How to configure a channel with thresholds depending by the value of ITF_LOCK_STATE 42
10.2 How to for the ComputingDMS process 42
10.2.1 How to configure a channel computed with the ping()function 42
10.2.2 How to configure a channel computed with the cpu_user()function 43
10.2.3 How to configure a channel computed with the load()function 43
10.2.4 How to configure a channel computed with the mem_use ()function 43
10.2.5 How to configure a channel computed with the mem_swap()function 43
10.3 How-to for DMSserver 43
10.3.1 How to stop and start the DMSserver 43
10.3.2 How to configure a provider 43
10.3.3 How to configure an alert recipient 44
10.3.4 How to test an alert recipient 44
10.3.5 How to configure a flag in the DMSserver 45
10.4 How-to for the homepage 46
10.4.1 How to open the DMS homepage 46
10.4.2 How to open the individual-flag-information section 46
10.4.3 How-to for Associated Condition Flag plots 46
10.4.4 How to log into-out the system 49
10.4.5 How to open the dashboard 49
10.4.6 How to open the alerts section 49
10.4.7 How to open the shelving section 49
10.4.8 How to open the muting section 49
10.4.9 How to shelve-unshleve a condition flag 50

10.4.10 How to mute-unmute a condition flag 51
10.5 How-to for DMS event monitor 52
10.5.1 How to open the DMS event monitor 52
10.5.2 How to know the last events in a specified time window 52
10.5.3 How to know a specific events for specific condition flags a specified time window 52
10.5.4 How to know details of the event 53
10.6 How-to for DMS playback 53
10.6.1 How to open the DMS playback 53
10.6.2 How to start-pause-stop the playback 53
10.6.3 How to download the snapshot JSON payload 53
10.6.4 How to upload the snapshot JSON payload 54
10.6.5 How to switch the display 54
10.7 How-to for DMS archive 54
10.7.1 How to open the DMS archive 54
10.7.2 How to edit summary plots 54
10.7.3 How to edit custom plots 54
10.8 How to for the DMS currently shelved condition-flags. 55
10.8.1 How to open the DMS currently shelved condition-flags. 55
10.8.2 How to order the results 55
10.8.3 How to log into the system 56
10.8.4 How to view the list of the reports 56
10.8.5 How to build the report 56
10.8.6 How to un-shelve a flag 56
11 Documentation 56

1 Introduction

1.1 The DMS system

Fundamentally, the DMS should constantly cycle through the following core steps:

- draw channel data from the Data Acquisition System (DAQ);
- elaborate this data and produce relevant Flags;
- provide alerts in various formats in pre-defined conditions;
- store the history of the Flags and associated metadata, e.g. alerts sent;
- allow the user to visualize these flags and their history via a web user interface (WUI);

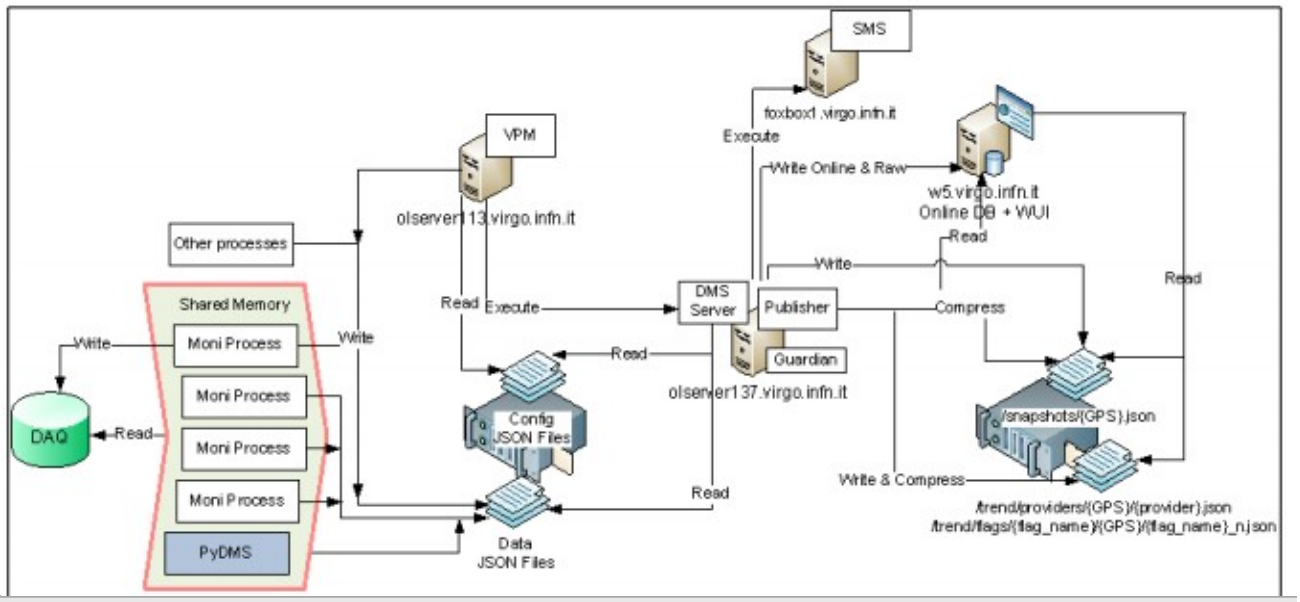


Figure 1 – The DMS system.

The information is displayed as a multicell table where each cell is named **flag** and it denotes the status of a specific item; the state of the flags reveal information relating to either a given channel or a combination of channels and related computations and expressions.

1.2 Flag states

Flags must be associated to one of the following states:

Color	*Color definition*	*State value*	*State definition*
	Green	1	Standard working condition (OK);
	Yellow	2	Non-standard working condition (Warning);
	Grey	3	Data unavailable;
	Dark-grey	4	Corrupted data
	Red	5	Non-standard working condition (Error).

1.3 Diagram block

The easiest and most common way to reduce too sensitive red flags or false alarm notifications (which may lead to the point where the operators ignore it, and thus fails to bring attention to a hazardous situation) is an appropriate set/configuration of the alarm criteria; some other techniques have been implemented to reduce the alarm activation as displayed in the following block diagram.

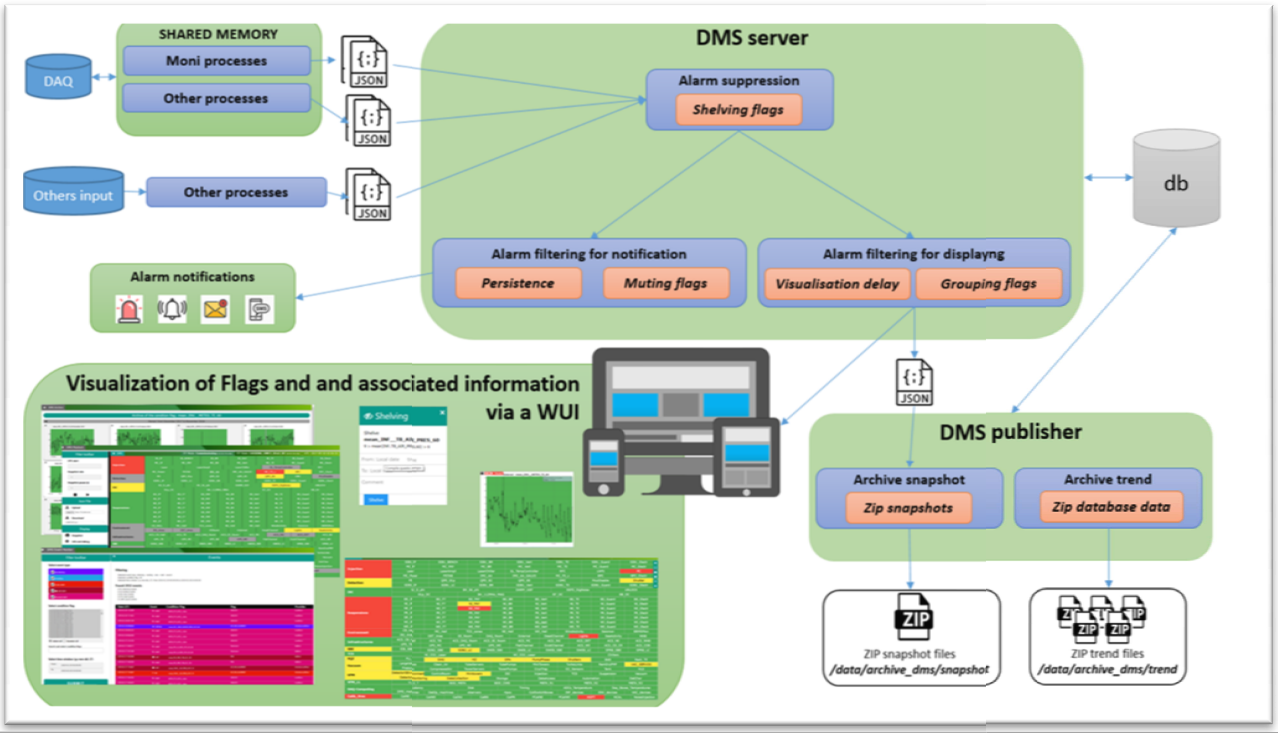


Figure 2 – Diagram block.

1.4 Channels data are written to downstream JSON-formatted files.

In this step, data are written to downstream JSON-formatted files. These data arrive via two main types of providers: they are built from the DAQ data by Moni processes connected to a common shared memory; and from so-called other providers.

1.4.1 DMS data providers connected to a shared memory

1.4.1.1 Moni processes

The Moni processes are based on the Moni library package [VIR-074A-08]. This library allows to define a set of flags whose values are determined by subflags (called "**Condition flags**") which are built by comparing to thresholds the result of computation made on a given input data channel taken from the DAQ stream. All the computed flag values are sent back to DAQ stream but also used to build the information (previously XML files, now JSON formatted files) taken by the DMS. The processing of the input channels, the thresholds, the flags dependence, etc... are defined in one configuration file per Moni process.
For more details see also Moni processes, How-to for the Moni processes.

1.4.2 DMS data providers not connected to a shared memory

1.4.2.1 ComputingDMS

This tool is used to get the status of the equipment connected on the network and build a JSON file to be read by the DMS.
The flags generated by this server are the ones displayed on the second row of the section DAQ-Computing on the DMS page.
There are two kinds of information updated every 120sec:

- ping: performed by the server by a system command;
- cpu use, memory use, etc..: elaborated by the server making a query to Ganglia;

For more details see also ComputingDMS, How to for the ComputingDMS process.

1.5 Data JSON files

Each process providing data to the DMS system must do so using a standardized format. The datapayload should be provided using JavaScript Object Notation (JSON).
For more details see also Data JSON files details.

1.6 The DMS server

The DMS server is written in Python and its start/stop is incorporated into the Virgo Process Monitoring (VPM) application. For more details see also The DMSserver in detail, How-to for DMSserver.

1.7 Alarm suppression

1.7.1 Shelving Flags

It is possible to shelve Flags for determinate lengths of time. Shelving means to hide the Flag entirely from the rest of the application. The history of the shelving of a Flag is recorded in order to available to users via the WUI. It is easily understandable to users when a Flag is in a shelved state.
For more details see also The Shelving section, How to shelve-unshleve a condition flag.

1.8 Alarm filtering for notification

1.8.1 Persistence

The DMS is able to manage checks on the persistence of an alert. In the event of an alert exceeding a pre-defined persistence time period, only at that the alert notification is sent.
For more details see also How to configure a flag in the DMSserver with email notification, How to configure a flag in the DMSserver with sms notification.

1.8.2 Muting Flags

It is possible to mute alerts for a pre-determined length of time. When an alert is muted, its Flag continues to appear in the DMS WUI and its state continues to be displayed, but any associated alert notification is not sent. The history of the muting of an alert is stored and available to the user via the WUI. It is easily understandable to users when an alert is in a muted state. For more details see also The Muting section, How to mute-unmute a condition flag.

1.9 Alarm filtering for displaying

1.9.1 Visualization delay

In order to avoid a constantly blinking WUI, DMS is able to delay the visualization of certain pre-defined Flags; the Condition Flags of which may fluctuate constantly around a given threshold. In the case of these Flags, from the moment in which one of its Condition Flags enters into a red state, the length of continuous time it passes in that state must be recorded and the Flag must only be set to red state in the event of a pre-determined time-length threshold being exceeded.
For more details see also How to configure a flag in the DMSserver with "delay_before_visualization".

1.9.2 Grouping Flags

It is possible to group specific flags into a single overall group; the state of which takes on the state, i.e. the color, of the lowest common denominator flag in the group. For example, in a group called 'flag_group', made up of 'flag_1', 'flag_2' and 'flag_3', when 'flag_1' and 'flag_3' are green, but 'flag_2' is red, the overall 'flag_group' group must be red.
For more details see also How to configure a flag in the DMSserver with "group".

1.10 Alert notifications

The DMS is able to alert pre-defined users in the event that a certain Flag is in a certain state for a pre-determined length of time. It is possible for the DMS to send alerts in the following formats:

- Sound - emitted via the WUI (when a Sound alert is emitted, the WUI must also display the flag so that it blinks for a pre-defined amount of time);
- Email - sent to pre-defined users;
- SMS - sent to pre-defined users;

The DMS is be able to send alerts when a Condition Flag remains red for a pre-determined length of time.
For more details see also The Alerts section, The DMS event monitor WUI, How to configure a flag in the DMSserver with email notification, How to configure a flag in the DMSserver with sms notification.

1.11 Snapshot JSON file

The DMS server creates a Snapshot JSON file every 10 seconds.
The Snapshot JSON files are used by the WUI to display the status of the DMS at any given point in its past.
Two types of Snaphot JSON file are stored:

- one file called **online.json**, which stores the current state values, thresholds and comments. This file is over-written each time the thread handles the Snapshot JSON loop;
- files called **{GPS_TIME}.json**, which display the values, thresholds and comments at a specific GPS time.

For more details see also Snapshot JSON payload.

1.12 Visualization of Flags and associated information via a WUI

It is possible for users to access the Flag information via a WUI. This section details the various pages of the WUI that must be available.
The Web User Interfaces (WUI) are written in HTML5, CSS7 and JavaScript8. Interaction with the under-lying DMS system is be done via PHP and AJAX.
By default, on large and medium-sized screens, the WUI automatically refresh every 10 seconds; although it is possible to pause this refresh via the WUI.
The WUI, is responsive, i.e. it is automatically adapted to the dimensions of the screen being used – smartphone, tablet, desktop, laptop – and display the information in an easy to understand and accessible manner. This clearly has implications in terms of the ways in which the application displays information in the different sections, particularly in relation to the homepage and sub-system-details pages.
Below the list of the available WUI:

- the "Homepage";
- The Flag-shelving WUI;
- The Flag-muting WUI;
- The Dashboard;
- Associated Condition-flag plots;

- The “DMS archive”;
- The “DMS event log”
- The “DMS playback”.
- The “currently shelved condition-flags”

For more details see also The Web User Interface, How-to for the homepage, How-to for DMS event monitor, How-to for DMS playback, How-to for DMS archive, How to for the DMS currently shelved condition-flags.

1.13 The DMS publisher

The DMS publisher is written in Python and its start/stop can be easily incorporated into the Virgo Process Monitoring (VPM) application.
For more details see The DMSpublisher in detail.

1.13.1 Archive snapshot

In this phase the online snapshot is compressed into a zip file building the archive snapshots. This make possible to reconstruct the status of the DMS at a given moment when a request is made via the DMS playback WUI
For more details see also Compression of Snapshot JSON files, The DMS playback WUI.

1.13.2 Archive trend

In this phase the publisher reads the data in the Raw tables of the Online database every 30 minutes – this time is configurable – copies all data and then deletes any data that is more than 30 minutes old; again this time is configurable. It then converts them into JSON payloads, which are ultimately stored as Trend JSON files, which are, in turn, compressed into ZIP archives. The Trend JSON files are used when a request is made via the WUI to reconstruct the history of a given Conditional Flag.
For more details see also Reading raw data and writing it to Trend, Compression of Trend JSON files, DMS archive.

2 Moni processes

2.1 Moni process configuration file

Its parsing is done using the Cfg library developed in Virgo. The main keywords used by the Moni process are:

- QC_JSON : followed by the path/name of the JSON formatted file which contains the Moni process results and the refresh period (in seconds) of this file.
- QC_NAME: followed by the Moni_name that is used in the prefix of the name of the output flags.
- QC_FLAG: followed by the name of the flag and a comment string to be used when the flag is on.
- QC_MONITOR: followed by a “lock_status” string, the name of the flag it is associated to, the condition to be tested and a comment string to be used when the condition is not fulfilled. In addition, a second condition and a second comment string can be added to define a warning (yellow color in DMS) instead of an alarm (red color in DMS). Each configuration line beginning with the keyword defines a “Condition flag”. All the Condition flags associated to a given flag are used to set the value of the flag.

For more details see also How-to for the Moni processes, How to configure a generic channel in the Moni process, How to configure a channel with thresholds depending by the value of ITF_LOCK_STATE.

```
CFG_PRIO 2
CFG_NOFILESAVE
CFG_PWD /virgoLog/VirgoOnline
CFG_SCHEDAFFINITY 13-31


# directory name - -1: means next file, the previous name is stored
FDIN_DIR /dev/shm/VirgoOnline/FbmAlp -1

FDIN_TAG " #SER V1:TCS* V1:ENV V1:META_ITF_LOCK"

FDOUT_CM_TIMEOUT -1

# channel prefix - channel tag to apply the prefix
#FDOUT_SET_PREFIX "V1:" "*"
FDOUT_ADD_PREFIX "V1:" "*"

# Compression type  - nbThread
FDOUT_COMPRESSION  -1  0

# Maximun server allows to connect - channel list lifetime(s)
FDOUT_CM_SERVER 2 0

FDOUT_STAT

# Destination Cm name - channel tags - 0: CmSend, >0: queue size and CmPost - retry - <checksum>
FDOUT_CM        FbmQc      "#SER Qc_TCS* Daq_TCSMoni*"       0       -1   0

/*** Name of the output json file ***/

QC_JSON /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcTCSData.json 10

/*** Name of the DAQ sms produced ***/

QC_NAME Qc_TCS

/*** NI_CO2_Laser ***/
QC_MONITOR * NI_CO2_Laser  "24<mean(ENV..TCS_NI_CO2laser_TE,10)< 24.5"    "Possible problem at NI CO2 Laser or Laser OFF"
QC_MONITOR * NI_CO2_Laser  "mean(TCS..NI_OverTempTrig,10)< 1.5"  "Possible problem at NI CO2 Laser"

/*** WI_CO2_Laser ***/
QC_MONITOR * WI_CO2_Laser  "26.7<mean(ENV..TCS_WI_CO2laser_TE,10)< 27.1"    "Possible problem at WI CO2 Laser or Laser OFF"
QC_MONITOR * WI_CO2_Laser  "mean(TCS..WI_OverTempTrig,10)< 1.5"  "Possible problem at WI CO2 Laser"
```

Figure 3 – Configuration file of TCSMoni.

2.2 Moni processes mathematical functions

The signal filtering is performed using some predefined mathematical functions that can be stated in the configuration file. The functions are the following:

- **mean(ChName,T)**: it computes the average value of the channel ChName over an interval of T seconds. For more details see also How to configure a channel computed with the mean() mathematical function.
- **rms(ChName,T)**: it computes the rms value of the channel ChName over an interval of T seconds. For more details see also How to configure a channel computed with the rms() mathematical function.
- **brms(ChName,T, Navg, f_min, f_max)**: it computes the rms value of the channel ChName over an interval of T seconds in the frequency band f_min f_max, averaging over Navg fft. For more details see also How to configure a channel computed with the brms() mathematical function.
- **dynamic(ChName, T)**: computes the difference max-min where max and min are computed over the duration indicated as second parameter (to monitor thatthe signal is moving different from zero). For more details see also How to configure a channel computed with the brms() mathematical function.
- **delta(ChName,T):**computes the max of the absolute difference between two consecutive samples. The samples taken into account are those contained into a buffer having a duration of T seconds. For more details see also How to configure a channel computed with the delta() mathematical function.
- **exist**: green flag if the channel is present (even if containing only zero) and red flag if absent. For more details see also How to configure a channel computed with the exist() mathematical function.

3 ComputingDMS

3.1 ComputingDMS configuration file

Its parsing is done using the Cfg library developed in Virgo. The main keywords used by the ComputingDMS process are:

- JSON : followed by the path/name of the JSON formatted file which contains the Moni process results and the refresh period (in seconds) of this file.
- RR: refresh rate of the process
- CF: it is a line to be considered as valid input; then every field separated by a blank space is:
    - Flag name in the DMS;
    - DNS name or IP address;
    - ganglia group name;
    - info to monitor;
    - arithmetical function over the values of the last hour;
    - low threshold for yellow flag (the flag is yellow if the value is higher);
    - low threshold for red flag (the flag is red if value is higher);

For more details see also [How to for the ComputingDMS process](#).

```
# Main priority
CFG_PRIO 1

CFG_DEBUG 0

FD_NO_CONFIG_CHECK # allow custom keywords

# No commit into file
CFG_NOFILESAVE

CFG_CMDOMAIN Cascina

# Current logfile path <path>/<cmName>
CFG_PWD /virgoLog/VirgoOnline

# Name and path for the output json file
JSON /opt/MonitoringWeb/buffer_dms/json_qcmoni/ComputingDMS.json

# Refresh rate in [s] of the server
RR 120

# FLAG olserver38 ----------------------------------------------------------------------------------------------------
CF    ping(olserver38)              olserver38       olserver38.virgo.infn.it        None              ping              None        0           0
CF    cpu_user(olserver38)          olserver38       olserver38.virgo.infn.it        New_Olservers     cpu_user          max         70          90
# --------------------------------------------------------------------------------------------------------------------
# FLAG olserver53 ----------------------------------------------------------------------------------------------------
CF    ping(olserver53)              olserver53       olserver53.virgo.infn.it        None              ping              None        0           0
CF    cpu_user(olserver53)          olserver53       olserver53.virgo.infn.it        On-LineNodes2     cpu_user          max         70          90
# --------------------------------------------------------------------------------------------------------------------
# FLAG olserver112 ---------------------------------------------------------------------------------------------------
CF    ping(olserver112)             olserver112       olserver112.virgo.infn.it       None              ping              None        0           0
CF    cpu_user(olserver112)         olserver112       olserver112.virgo.infn.it       New_Olservers     cpu_user          max         70          90
# --------------------------------------------------------------------------------------------------------------------
# FLAG olserver113 ---------------------------------------------------------------------------------------------------
CF    ping(olserver113)             olserver113       olserver113.virgo.infn.it       None              ping              None        0           0
CF    cpu_user(olserver113)         olserver113       olserver113.virgo.infn.it       New_Olservers     cpu_user          max         70          90
# --------------------------------------------------------------------------------------------------------------------
# FLAG olserver117 ---------------------------------------------------------------------------------------------------
CF    ping(olserver117)             olserver117       olserver117.virgo.infn.it       None              ping              None        0           0
CF    cpu_user(olserver117)         olserver117       olserver117.virgo.infn.it       New_Olservers     cpu_user          max         70          90
# --------------------------------------------------------------------------------------------------------------------
# FLAG olserver118 ---------------------------------------------------------------------------------------------------
CF    ping(olserver118)             olserver118       olserver118.virgo.infn.it       None              ping              None        0           0
CF    cpu_user(olserver118)         olserver118       olserver118.virgo.infn.it       New_Olservers     cpu_user          max         70          90
# --------------------------------------------------------------------------------------------------------------------
# FLAG olserver120 ---------------------------------------------------------------------------------------------------
CF    ping(olserver120)             olserver120       olserver120.virgo.infn.it       None              ping              None        0           0
```

Figure 4 –Configuration file of ComputingDMS.

3.2 Available information

The information about are the following:

- PING; for more details see also [How to configure a channel computed with the ping()function](#).
- LOAD: load_1-min [number] , load_cpus [number], load_procs [number]; for more details see also [How to configure a channel computed with the load()function](#)
- MEMORY: mem_use [Bytes], mem_share [Bytes], mem_cache [Bytes], mem_buffer [Bytes], mem_free [Bytes], mem_swap [Bytes], mem_total [Bytes]; for more details see also [How to configure a channel computed with the mem_use ()function](#), [How to configure a channel computed with the mem_swap()function](#)
- CPU: cpu_user, cpu_nice [%], cpu_system [%], cpu_wait [%], cpu_steal [%], cpu_sintr [%], cpu_idle [%]; for more details see also [How to configure a channel computed with the cpu_user()function](#).
- NETWORK: net_in [Bytes], net_out [Bytes]; for more details see also

3.3 ComputingDMS mathematical functions

The signal filtering is performed using some predefined mathematical functions that can be stated in the configuration file. The functions are the following:

- none: no operations; mean: compute the mean in the last hour of samples;
- max: compute the max in the last hour of samples; min: compute the max in the last hour of samples;

4 Data JSON files details

Each process providing data to the DMS system must do so using a standardized format. The data payload should be provided using JavaScript Object Notation (JSON). These data JSON files are defined in this section.

4.1 Payload


```
{
   "frame_latency": 2.981117,
   "frame_gps": 1235398640,
   "frame_number": 6725771,
   "frame_time": "Thu Feb 28 14:17:02 2019",
   "provider": "TCSMoni",
   "flags" : {
      "NI_CO2_Laser": {
         "flag_state": 1,
         "flag_dq_value":0,
         "condition_flags" : {
            "NI_CO2_Laser_00": {
               "channel_name": "ENV..TCS_NI_CO2laser_TE",
               "condition_flag_thresholds": "24.5 > mean(ENV..TCS_NI_CO2laser_TE,10) > 24",
               "condition_flag_comment": "Possible_problem_at_NI_CO2_Laser_or_Laser_OFF",
               "condition_flag_computed_value": 24.2188,
               "condition_flag_state": 1,
               "condition_flag_dq_value": 0
            },
            "NI_CO2_Laser_01": {
               "channel_name": "TCS..NI_OverTempTrig",
               "condition_flag_thresholds": "mean(TCS..NI_OverTempTrig,10) < 1.5",
               "condition_flag_comment": "Possible_problem_at_NI_CO2_Laser",
               "condition_flag_computed_value": 1,
               "condition_flag_state": 1,
               "condition_flag_dq_value": 0
            }
         }
      },
      "WI_CO2_Laser": {
         "flag_state": 1,
         "flag_dq_value":0,
         "condition_flags" : {
            "WI_CO2_Laser_00": {
               "channel_name": "ENV..TCS_WI_CO2laser_TE",
               "condition_flag_thresholds": "27.1 > mean(ENV..TCS_WI_CO2laser_TE,10) > 26.7",
               "condition_flag_comment": "Possible_problem_at_WI_CO2_Laser_or_Laser_OFF",
               "condition_flag_computed_value": 26.8411,
               "condition_flag_state": 1,
               "condition_flag_dq_value": 0
            },
            "WI_CO2_Laser_01": {
               "channel_name": "TCS..WI_OverTempTrig",
               "condition_flag_thresholds": "mean(TCS..WI_OverTempTrig,10) < 1.5",
               "condition_flag_comment": "Possible_problem_at_WI_CO2_Laser",
               "condition_flag_computed_value": 1,
               "condition_flag_state": 1,
               "condition_flag_dq_value": 0
            }
         }
      },
      "Chillers": {
         "flag_state": 3,
         "flag_dq_value":-1,
         "condition_flags" : {
            "Chillers_00": {
               "channel_name": "TCSR..TE_NI_Chiller",
               "condition_flag_thresholds": "mean(TCSR..TE_NI_Chiller,10) < 25",
               "condition_flag_comment": "Possible_problem_at_NI_CO2_Chiller",
```


Figure 5 – QcTCSData.json, JSON file generated by the Moni process TCSMoni.

The branches of the above payload are explained here

Name	Value	Details	Description
frame_latency	number	float, 2 decimal places	The frame latency value received from theDAQ.
frame_gps	number	integer	The frame GPS time received from theDAQ.
frame_number	number	integer	The frame number received from theDAQ.
frame_time	string	UTC	The UTC date/time received from

			theDAQ.
provider	string		The name of the provider - Moni script orother - that has written the data file.
flags	object	dictionary	Contains a dictionary of flag data.

Flag names are used as the root index of the flags dictionary. To each flag-name index is associated another dictionary containing the following leaves:

Name	Value	Details	Description
flag_state	number	integer	The DMS flag state
flag_dq_value	number	integer	The DMS flag value for use in data-quality processing
condition_flags	object	dictionary	Contains a dictionary of data related to a condition applied to a flag.

The index of the condition_flags dictionary is constituted of the related flag name, an underscore and an incrementing integer value, beginning at zero. For example:
{flag_name}_0
{flag_name}_1
{flag_name}_2
Each of these indices is associated to an associated dictionary, which contains the following elements:

Name	Value	Details	Description
channel_name	string	-	The name of the DAQ channel used in the condition.
condition_flag_thresholds	string	-	The thresholds against which the channel data has been applied in order to arrive at the condition-flag state.
condition_flag_comment	string	-	An underscore-delimited comment for use in describing why a condition-flag is in a non-standard state.
condition_flag_computed_value	number	float	The computed value that is compared to thresholds to set the condition-flag.
condition_flag_state	number	integer	The state of the condition-flag in DMS terms.
condition_flag_dq_value	number	integer	The value of the condition-flag for use in data-quality processing

5 The DMSserver in detail

Once started through the VPM, the DMS Server undertakes the following actions:

- reads the latest Data JSON files;
- checks whether the Provider is providing data;
- elaborates the data and writes it to the Online Database;
- removes old data from the Online table;
- handles flag state;
- handles alerts;
- and, when required, creates Snapshot JSON files.

The handling of flag states and alerts is not necessarily done in the order defined above, but is inter-changeable dependent upon the data provided.
Each of these actions are detailed in this section:

- **Reading latest Data JSON files**

At the start of the process, following the configuration phase, the DMS Server launches threads for

each Provider. Each of these, concurrently, reads the contents of each of the Data JSON files and

undertakes the following actions:

- uses Python native JSON library to directly read the contents of each file into a Python dictionary.

- **Checking the provision of Provider data**
  Following the reading and initial actions, each DMS-Server thread checks whether the Provider is correctly providing data. To do this, the thread gets the GPS time provided in the Data JSON files and compares it with the actual GPS time. If the difference between the two times is greater than that specified by the delay_before_unavailable_s variable, the Provider is considered to be inactive, the sub-system dedicated to the Provider is displayed in the Data Unavailable state and the alert-handling process is triggered. Otherwise, the Provider is considered to be active and the elaboration of the data continues.
- **Elaborating and writing the online data**
  Once certain that the Provider is not unavailable and up-to-date data is being provided, each DMS-Server thread undertakes the following actions:
  - checks whether each provider exists already in the database; where it does not exist, it inserts it, where it does, it updates the last-seen time;
  - checks whether each flag exists already in the database; where it does not exist, it inserts it, where it does, it updates the last-seen time;
  - checks whether each conditional flag exists already in the database; where it does not exist, it inserts it, where it does, it updates the last-seen time;
  - inserts the new data to the Online table and Raw table;
- **Removing the non-online data**
  - The next step undertaken by each DMS-Server thread is to delete data associated to non-current GPS times from the Online table.
- **Handling flag state**

Each thread determines the state of each flag, taking into consideration any state_time_delayassociated to a specific flag in the tb_time_delay table; applying the following: 0 = delay has been reset; 1 = error state in progress; 2 = completed successfully

- **Handling alerts**
  When handling alerts, each thread first undertakes the following actions:
  - checks if the state of a provider or condition flag is in error;
  - checks if an alert has been configured for this provider or condition flag;
  - checks that the flag has not been shelved;
  - checks that the flag has not been muted.

  If all of these checks are not met, the alert-handling process for the provider or condition flag is abandoned. Instead, if they are all met, the following actions are undertaken:
  - checks the persistence - the alert_recipient_mail_persistance, alert_recipient_sound and alert_recipient_sms_persistance - values, defined within the configuration, for each individual provider or conditional flag;
  - if the provider or conditional flag remains in error state beyond the persistence times, an associated alert is generated.
- **Creating Snapshot JSON files**

The Snapshot JSON files are used by the WUI to display the status of the DMS at any given point in its past.

For more details see also How-to for DMSserver, How to stop and start the DMSserver.

5.1 DMSserver configuration file

The DMS Server is started via the VPM. It provides it with a configuration file, which is read by the VPM at the moment in which the Server is activated.

The header

Certain variables are required by the VPM in a header by default.

The body
The remainder of the configuration file is dedicated to the definition of the manner in which flags are displayed and alerts are disseminated.
The information that must be supplied in this part of the configuration file is divided into three main areas:

- ***PROVIDER*** - the Moni or other processes supplying data to the DMS Server;
- ***ALERT_RECIPIENT*** - address of recipient of a specific alert;
- ***FLAG*** - the name of the flag.

The PROVIDER group includes the following fields:

Name	Type	Description
file_path	string	The location of the providing file.
provider_name	string	Alias of the provider, e.g. SuspMoni.
delay_before_unavailable_s	integer	Period of time, in seconds, before the Provider is recognized as being Unavailable.
alert_recipient_mail	list	Names of the ALERT_RECIPIENT groups that should receive an email alert.
alert_recipient_mail_persistance	integer	The length of time, in seconds, required to have passed with the flag in Error state before email alerts are sent.
alert_recipient_sms	list	Names of the ALERT_RECIPIENT groups that should receive an SMS alert.
alert_recipient_sms_persistance	integer	The length of time, in seconds, required to have passed with the flag in Error state before SMS alerts are sent
alert_recipient_sound_persistance	integer	An integer defining the number of seconds that need to pass before a default sound is played and the length of time for which the flag blinks in the WUI

For more details see also [How to configure a provider](), [How to configure an alert recipient](), [How to test an alert recipient]().

```
### ALL COLUMNS ARE WHITE-SPACE SEPARATED ###
# 0 ? keyword (upper-case string
# 1 ? file_path (string
# 2 ? provider_name (string
# 3 ? delay_before_unavailable_s (integer
# 4 ? provider_alert_mail_recipient (comma-separated list, or None
# 5 ? provider_alert_mail_persistance (integer
# 6 ? provider_alert_sms_recipient (comma-separated list, or None
# 7 ? provider_alert_sms_persistance (integer
# 8 ? provider_alert_sound_persistence (integer


PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcInfrastructuresAlarmedData.json      InfraAlarmedMoni     120   OPERATION_MAIL    600    OPERATION_SMS   7200     60
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcInjectionData.json                   InjMoni              120   OPERATION_MAIL    600    OPERATION_SMS   7200     60
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcSuspensionsShortData.json            SuspShortMoni        120   OPERATION_MAIL    600    OPERATION_SMS   7200     60
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcSuspensionsData.json                 SuspMoni             120   OPERATION_MAIL    600    OPERATION_SMS   7200     60
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcVacuumData.json                      VacuumMoni           120   OPERATION_MAIL    600    OPERATION_SMS   7200     60
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcITFOnCallData.json                   ITFOnCallMoni        120   OPERATION_MAIL    600    OPERATION_SMS   7200     60

PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcEnvironmentData.json                 EnvMoni              120   OPERATION_MAIL    600    None               0     60
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcTCSData.json                         TCSMoni              120   OPERATION_MAIL    600    None               0     60
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcVpmData.json                         VPM                  120   OPERATION_MAIL    600    None               0     60
#PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcVpmLowLatencyData.json               VPM_LL               120   OPERATION_MAIL    600    None               0     60
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcInfrastructuresData.json             InfraMoni            120   OPERATION_MAIL    600    None               0     60
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcDetectionData.json                   DetMoni              120   OPERATION_MAIL    600    None               0     60
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcSuspensionsEBData.json               SuspEBMoni           120   OPERATION_MAIL    600    None               0     60
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcLockingData.json                     LockMoni             120   OPERATION_MAIL    600    None               0     60
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcServersData.json                     ServersMoni          120   OPERATION_MAIL    600    None               0     60
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcSqzData.json                         SqzMoni              120   OPERATION_MAIL    600    None               0     60
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcCaliData.json                        CaliMoni             120   OPERATION_MAIL    600    None               0     60
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcDAQData.json                         DAQMoni              120   OPERATION_MAIL    600    None               0     60

PROVIDER  /opt/MonitoringWeb/buffer_dms/json_brmsmon/brmsmon.json                          BRMSMon              120   OPERATION_MAIL    600    None               0     60

PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcInfrastructuresSqzData.json          InfraMoniSQZ         120   None                0    None               0     60

PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/VacuumDMS.json                         VaccumDMS            120   None                0    None               0      0
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/ComputingDMS.json                      ComputingDMS         180   None                0    None               0      0
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/ITFOperationDMS.json                   ITFOperationDMS      180   None                0    None               0      0

#PROVIDER  /opt/MonitoringWeb/buffer_dms/json_dqr/dqr.json.test                                  DQR                 180   None                0    None               0      0

PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/AlertGraceDB.json                       AlertGraceDB         600   None                0    None               0      0
#-----------------------------------------------------------------------------------------#
```

Figure 7 – DMSserver configuration file, PROVIDER part.

The ALERT_RECIPIENT group includes the following fields:

Name	Type	Description

alert_recipient_group	string	An alias given to the group to distinguish it from the others.
alert_recipient_addresses	list	List of addresses (email, telephone)associated to the group.

For more details see also [How to configure an alert recipient](), [How to test an alert recipient]().

```
68  # ------------------------------ Alert recipients ------------------------------------------
69  ### ALL COLUMNS ARE WHITE-SPACE SEPARATED ###
70  # 0 ? keyword (upper-case string
71  # 1 ? alert_recipient_group (upper-case string
72  # 2 ? alert_recipient_addresses (comma-separated list
73
74  ALERT_RECIPIENT OPERATION_MAIL fabio.gherardini@ego-gw.it,francesco.berni@ego-gw.it,vincenzo.dattilo@ego-gw.it,nicola.menzione@ego-gw.it,beatrice.montanari@ego-gw.it,gianmatteo.sposito@ego-gw.it,federico.nenci@e
    gw.it,andrea.magazzu@ego-gw.it
75  ALERT_RECIPIENT SUSPENSIONS_MAIL valerio.boschi@ego-gw.it,vincenzo.dattilo@ego-gw.it,paolo.ruggi@ego-gw.it,Ettore.Majorana@roma1.infn.it
76  ALERT_RECIPIENT OPTICS_MAIL maddalena.mantovani@ego-gw.it,eric.genin@ego-gw.it,gabriel.pillant@ego-gw.it,vincenzo.dattilo@ego-gw.it,derossi@ego-gw.it,chiummo@ego-gw.it,gosselin@ego-gw.it
77  ALERT_RECIPIENT CONDITIONING_MAIL davide.soldani@ego-gw.it,hvac@ego-gw.it
78  ALERT_RECIPIENT VACUUM_MAIL antonio.pasqualetti@ego-gw.it
79  ALERT_RECIPIENT MANTOVANI_MAIL maddalena.mantovani@ego-gw.it
80  ALERT_RECIPIENT CARBOGNANI_MAIL franco.carbognani@ego-gw.it
81  ALERT_RECIPIENT SPP_MAIL spp@ego-gw.it
82  ALERT_RECIPIENT DETCHAR_MAIL nicolas.arnaud@ego-gw.it,leroy@lal.in2p3.fr
83  ALERT_RECIPIENT DETENV_MAIL nicolas.arnaud@ego-gw.it
84
85  #Majorana,    Boschi,   Ruggi
86  ALERT_RECIPIENT SUSPENSIONS_SMS 393384860816,393389429356,393493732868
87
88  ALERT_RECIPIENT CONDITIONING_SMS 393338793935,393331253553
89  ALERT_RECIPIENT CARBOGNANI_SMS 393669826503
90
91  # Richard
92  ALERT_RECIPIENT SPP_SMS 393887336448
93
94  # DET SMS -
95  ALERT_RECIPIENT DET_SMS 33672500104,33671949857,33678221611,33659041324
96
97  #Gabriel Pillant - Eric Genin - Chiummo
98  ALERT_RECIPIENT OPTICS_SMS 393429479263,393346193026,393407737044
```

Figure 8 – DMSserver configuration file, ALERT_RECIPIENT part.

The FLAG group includes the following fields:

Name	Type	Description
flag_name	string	An alias given to the flag to distinguish it from the others.
subsystem	string	The name of the sub-system to which the flag is associated.
display_row	integer	The row on which the flag appears in the associated sub-system area of the WUI.
group_flag	string	Alias of the group flag to which the flag belongs, in the event of it forming a part of a wider group.
delay_before_error_s	integer	Length of time, in seconds, that must pass before the flag is considered to be in error state.
alert_recipient_mail	list	Names of the ALERT_RECIPIENT groups that should receive an email alert.
alert_recipient_mail_persistence	integer	The length of time, in seconds, required to have passed with the flag in Error state before email alerts are sent.
alert_recipient_sms	list	Names of the ALERT_RECIPIENT groups that should receive an SMS alert.
alert_recipient_sms_persistence	integer	The length of time, in seconds, required to have passed with the flag in Error state before SMS alerts are sent.
alert_recipient_sound_persistence	integer	An integer defining the number of seconds that need to pass before a default sound is played and the length of time for which the flag blinks in the WUI.

For more details see also [How to configure a flag in the DMSserver](), [How to configure an alert recipient](), [How to test an alert recipient]().

```
### ALL COLUMNS ARE WHITE-SPACE SEPARATED ###
# 0  ? keyword (upper-case string
# 1  ? flag_name (string
# 2  ? subsystem (upper-case string
# 3  ? display_row (integer
# 4  ? group_flag (string
# 5  ? delay_before_error_s (integer
# 6  ? flag_alert_mail_recipient (comma-separated list, or None
# 7  ? flag_alert_mail_persistance (integer
# 8  ? flag_alert_sms_recipient (comma-separated list, or None
# 9  ? flag_alert_sms_persistance (integer
# 10 ? flag_alert_sound_persistence (integer



# INJECTION ----------------------------------------------------------------------------------------------------------------------
FLAG SIB1_IP              Injection    1 None         30    None                                  0    None                0        0
FLAG SIB1_BENCH           Injection    1 None         30    None                                  0    None                0        0
FLAG SIB1_BR              Injection    1 None         30    None                                  0    None                0        0
FLAG SIB1_Vert            Injection    1 None         30    None                                  0    None                0        0
FLAG SIB1_TE              Injection    1 None         30    None                                  0    None                0        0
FLAG SIB1_Guard           Injection    1 SIB1_Guard     30    None                                0    None                0        0
FLAG SIB1_Guard_Trigger   Injection    1 SIB1_Guard     30    OPERATION_MAIL,SUSPENSIONS_MAIL   1800    SUSPENSIONS_SMS  1800     1800
FLAG SIB1_Electr          Injection    1 None         30    None                                  0    None 0    0

FLAG MC_IP              Injection    2 None         30    None                                  0    None                0        0
FLAG MC_PAY             Injection    2 None         30    None                                  0    None                0        0
FLAG MC_BR              Injection    2 None         30    None                                  0    None                0        0
FLAG MC_Vert            Injection    2 None         30    None                                  0    None                0        0
FLAG MC_TE              Injection    2 None         30    None                                  0    None                0        0
FLAG MC_Guard           Injection    2 MC_Guard     30    None                                  0    None                0        0
FLAG MC_Guard_Trigger   Injection    2 MC_Guard     30    OPERATION_MAIL,SUSPENSIONS_MAIL   1800    SUSPENSIONS_SMS  1800     1800
FLAG MC_Electr          Injection    2 None         30    None                                  0    None                0        0

FLAG MasterLaser        Injection    3 Laser        30    OPERATION_MAIL,OPTICS_MAIL          300    None                0       60
FLAG SlaveLaser         Injection    3 Laser        30    OPERATION_MAIL,OPTICS_MAIL          300    None                0       60
FLAG PMC                Injection    3 Laser        30    OPERATION_MAIL,OPTICS_MAIL          600    None                0       60
FLAG LaserAmpliPower    Injection    3 LaserAmpli   30    OPERATION_MAIL,OPTICS_MAIL          600    None                0        0
FLAG LaserChiller       Injection    3 None        200    OPERATION_MAIL,OPTICS_MAIL          600    None                0        0
FLAG SL_TempController  Injection    3 None        200    OPERATION_MAIL,OPTICS_MAIL          300    None                0        0
FLAG RFC                Injection    3 None        120    None                                  0    None                0        0
FLAG LNFS               Injection    3 None        120    None                                  0    None                0        0
FLAG PC                 Injection    3 None        120    None                                  0    None                0        0

FLAG MC_Power           Injection    4 None        120    None                                  0    None                0        0
FLAG PSTAB              Injection    4 None        120    None                                  0    None                0        0
FLAG IMC_AA             Injection    4 None        120    None                                  0    None                0        0
FLAG IMC_AA_GALVO       Injection    4 None        120    None                                  0    None                0        0
FLAG MC_F0_z            Injection    4 None         30    None                                  0    None                0        0
FLAG BPC                Injection    4 None        120    None                                  0    None                0        0
FLAG BPC_Electr         Injection    4 None         30    None                                  0    None                0        0
# --------------------------------------------------------------------------------------------------------------------------------

# DETECTION ----------------------------------------------------------------------------------------------------------------------
FLAG PD                 Detection    1 None         30    None                                  0    None                0        0
FLAG QPD_B1p            Detection    1 None         30    None                                  0    None                0        0
FLAG QPD_B2             Detection    1 None         30    None                                  0    None                0        0
```

Figure 9 – DMSserver configuration file, FLAG part.

6 Snapshot JSON payload

The Snapshot JSON files are used by the WUI to display the status of the DMS at any given point in its past.

```
  1 {
  2     "metatron_info": ...,
 14     "subsystems": ...,
 31     "active_comments": {},
 32     "active_muting": ...,
100     "group_flags": ...,
164     "dms_info": ...,
179     "active_shelving": ...,
220     "providers": ...,
291     "active_alerts": ...,
315     "configuration_info": ...,
320     "flags": ...,
18790   "cond_flags_conflicts": ...
18834 }
```

Figure 10 – OnlineSnapshot.json, JSON file generated by DMSserver.

The branches of the above payload are explained here

Name	Value	Details	Description
metatron_info	object	dictionary	Contains a dictionary of Metatron data
subsystem	object	dictionary	Contains a dictionary of subsystem data
active_comments	object	dictionary	Contains a dictionary of active comments data
active_muting	object	dictionary	Contains a dictionary of active muting data
group_flags	object	dictionary	Contains a dictionary of group flags data
dms_info	object	dictionary	Contains a dictionary of dms info data

active_shelving	object	dictionary	Contains a dictionary of active shelving data
providers	object	dictionary	Contains a dictionary of providers data
active_alerts	object	dictionary	Contains a dictionary of active alerts data
configuration_info	object	dictionary	Contains a dictionary of configuration info data
flags	object	dictionary	Contains a dictionary of flags data
cond_flags_conflicts	object	dictionary	Contains a dictionary of condition flags conflicts data

6.1 metatron_info


```
2   "metatron_info": {
3       "state_info": {
4           "state": "DOWN",
5           "mode_length_s": 5176,
6           "state_length_s": 373,
7           "mode": "Maintenance"
8       },
9       "metadata": {
10          "last_mode_change_time": 1237621194,
11          "last_state_change_time": 1237625997
12      }
13  },
```


Figure 11 – OnlineSnapshot.json, metatron_info branch.

metatron_info is used as the root index of a dictionary having the following keywords:

Name	Value	Details	Description
state_info	object	dictionary	Contains a dictionary of Metatron info data
metadata	object	dictionary	Contains a dictionary of metadata

6.1.1 state_info

state_into is a dictionary containing the following elements:

Name	Value	Details	Description
state	string		Current ITF state
mode_lenght_s	number	integer	Segment of the current ITF mode [S]
state_lenght_s	number	integer	Segment of the current ITF state [S]
mode	string		Current ITF mode

6.1.2 metadata

metadata is a dictionary containing the following elements:

Name	Value	Details	Description
last_mode_change_time	Number	Integer	UNIXTIMESTAMP of the last ITF mode change
last_state_change_time	Number	Integer	UNIXTIMESTAMP of the last ITF state change

6.2 subsystem

```
14    "subsystems": {
15        "Calib_Hrec": 1,
16        "Environment": 5,
17        "SQZ": 5,
18        "Suspensions": 2,
19        "DAQ-Computing": 1,
20        "Vacuum": 3,
21        "TCS": 1,
22        "Detection": 2,
23        "VPM": 5,
24        "SBE": 5,
25        "ISC": 5,
26        "DetChar": 3,
27        "Injection": 5,
28        "Infrastructures": 5,
29        "ITFOnCall": 1
30    },
```

Figure 12 – OnlineSnapshot.json, subsystem branch.

The index of the subsystem dictionary is constituted of the related subsystem name and the codified value of the subsystem. For example:
"Calib_Hrec": 5,
"Environment": 3,

6.3 active_comments

active_comments is used as the root index of a dictionary that at moment is always empty because the functionality to add comment is not implemented.

6.4 active_muting

```
32    "active_muting": {
33        "mean_HVAC__CEB_HOT_CORR_60": {
34            "id_muting": 119
35        },
36        "mean_HVAC__EER_COLD_TE_ERR_60": {
37            "id_muting": 141
38        },
```

Figure 13 – OnlineSnapshot.json, active_muting branch.

active_muting is used as the root index of a dictionary; the indexes of this dictionary are the flag-name of the flag currently muted. For example:
mean_HVAC__CEB_HOT_CORR_60
mean_HVAC__EER_COLD_TE_ERR_60
To those flag-name index is associated another dictionary containing the following leaves:

Name	Value	Details	Description
id_muting	number	integer	Index of the related entry in the database

6.5 group_flags

```
100    "group_flags": {
101        "PR_Guard": 1,
102        "TcsAl": 1,
103        "ACS_WE": 1,
104        "Storage": 1,
105        "ACS_WAB": 1,
106        "DET_Area": 5,
```

Figure 14 – OnlineSnapshot.json, group_flags branch.

The index of the group_flags dictionary is constituted of the related group name and the codified value of the group. For example:
"PR_Guard": 1,
"TcsAl": 1,
"ACS_WE": 1,

6.6 dms_info


```
164    "dms_info": {
165        "cond_flags_conflicts": 6,
166        "tot_providers": "29",
167        "active_providers": "23",
168        "active_flags": "350",
169        "tot_flags": "549",
170        "file_write_unixtimestamp": 1553591145,
171        "active_channels": 3112,
172        "active_cond_flags": "2739",
173        "tot_channels:": 0,
174        "tot_cond_flags": "5826",
175        "file_write_gps": 1237626363,
176        "tot_channels": 6404,
177        "file_write_utc": "2019-03-26 09:05:45"
178    },
```


Figure 15 – OnlineSnapshot.json, dms_info branch.

dms_info is a dictionary containing the following elements:

Name	Value	Details	Description
cond_flags_conflicts	number	integer	Number of the current condition flags conflicts
tot_providers	number	integer	Number of the providers inserted in the database
active_providers	number	integer	Number of the providers read by the DMSserver
active_flags	number	integer	Number of the flags read by the DMSserver
tot_flags	number	integer	Number of the flags inserted in the database
file_write_unixtimestamp	number	integer	UNIXTIMESTAMP of the writing of the snapshot
active_channels	number	integer	Number of the channels (providers + flags + condition flags) read by the DMSserver
active_cond_flags	number	integer	Number of the condition flags read by the DMSserver
tot_channels	number	integer	Number of the channels (providers + flags + condition flags) inserted in the database
tot_cond_flags	number	integer	Number of the condition flags inserted in the database
file_write_gps	number	integer	GPS of the writing of the snapshot
file_write_utc	string		UTC of the writing of the snapshot

6.7 active_shelving

```
179     "active_shelving": {
180         "mean_ENV_DETR__DUST_0P5UM_30": {
181             "id_shelving": 889
182         },
183         "mean_HVAC__DET_FLUX_OUT_10": {
184             "id_shelving": 917
185         },
186         "mean_ENV_DETR__DUST_0P7UM_30": {
187             "id_shelving": 890
188         },
```

Figure 16 – OnlineSnapshot.json, active_shelving branch.

active_shelving is used as the root index of a dictionary; the indexes of this dictionary are the flag-name of the flag currently shelved. For example:
mean_HVAC__MCB_2_COLD_CORR_60
mean_HVAC__MCB_1_COLD_CORR_60
To those flag-name index is associated another dictionary containing the following leaves:

Name	Value	Details	Description
id_shelving	number	integer	Index of the related entry in the database

6.8 providers

```
220     "providers": {
221         "InjMoni": {
222             "state": 1
223         },
224         "InfraMoni": {
225             "state": 1
226         },
227         "SuspMoni": {
228             "state": 1
229         },
```

Figure 17 – OnlineSnapshot.json, providers branch.

providers is used as the root index of a dictionary; the indexes of this dictionary are the provider-name. For example:
InjMoni,
InfraMoni
To those provider-name index is associated another dictionary containing the following leaves:

Name	Value	Details	Description
state	number	integer	State (color) of the DMS subsystem

6.9 active_alerts


```
291     "active_alerts": {
292         "flags": [
293             "UNLOCK"
294         ],
295         "DMS": {
296             "channels": null,
297             "condition_flags": null,
298             "condition_flags_conflicts": null,
299             "flags": null,
300             "providers": null
301         },
302         "condition_flags": {
303             "META_ITF_LOCK__index": {
304                 "alert_sms_state": -1,
305                 "alert_notification_id": [
306                     30973
307                 ],
308                 "alert_email_state": -1,
309                 "alert_sound_state": 1
310             }
311         },
312         "group_flags": [],
313         "providers": {}
314     },
```


Figure 18 – OnlineSnapshot.json, active_alerts branch.

Active_alert is a dictionary containing the following elements:

Name	Value	Details	Description
flags	object	list	Contains a list of flags-names that has at least one condition flag in alarm
DMS	object	dictionary	Contains a dictionary of DMS alert data
condition_flags	object	dictionary	Contains a dictionary of condition flags alert data
group_flags	object	list	Contains a list of the current group-names that has at least one condition flag in alarm
providers	object	dictionary	Contains a dictionary of provider alert data

6.9.1 flags

the elements of the list are the flag-names that has at least one condition flag in alarm.

6.9.2 DMS

DMS is a dictionary containing the following elements:

Name	Value	Details	Description
channels	string		Number of the channel above the allowed threshold
condition_flags	string		Number of the condition flags above the allowed threshold
condition_flags_conflicts	string		Number of the condition flags conflicts above the allowed threshold
flags	string		Number of the flags above the allowed threshold
providers	string		Number of the providers above the allowed threshold

6.9.3 condition_flags

condition_flags is used as the root index of a dictionary; the indexes of this dictionary are the conditions flag-name who have an active alert. For example:
mean_ENV_CEB__DUST_0P3UM_30
mean_INF__WEB_HEATER_TE_IN_60
To those conditions flag-name index is associated another dictionary containing the following leaves:

Name	Value	Details	Description
alert_sms_state	number	integer	Current state of sms notification
alert_email_state	number	integer	Current state of email notification
alert_sound_state	number	integer	Current state of sound notification
alert_notification_id	number	integer	Index of the related entry in the database of the current notification

6.9.4 group_flags

the elements of the list are the group-names that has at least one condition flag in alarm.

6.9.5 providers

providers is used as the root index of a dictionary; the indexes of this dictionary are the providers-name who have an active alert. For example:
CaliMoni
EnvMoni
To those providers-name index is associated another dictionary containing the following leaves:

Name	Value	Details	Description
alert_sms_state	number	integer	Current state of sms notification
alert_email_state	number	integer	Current state of email notification
alert_sound_state	number	integer	Current state of sound notification
alert_notification_id	number	integer	Index of the related entry in the database of the current notification

6.10 configuration_info

```
315     "configuration_info": {
316         "id_configuration_file": 723,
317         "archive_path_json_configuration_file": "/data/archive_dms/json_configuration_files/
318         "archive_path_configuration_file": "/data/archive_dms/configuration_files/723.json"
319     },
```

Figure 19 – OnlineSnapshot.json, configuration_info branch.

Configuration_info is a dictionary containing the following elements:

Name	Value	Details	Description
id_configuration_file	number	integer	
archive_path_json_configuration_file	string		Path of the JSON configuration file
archive_path_configuration_file	string		Path of the configuration file

6.11 flags

```
320     "flags": {
321         "BACnet_devices": {
322             "state_delay_before_error": 0,
323             "flag_state": 1,
324             "cond_flags": {
325                 "ping_det_device": {
326                     "condition_flag_computed_value": 1.0,
327                     "condition_flag_comment": "",
328                     "condition_thresholds": "ping(det_device)",
329                     "condition_flag_state": 1
330                 },
331                 "ping_inj_device": {
332                     "condition_flag_computed_value": 1.0,
333                     "condition_flag_comment": "",
334                     "condition_thresholds": "ping(inj_device)",
335                     "condition_flag_state": 1
336                 }
337             }
338         },
339         "SWEB_SBE": {
```

Figure 20 – OnlineSnapshot.json, flags branch.

Flag names are used as the root index of the flags dictionary. To each flag-name index is associated another dictionary containing the following leaves:

Name	Value	Details	Description
state_delay_before_error	number	integer	
flag_state	number	integer	
cond_flags	object	dictionary	Contains a dictionary of data related to a condition applied to a flag.

The index of the condition_flags dictionary is constituted of the related condition flag name. For example:
mean_SPRB_DBOX_SBE_ps_temp_60
mean_SPRB_DBOX_SBE_slot3_temp_60
Each of these indices is associated to an associated dictionary, which contains the following elements:

Name	Value	Details	Description
condition_flag_computed_value	number	integer	
condition_flag_comment	string		
condition_thresholds	string		
condition_flag_state	number	integer	

6.12 cond_flags_conflicts


```
18790    "cond_flags_conflicts": {
18791        "mean_TFMoni_WEPCAL_PD1_PD2_60_Phase_10": {
18792            "10162": {
18793                "unix_cr": 1538156425,
18794                "unix_up": 1547398868,
18795                "flag_name": "NCAL",
18796                "provider": "CaliMoni"
18797            },
18798            "10166": {
18799                "unix_cr": 1538156425,
18800                "unix_up": 1553561714,
18801                "flag_name": "PCalWE",
18802                "provider": "CaliMoni"
18803            }
18804        },
```


Figure 21 – OnlineSnapshot.json, cond_flags_conflict branch.

cond_flags_conflicts is used as the root index of a dictionary; the indexes of this dictionary are the conditions flag-name that have name conflicts. For example:
mean_TFMoni_WEPCAL_PD1_PD2_60_Phase_10
mean_TFMoni_WEPCAL_PD1_PD2_60_Phase_10
The name conflict arises when the same channel is configured with the same criteria in two different places; to those conditions flag-name conflict are contains other dictionary having the root index of the condition flag-id of the database. For example:
10162
10166
Each of these indices is associated to a dictionary, which contains the following elements:

Name	Value	Details	Description
unix_cr	number	integer	UNIXTIMESTAMP of the first insertion in the database
unix_up	number	integer	UNIXTIMESTAMP of the last update in the database
flag_name	string		Name of the flag
provider	string		

6.13 Notes on the Snapshot JSON payload

While much of the Snapshot JSON payload is easily understandable, a few points should be taken into consideration when using it:

- the Snapshot JSON files do not contain the GPS start and stop times for alerts, muting or shelving. Instead they contain the values inserted to the database in relation whether they are

active or not;

- the alert_email_state, alert_sound_state and alert_sms_state default to a -1 value meaning that no alert of that type has been triggered. This value exists only in the Snapshot JSON files and not in the Online Database. This is because, where no alert is present, related alert information is obviously not stored in the database. However, the Snapshot JSON file, in order to provide a coherent and reproducible payload, must provide each of the alert branches regardless of whether an alert of that type is associated to it or not;
- the active alerts, shelving, muting and comments dictionaries are populated only by those conditional flags that are actually considered to be in any of those states at that moment.

7 The DMSpublisher in detail

The Publisher undertakes the following specific actions:

- compression of Snapshot JSON files;
- reading of raw data from the Online Database and writing it to trend;
- removing raw data from the Online Database, once it has been written to trend;
- compression of Trend JSON files.

This section describes each action in detail.
The Publisher runs every 10 seconds compresses the Snapshot JSON files. Every 30 minutes, it also converts the raw data to trend and compresses the results.
These parameters are configurable within the Publisher code. Furthermore, the Publisher is designed so that it can be paused as and when required, yet, on restart, recover the complete archive process, i.e. if paused, it simply begins to archive all those Snapshot files that have been produced during the period of time in which it has been offline.
The Publisher is written in Python.

7.1 Compression of Snapshot JSON files

When compressing the Snapshot JSON files, the Publisher undertakes the following actions:

- creates a JSON file that serves as an index in order to make finding the correct information again later, as easy as possible;
- checks if an open archive file in ZIP format is available, if not, it creates one. If an open file is available, it uses it. The new archive file uses the naming convention: {GPS_TIME_OF_FIRST_FILE_IN_ARCHIVE}.zip;
- gets any available Snapshot JSON files and puts them into the currently-open archive;
- updates the index file with the new information;
- deletes those original Snasphot JSON files that have been archived.

7.2 Reading raw data and writing it to Trend

The publisher reads the data in the Raw tables of the Online database every 30 minutes - this time is configurable - copies all data and then deletes any data that is more than 30 minutes old; again this time is configurable. It then converts them into JSON payloads, which are ultimately stored as Trend JSON files, which are, in turn, compressed into ZIP archives.
The Trend JSON files are used when a request is made via the WUI to reconstruct the history of a given Conditional Flag.

7.3 Compression of Trend JSON files

Periodically, the Publisher takes any data older than 30 minutes within the buffer files and transfers it directly to files that exist within the DMS Server memory and transfers them directly to a dedicated ZIP archive within the dedicated archive sub-directory. This leads to the production of archives

8 The Web User Interface

8.1 The homepage

The main-screen area display the sub-system and flag information. Each sub-system and flag have a dedicated container, the background color of which depends upon the associated state.
he homepage of the WUI should provide the following functionality:

- Flags, grouped into Subsystems, displaying the name of the flag, within a container taking the color of the Flag state;

- each Subsystem must have a container that takes the color of the state of the lowest-common denominator Flag within the Subsystem; exact details are described in the subsequent sections

of this paper;

- clicking on a Flag should provide details on the state of the Subsystem, with the focus being on the called Flag;
- clicking on a Subsystem should provide details on the state of the Subsystem;
- general information, including:
  - current UTC time, in the following format, e.g.: 'Fri Nov 18 11:09:58 2016';
  - current GPS time, in seconds, e.g.: 1163502615;
- information on the status of the interferometer:
  - ITF mode, e.g.: COMMISSIONING;
  - ITF State, e.g.: LOW_NOISE_3

The WUI is divided into the following main containers:

- a top bar - which sits in a fixed position at the top of the screen;
- a main-screen area - the whole main screen area;
- a group of buttons - this is available to the right-hand-side of the screen. These buttons include links to display the following:
  - a dashboard;
  - the all-flag list;
  - a log of all alerts;
  - the shelving interface;
  - the muting interface;
- a fixed-width left-sided container, which slides-in from the left-hand side of the screen when called and is used to display the following information:
  - a history of all alerts;
  - the shelving interface;
  - the muting interface.
- a dashboard - a responsive left-sided container, which fits to 95% of the width of the screen and displays a general DMS overview
- an individual-flag-details container, which fits to 90% of the width of the screen and is used to display individual flag-details, when called;
- a covering modal - a modal container that slides-in from the top of the screen and is used principally to provide users with generic information, such as warnings

For more details see also [How-to for the homepage]() and [How to open the DMS homepage]().

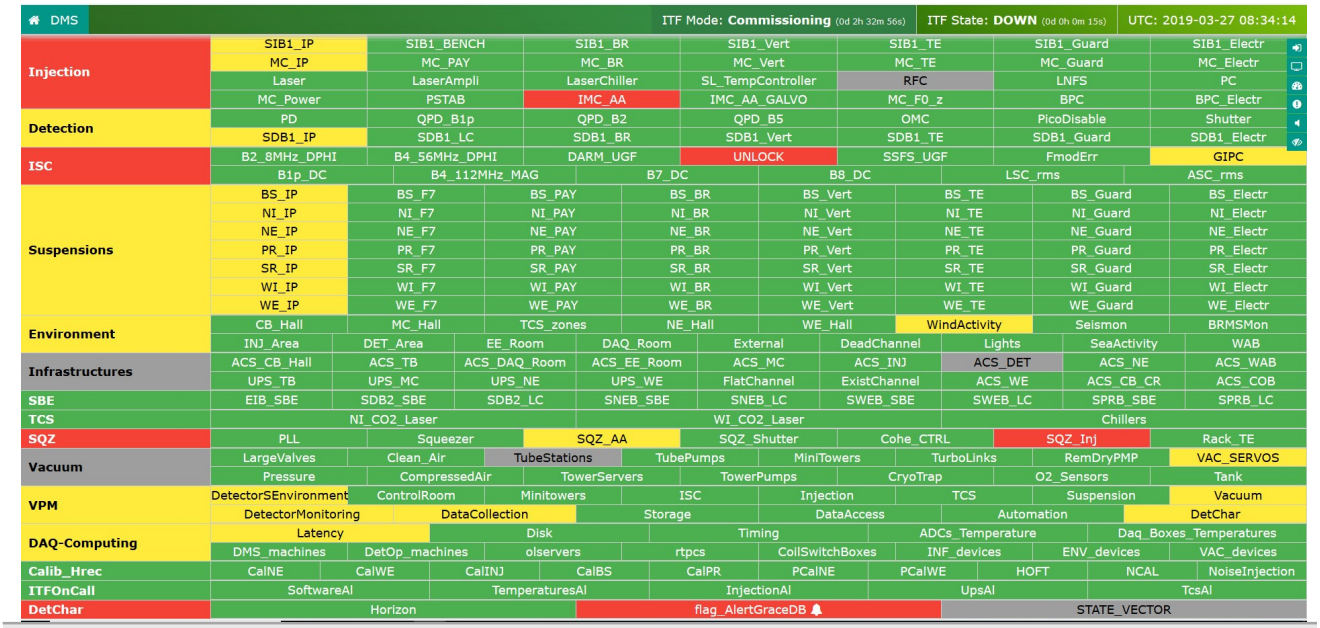


Figure 22 – DMS homepage

8.1.1 Subsystem states

The exact color a Subsystem container takes depends upon the lowest-common denominator Flag within that Subsystem and follows the logic described here:

- Subsystem Green - All Flags Green;
- Subsystem Yellow - At least one Flag Yellow and all other Flags Green;
- Subsystem Grey - At least one Flag Grey and all other Flags are not Red and are not Dark Grey;
- Subsystem Dark Grey - At least one Flag Dark Grey and no Flag is Red;
- Subsystem Red - At least one Flag Red

8.1.2 The individual-flag-information section

This section displays in the dedicated container and is automatically refreshed every 10s.
The following information is displayed in relation to the specific called flag:

- the called flag;
- if the flag is a group flag, the flags that constitute the group flag;
- the condition flags from which the flag (and any group-composition flags) is composed.
- the history - state, value, alert-notification, shelving and muting - of the specific flag;
- the possibility to shelve and/or mute the flag;
- a configurable plot showing, by default the last 30 minutes, of the life of the flag;
- the possibility to show plots of the associated condition flag(s).

In addition, it is possible in this page to select up to three more flags, against which the first flag can be compared
For more details see also [How to open the individual-flag-information section]

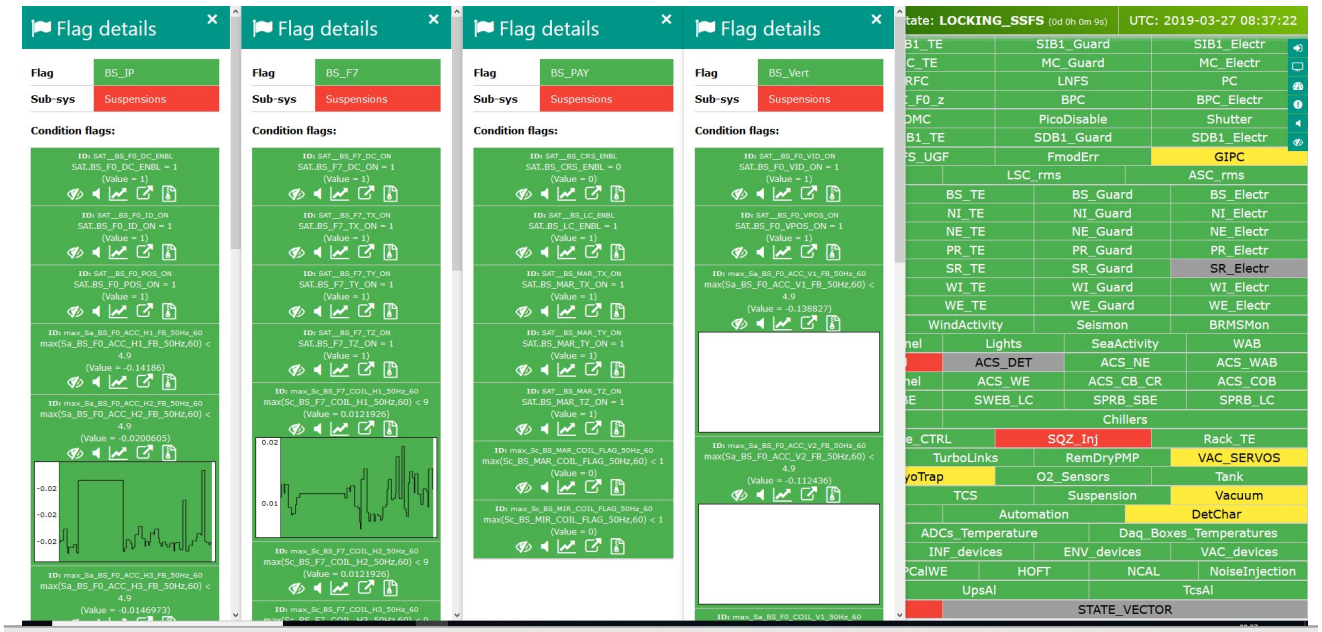


Figure 23 – DMS individual-flag-information section

8.1.2.1 Detail of the individual-flag-information section

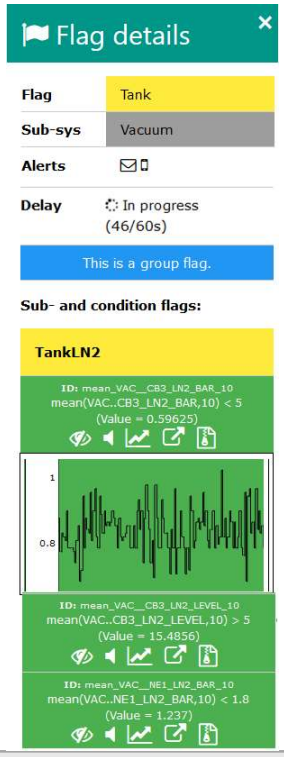
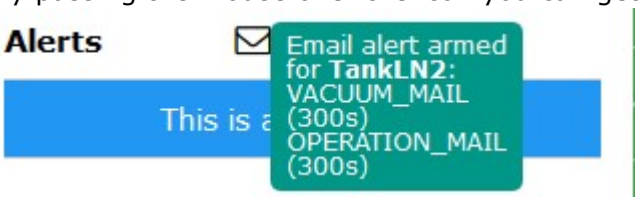


Figure 24 – detail of the individual-flag-information section

- Flag: it is name shown in the homepage, it could be also a group.
- Sub-sys: it is subsystem to which the flag belongs.
- Alerts: if present it indicates that the flag has been configured to send alert. Three kind of icons denotes the type of notification:
  →email notification
  → sms notification
  → sound notification
  By passing the mouse over the icon you can get info about the alert configuration:



- Delay: it indicates the status of the visualization delay
- **This is a group**

  It shows that this is a group.
- Sub- and condition flags: starting from here there is the list of flags and related condition flags;

The info for each condition flag are the following

- ID: is the ID associated to each condition flag;
- Condition flag criteria and thresholds;
- Condition flag current value;
- Condition flag comment associated to yellow or red flag;
- link to the shelving interface;
- link to the muting interface;
- link to the associated plot;

- link to the DMS event monitor;
- link to the DMS archive;
- a plot of the last 30 minutes that can be open/closed by clicking on the icon

8.1.3 The Dashboard

The dashboard provides information describing the current state of the DMS. It shows information including:

- User information;
- Sound alert acctivation;
- 100 most recent events: alert-notification, recently-shelved flags, recently-muted flags;
- links to open the "DMS event monitor WUI" and the "DMS currently shelved flags WUI"
- a list of all flags currently in conflict;

For more details see also How to open the dashboard.

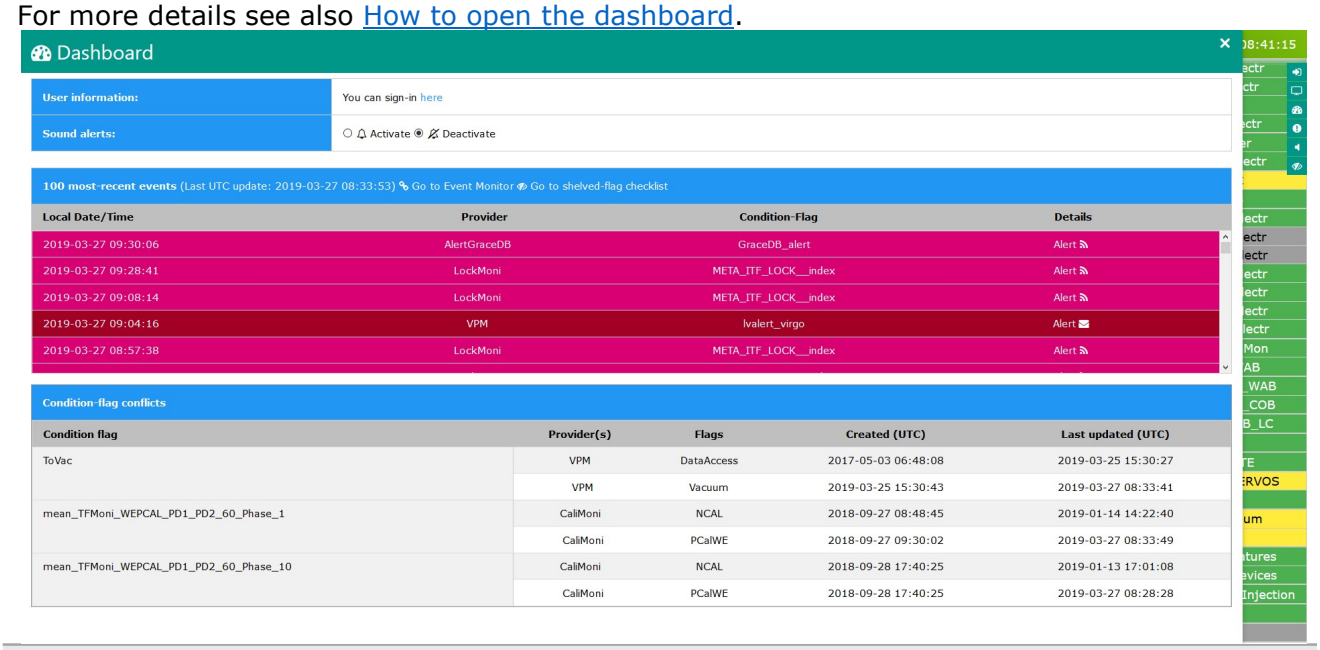


Figure 25 – DMS dashboard

8.1.4 The Alerts section

In this section, the 500 most recent alerts are displayed.
For each alert, the following information is displayed:

- the UTC time at which the notification was sent;
- the name of the Provider, Flag, Condition-Flag for which the notification has been sent;
- the message string sent with the notification;
- the recipients that received the notification;

For more details see also How to open the alerts section

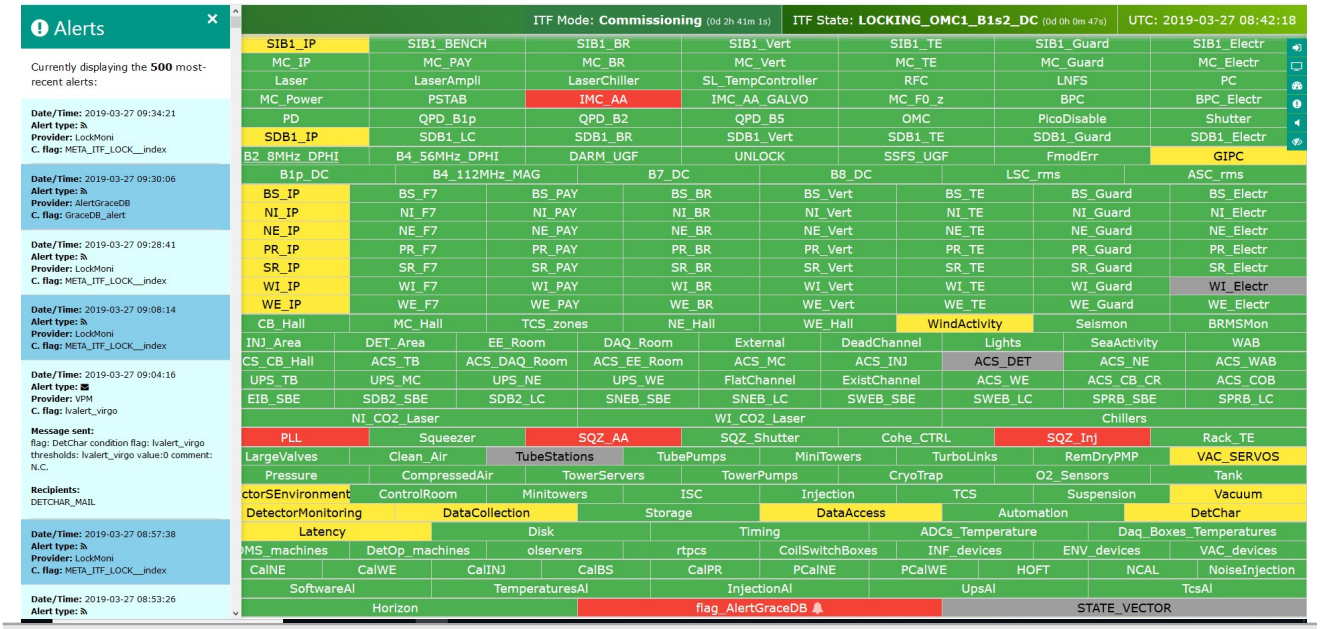


Figure 26 –DMS alert section

8.1.5 The Shelving section

In this section, condition-flags currently shelved displayed.
For each shelving, the following information is displayed:

- the UTC time at which the shelving was inserted;
- the name of condition-flag for which the shelving has been done;
- the name of the user that made the shelving;
- the UTC time at which the shelving starts;
- the UTC time at which the shelving ends;
- the reason of the shelving

For more details see also [How to open the shelving section](), [How to shelve-unshleve a condition flag]().

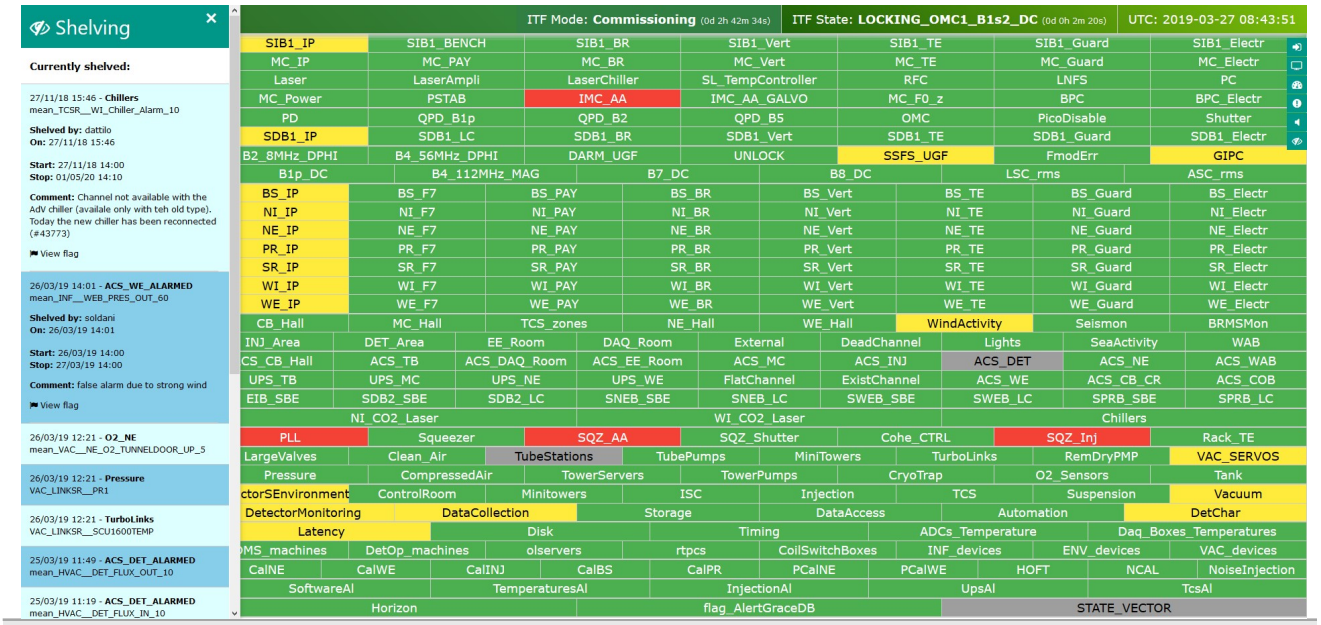


Figure 27 – DMS shelving section

8.1.6 The Muting section

In this section, condition-flags currently muted are displayed.
For each shelving, the following information is displayed:

- the UTC time at which the muting was inserted;
- the name of condition-flag for which the muting has been done;
- the name of the user that made the muting;
- the UTC time at which the muting starts;
- the UTC time at which the muting ends;
- the reason of the muting

For more details see also How to open the muting section, How to mute-unmute a condition flag.

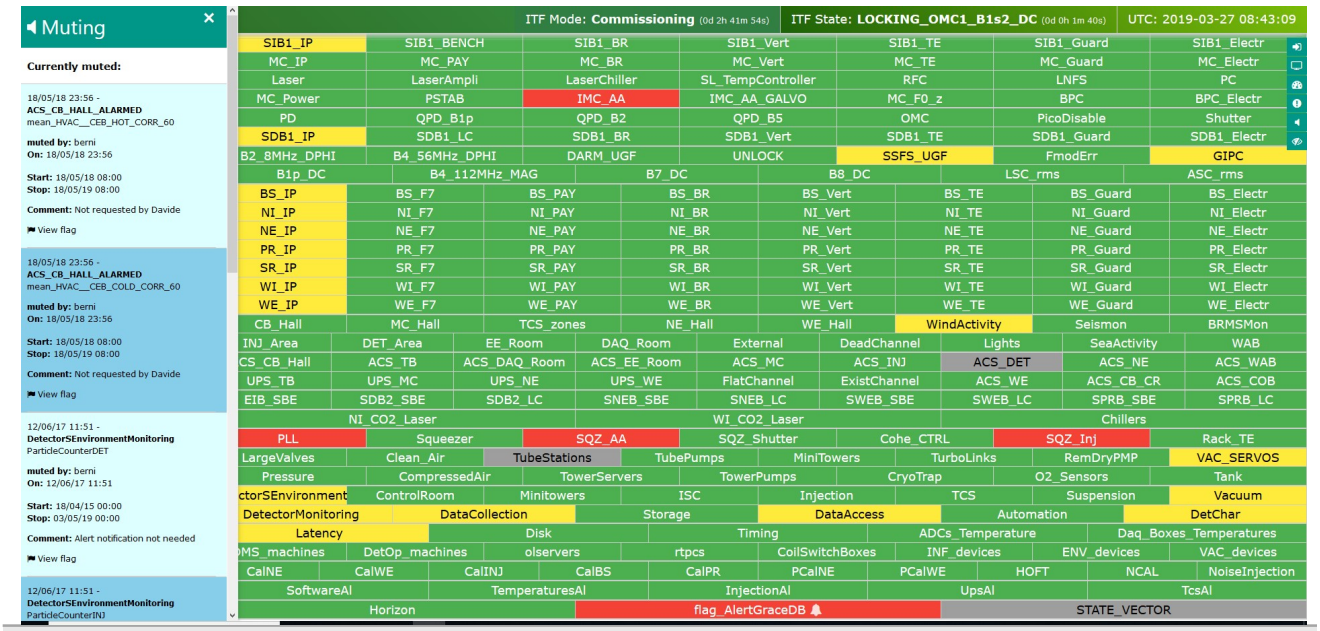


Figure 28 – DMS muting section

8.1.7 Associated Condition Flag plots

When selected this provides recent information on the state of the called Condition Flag. This takes the following form:

- by default the "up-sampled" data are displayed, with the option to choose the "down-sampled" data. "up-sampled" data are the last 30 minutes of data sampled every 10s while the "down-sampled" data are the last 3 weeks of data sampled every 60s.
- In the "up-sampled" data mode the plot is automatically refreshed every 10 seconds;
- In the "down-sampled" data mode the plot is paused;
- the data are zoom-able;
- it is possible to pan across previous and successive time periods;
- the background provides a shaded representation of the alert state associated to the Condition Flag during the selected time period;
- there is the possibility to pause the refresh of the plot;
- there is the possibility to download the plot as a png image.

For more details see also How-to for Associated Condition Flag plots.

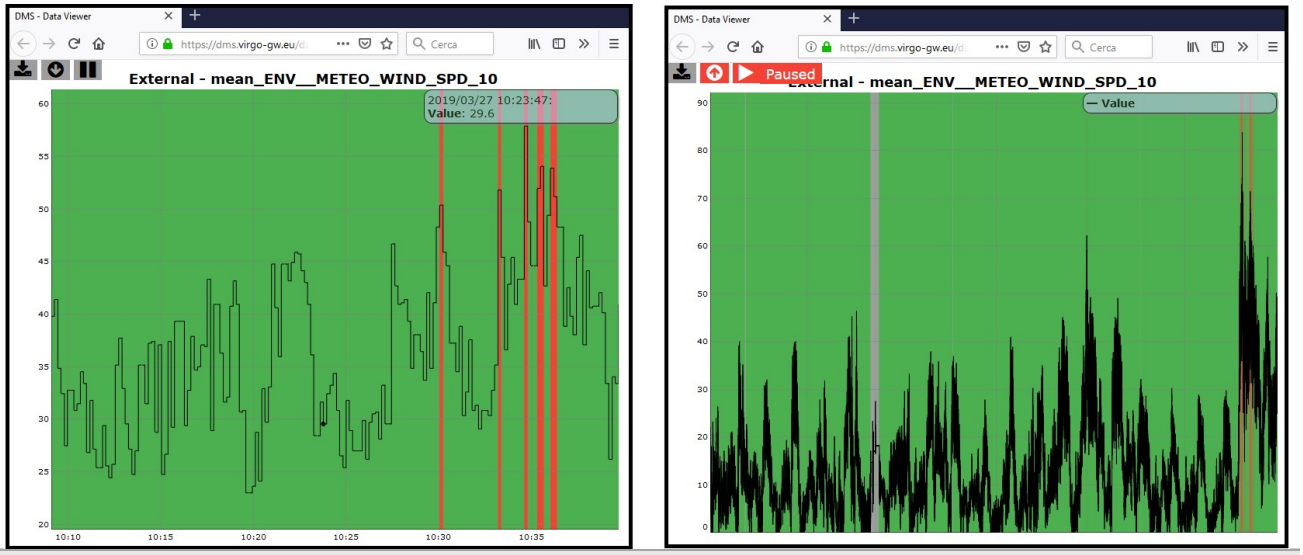


Figure 29 – associated condition flag plots

8.1.8 Responsiveness

The sub-systems display differently dependent upon the size of the screen being used.
On small screens, the DMS main-screen area display as follows on small-screens:

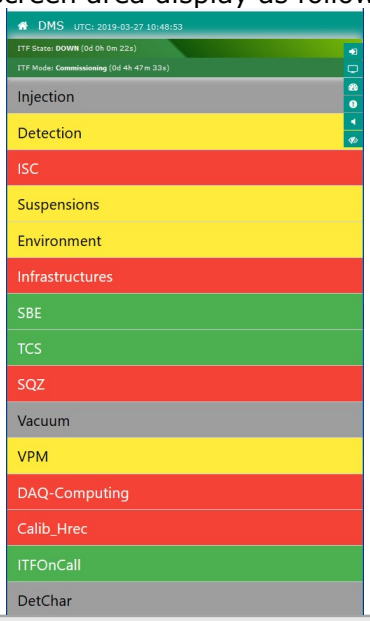


Figure 30 – DMS homepage on small screen

Note that, as long as performance allows, clicking on a flag should not open a new page in the browser, instead it should simply display the information that by default is hidden within the page, via a JavaScript onclick event.
Clicking on the name of the sub-system should re-close the sub-system, i.e. re-hide the associated flags.
Clicking on an individual flag shows its detailed information in the left nav-bar.

8.2 The DMS event monitor WUI

The DMS event log WUI shows the following DMS events:

- shelving events: new shelving, shelving expired, manual unshelve;
- muting events: new muting, muting expired, manual unmute;

- SMS alert notification;
- EMAIL alert notification;
- SOUND alert notification

Events are associated to one of the following color:

Color	*Event type*
	shelving
	muting
	SMS
	EMAIL
	SOUND

Through the section on the left it is possible to set some search criteria; the detail search provides a variety of searchable options. These have been tailored in such a way as to produce easily manageable and modifiable result sets.
The possible searchable areas are:

- Event type
- Condition flag
- Time window

The events are listed on the right part in ordered way; from the most recent to the less recent. For more details see also [How-to for DMS event monitor](#).

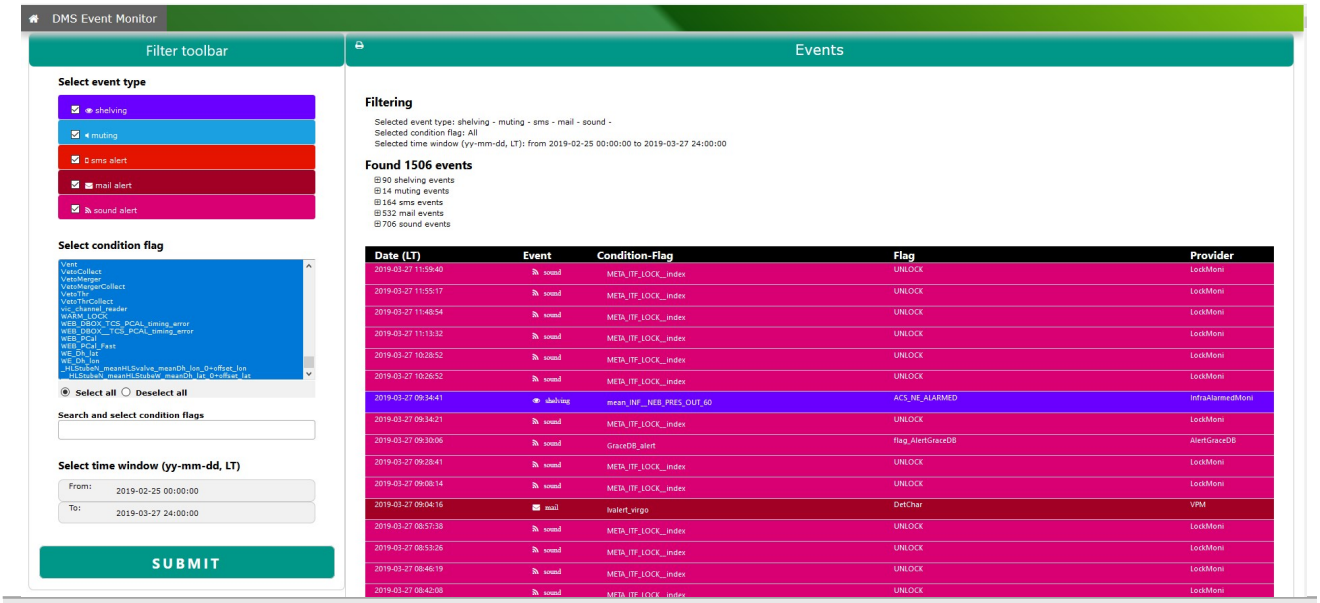


Figure 31 – DMS event monitor

8.3 The DMS playback WUI

ThePlay-back section allows users to display the DMS in the exact configuration and state in which it was at a given GPS time or between two different GPS times.
In the event of a single GPS time being selected, the main-content area ceases to display the currentDMS and instead shows the image at the requested time.
If both GPS-start and GPS-stop times are provided, then main-content area again ceases to display the current DMS information and instead shows the state between the two times. By default, the WUI loops through the snapshots within the GPS-time range, changing them with a customable frequency. However, the user is be able to pause/un-pause the play-back as well as change the play-back speed; both in terms of speeding it up and slowing it down.

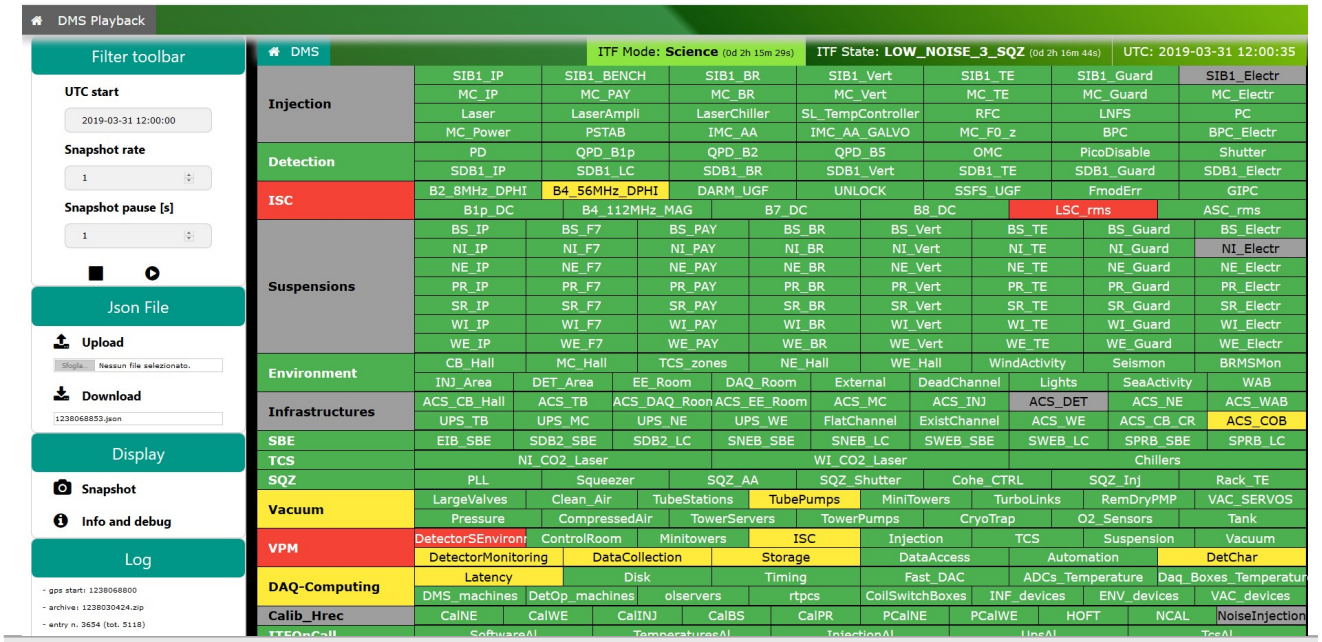


Figure 32 – DMS playback, "snapshot" display

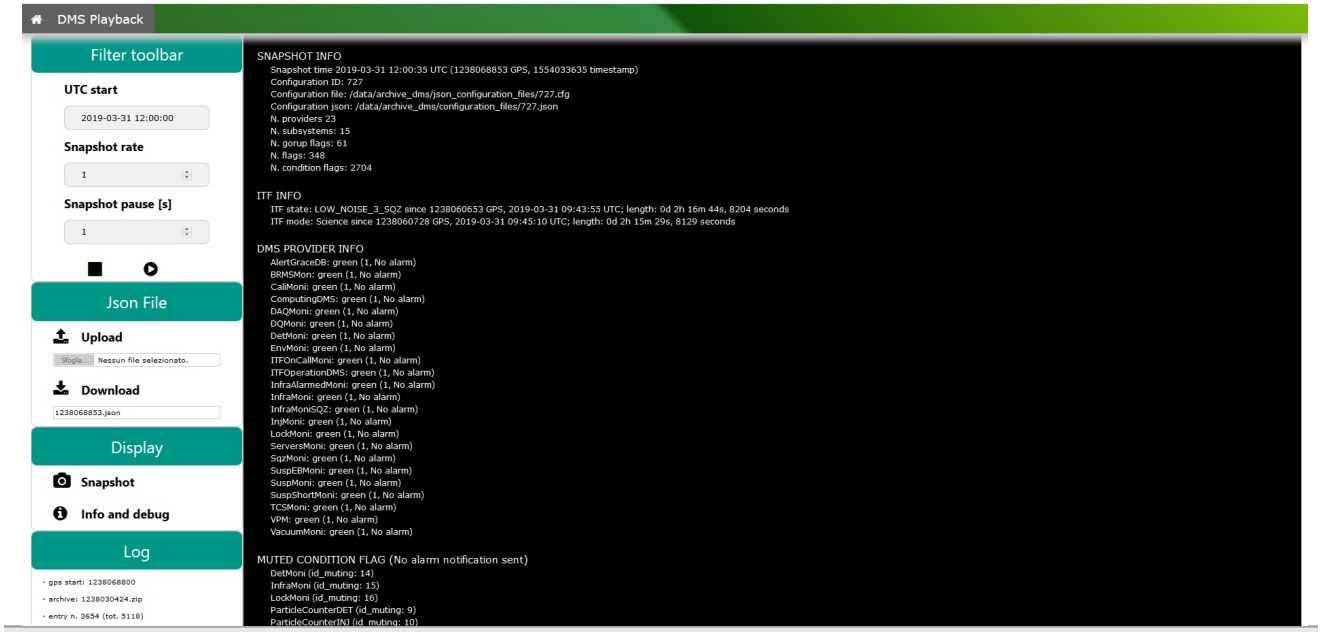


Figure 33 – DMS playback, "info and debug" display

There is the possibility to select the desired display:

- Snapshot: is the standard visualization of the DMS;
- Info and debug: is a different visualization of the DMS in which all the info contained in the JSON payload are shown in a more readable way.

The JSON payload can be downloaded and uploaded.

A dedicated python script, called GetPlaybackLinks, can be launched to get directly the link to a desired snapshot. The script needs as argument the GPS start and the GPS stop, it extracts from the archives all the snapshot between the two GPS and provides the URLs.

For more details see also [How-to for DMS playback](#).

8.4 The DMS archive WUI

The DMS archive section allows user to retrieve one year of information on the state of the called Condition Flag. The information is organized in twelve boxes; one box for one month. Each box behaves as the plot associated to a condition flag.
The user can select:

- Time window:
  - Last 12 months;
  - Year 2017, 2018, etc …;
- The sampling rate of the plot:
  - 60s
  - 600s
  - 1800s
  - 3600s
- The y scale:
  - autoscale;
  - manual scale:
    - y min;
    - y max;
- the plot resolution
  - x width [px]
  - y width [px]

For more details see also [How-to for DMS archive](#).

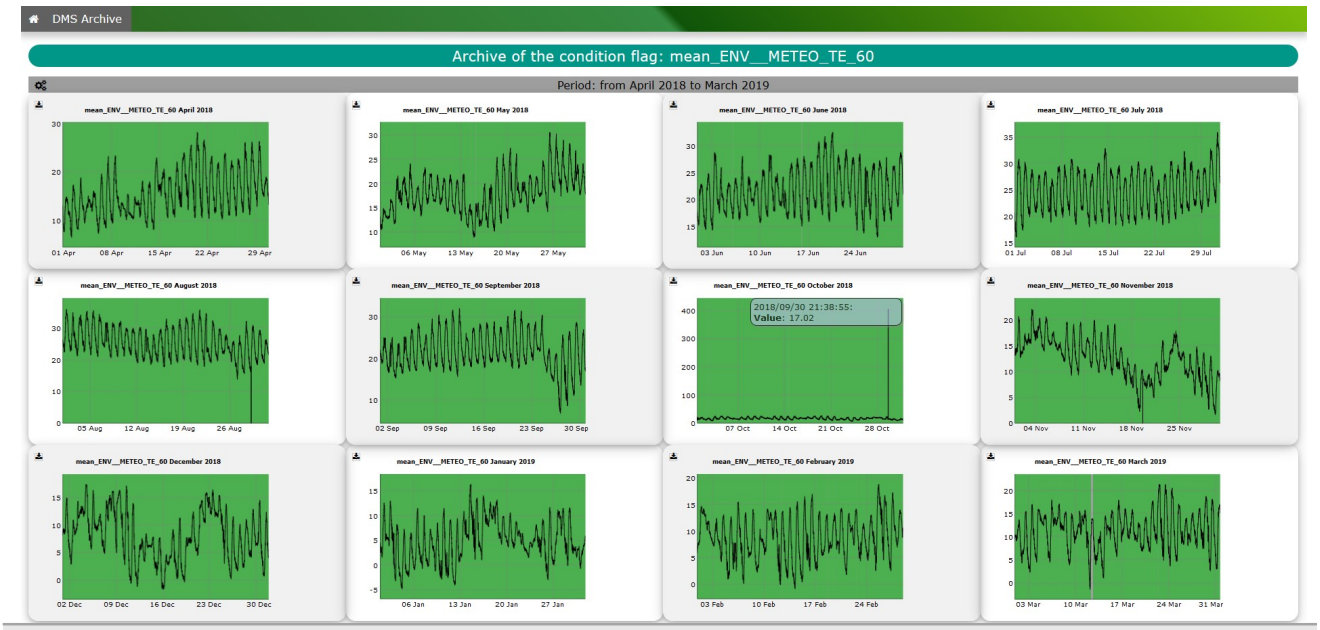


Figure 34 – DMS archive

8.5 The DMS currently shelved condition-flags WUI

The DMS currently shelved condition-flags shows the detailed information about the condition flags currently shelved. The information can be ordered by "ASC" or "DESC" by clicking on the specific column. After authentication the user can add additional note to the shelving, build a report to store an HTML file and also to un-shelve the condition flag.

For more details see also [How to for the DMS currently shelved condition-flags.](#)

Figure 35 – DMS currently shelved condition-flags

9 DMS workflow

This section describes with same example how a flag can be created from scratch: from the configuration of a Moni process, passing to the configuration of the DMS and then the visualization on the WUI.

9.1 Moni process configuration

```
…
QC_MONITOR * ACS_DET_ALARMED  "mean(HVAC..DET_FLUX_IN,10)<1000"        "Possible AHU failure"
…
```

Figure 36 – InfraMoni.cfg, detail of the configuration for ACS_DET_ALARMED

9.2 Moni process output: JSON payload

```
"ACS_DET_ALARMED": {
        "flag_state": 5,
        "flag_dq_value":1,
        "condition_flags" : {
          …
          "ACS_DET_ALARMED_04": {
            "channel_name": "HVAC..DET_FLUX_IN",
            "condition_flag_thresholds": "mean(HVAC..DET_FLUX_IN,10) < 1000",
            "condition_flag_comment": "Possible_AHU_failure",
            "condition_flag_computed_value": 10402.7,
            "condition_flag_state": 5,
            "condition_flag_dq_value": 1
          },
          …
```

Figure 37 – QcInfraMoni.json, flag ACS_DET_ALARMED branch

9.3 DMSserver configuration

```
…
FLAG ACS_DET_ALARMED          Infrastructures   1   ACS_DET              600
CONDITIONING_MAIL,OPERATION_MAIL     600    CONDITIONING_SMS   600    0
…
```

Figure 38 – DMSserver.cfg, detail of the configuration for ACS_DET_ALARMED

9.4 DMSserver output: snapshot JSON payload


```
…
"subsystems": {
        ...
        "Infrastructures": 3,
        ...
},
"group_flags": {
        …,
        "ACS_DET": 1,
       ...,
},
"active_shelving": {
        ...
        "mean_HVAC__DET_FLUX_IN_10": {
            "id_shelving": 916
        },
        ...
},
"providers": {
        ...
        "DetMoni": {
            "state": 1
        },
        ...
},
"flags": {
         ...,
         "ACS_DET_ALARMED": {
            "state_delay_before_error": 0,
            "flag_state": 1,
            "cond_flags": {
                    ...,
                    "mean_HVAC__DET_FLUX_IN_10": {
                        "condition_flag_computed_value": 10402.7,
                        "condition_flag_comment": "Possible_AHU_failure",
                        "condition_thresholds": "mean(HVAC..DET_FLUX_IN,10) < 1000",
                        "condition_flag_state": 1
                    },
                    ...
                }
            },
            ....
```


Figure 39 – online_snapshot.json, flag ACS_DET_ALARMED branch

9.5 Homepage

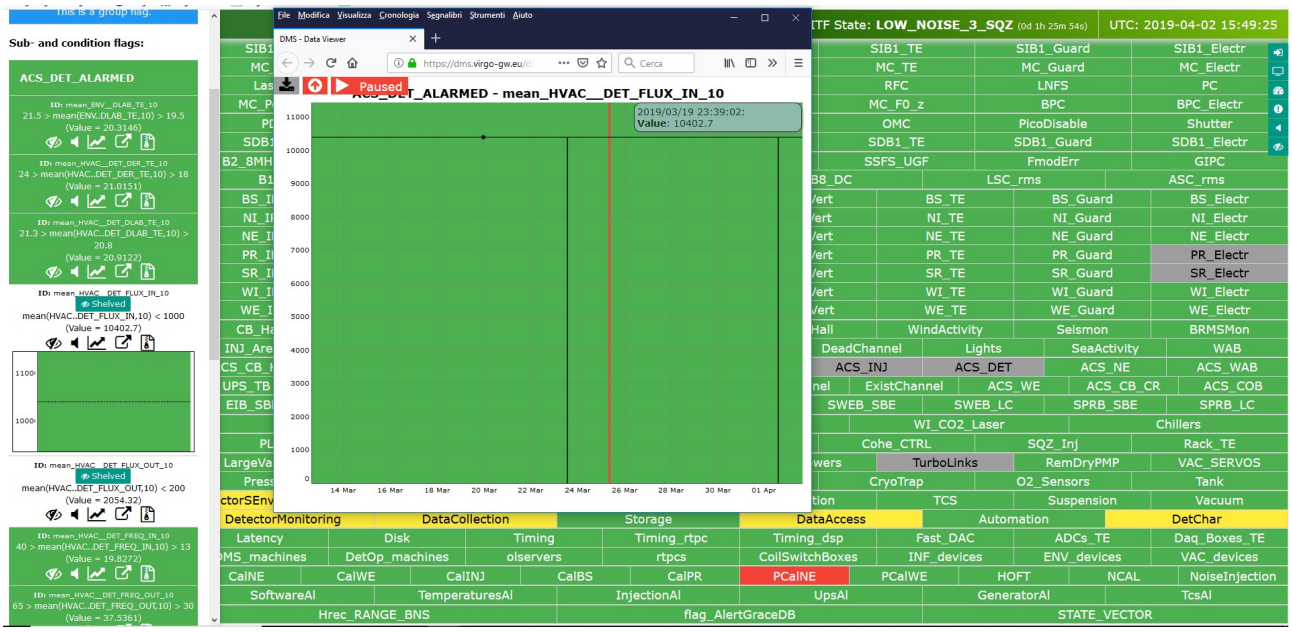


Figure 40 – DMS homepage, individual-flag-information section for ACS_DET_ALARMED, associated condition flag plots for ACS_DET_ALARMED

9.6 DMSpublisher: reading raw data and writing it to Trend

```
ctrl6[/data/archive_dms/trend/conditional_flags/archive]: ll
drwxrwxr-x. 2553 virgorun virgo   98304 Dec 15  2017 2017
drwxrwxr-x. 4485 virgorun virgo  172032 Dec 23 19:33 2018
drwxrwxr-x. 3366 virgorun virgo  135168 Apr  2 17:36 2019


ctrl6[/data/archive_dms/trend/conditional_flags/archive/2018]:
abs_Dh_lat
abs_NE_Dh_lat
…
drwxrwxr-x. 2 virgorun virgo   44 Dec  1 00:28 mean_HVAC__DET_FLUX_IN_10…
…
WE_Dh_lat
WE_Dh_lon

ctrl6[/data/archive_dms/trend/conditional_flags/archive/2018/mean_HVAC__DET_FLUX_IN_10]:
01.zip
02.zip
…
11.zip
12.zip
```

Figure 41 – content of the folder

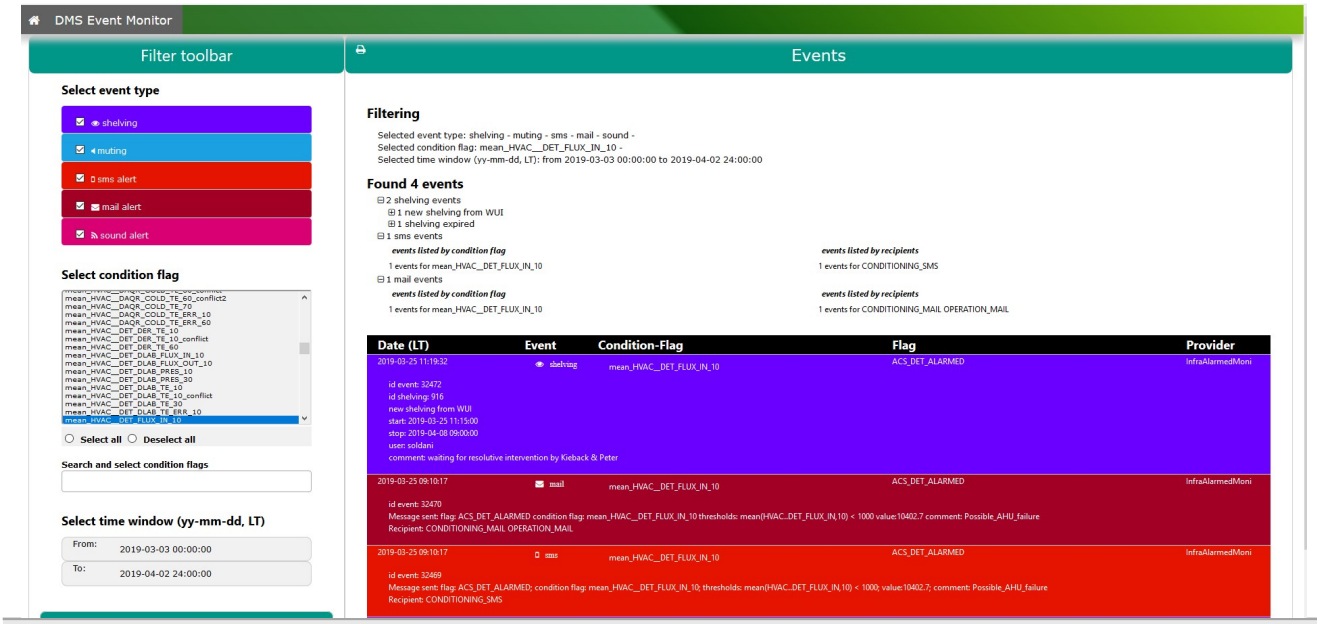
9.7 DMSpublisher: compression JSON snapshot

```
ctrl6[/data/archive_dms/archive_snapshots]: ll
total 257763584
-rwxrwxrwx. 1 virgorun virgo  544746804 Sep 12  2017 1189074163.zip
-rwxrwxrwx. 1 virgorun virgo  535867367 Sep 13  2017 1189214934.zip
```

```
-rwxrwxrwx. 1 virgorun virgo   535011050 Sep 15  2017 1189353246.zip
…
-rw-rw-r--. 1 virgorun virgo   652417981 Mar 31 03:21 1237925177.zip
-rw-rw-r--. 1 virgorun virgo   651368959 Apr  1 08:40 1238030424.zip
-rw-rw-r--. 1 virgorun virgo   658070236 Apr  2 14:25 1238135996.zip
-rw-rw-r--. 1 virgorun virgo    78787826 Apr  2 17:50 1238243059.zip
```

Figure 42 – content of the folder

9.8 DMS event monitor WUI



Select event type

☑ shelving
☑ muting
☑ sms alert
☑ mail alert
☑ sound alert

Select condition flag

mean_HVAC__DAQR_COLD_TE_60_conflict2
mean_HVAC__DAQR_COLD_TE_70
mean_HVAC__DAQR_COLD_TE_ERR_10
mean_HVAC__DAQR_COLD_TE_ERR_60
mean_HVAC__DET_DER_TE_10
mean_HVAC__DET_DER_TE_10_conflict
mean_HVAC__DET_DER_TE_60
mean_HVAC__DET_DLAB_FLUX_IN_10
mean_HVAC__DET_DLAB_FLUX_OUT_10
mean_HVAC__DET_DLAB_PRES_10
mean_HVAC__DET_DLAB_PRES_30
mean_HVAC__DET_DLAB_TE_10
mean_HVAC__DET_DLAB_TE_10_conflict
mean_HVAC__DET_DLAB_TE_30
mean_HVAC__DET_DLAB_TE_ERR_10
mean_HVAC__DET_FLUX_IN_10

○ Select all ○ Deselect all

Search and select condition flags

Select time window (yy-mm-dd, LT)

From: 2019-03-03 00:00:00
To: 2019-04-02 24:00:00

Filtering

Selected event type: shelving - muting - sms - mail - sound -
Selected condition flag: mean_HVAC__DET_FLUX_IN_10 -
Selected time window (yy-mm-dd, LT): from 2019-03-03 00:00:00 to 2019-04-02 24:00:00

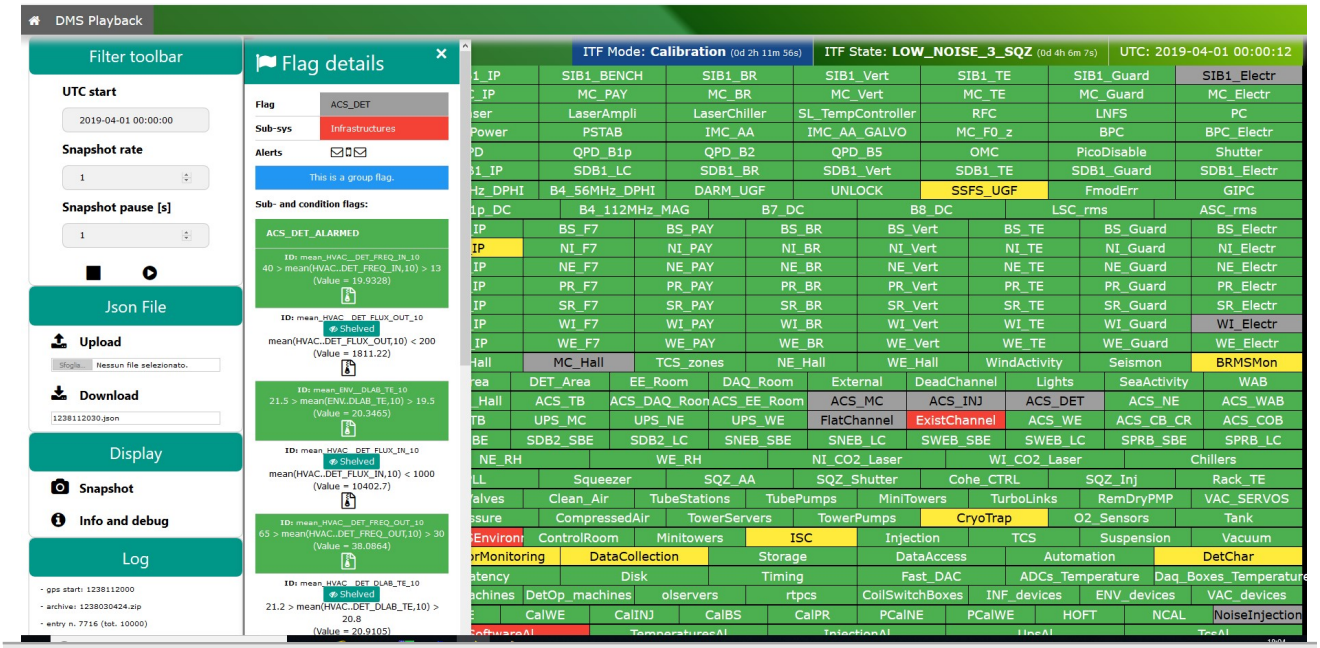
Found 4 events

- ⊟ 2 shelving events
  - ⊞ 1 new shelving from WUI
  - ⊞ 1 shelving expired
- ⊟ 1 sms events
  - *events listed by condition flag* — 1 events for mean_HVAC__DET_FLUX_IN_10
  - *events listed by recipients* — 1 events for CONDITIONING_SMS
- ⊟ 1 mail events
  - *events listed by condition flag* — 1 events for mean_HVAC__DET_FLUX_IN_10
  - *events listed by recipients* — 1 events for CONDITIONING_MAIL OPERATION_MAIL

Date (LT)	Event	Condition-Flag	Flag	Provider
2019-03-25 11:19:32	shelving	mean_HVAC__DET_FLUX_IN_10	ACS_DET_ALARMED	InfraAlarmedMoni
id event: 32472 id shelving: 916 new shelving from WUI start: 2019-03-25 11:15:00 stop: 2019-04-08 09:00:00 user: soldani comment: waiting for resolutive intervention by Kieback & Peter				
2019-03-25 09:10:17	mail	mean_HVAC__DET_FLUX_IN_10	ACS_DET_ALARMED	InfraAlarmedMoni
id event: 32470 Message sent: flag: ACS_DET_ALARMED condition flag: mean_HVAC__DET_FLUX_IN_10 thresholds: mean(HVAC..DET_FLUX_IN,10) < 1000 value:10402.7 comment: Possible_AHU_failure Recipient: CONDITIONING_MAIL OPERATION_MAIL				
2019-03-25 09:10:17	sms	mean_HVAC__DET_FLUX_IN_10	ACS_DET_ALARMED	InfraAlarmedMoni
id event: 32469 Message sent: flag: ACS_DET_ALARMED; condition flag: mean_HVAC__DET_FLUX_IN_10; thresholds: mean(HVAC..DET_FLUX_IN,10) < 1000; value:10402.7; comment: Possible_AHU_failure Recipient: CONDITIONING_SMS				

Figure 43 –DMS event monitor WUI

9.9 DMS playback WUI



UTC start: 2019-04-01 00:00:00

Snapshot rate: 1

Snapshot pause [s]: 1

Upload: Sfoglia... Nessun file selezionato.

Download: 1238112030.json

Snapshot

Info and debug

- gps start: 1238112000
- archive: 1238030424.zip
- entry n. 7716 (tot. 10000)

Flag	ACS_DET
Sub-sys	Infrastructures
Alerts	✉ ▯ ✉

This is a group flag.

Sub- and condition flags:

ACS_DET_ALARMED

ID: mean_HVAC__DET_FREQ_IN_10
40 > mean(HVAC..DET_FREQ_IN,10) > 13
(Value = 19.9328)

ID: mean_HVAC__DET_FLUX_OUT_10
Shelved
mean(HVAC..DET_FLUX_OUT,10) < 200
(Value = 1811.22)

ID: mean_ENV__DLAB_TE_10
21.5 > mean(ENV..DLAB_TE,10) > 19.5
(Value = 20.3465)

ID: mean_HVAC__DET_FLUX_IN_10
Shelved
mean(HVAC..DET_FLUX_IN,10) < 1000
(Value = 10402.7)

ID: mean_HVAC__DET_FREQ_OUT_10
65 > mean(HVAC..DET_FREQ_OUT,10) > 30
(Value = 38.0864)

ID: mean_HVAC__DET_DLAB_TE_10
Shelved
21.2 > mean(HVAC..DET_DLAB_TE,10) > 20.8
(Value = 20.9105)

ITF Mode: **Calibration** (0d 2h 11m 56s) | ITF State: **LOW_NOISE_3_SQZ** (0d 4h 6m 7s) | UTC: 2019-04-01 00:00:12

Figure 44 – DMS playback WUI

9.10 DMS archive WUI

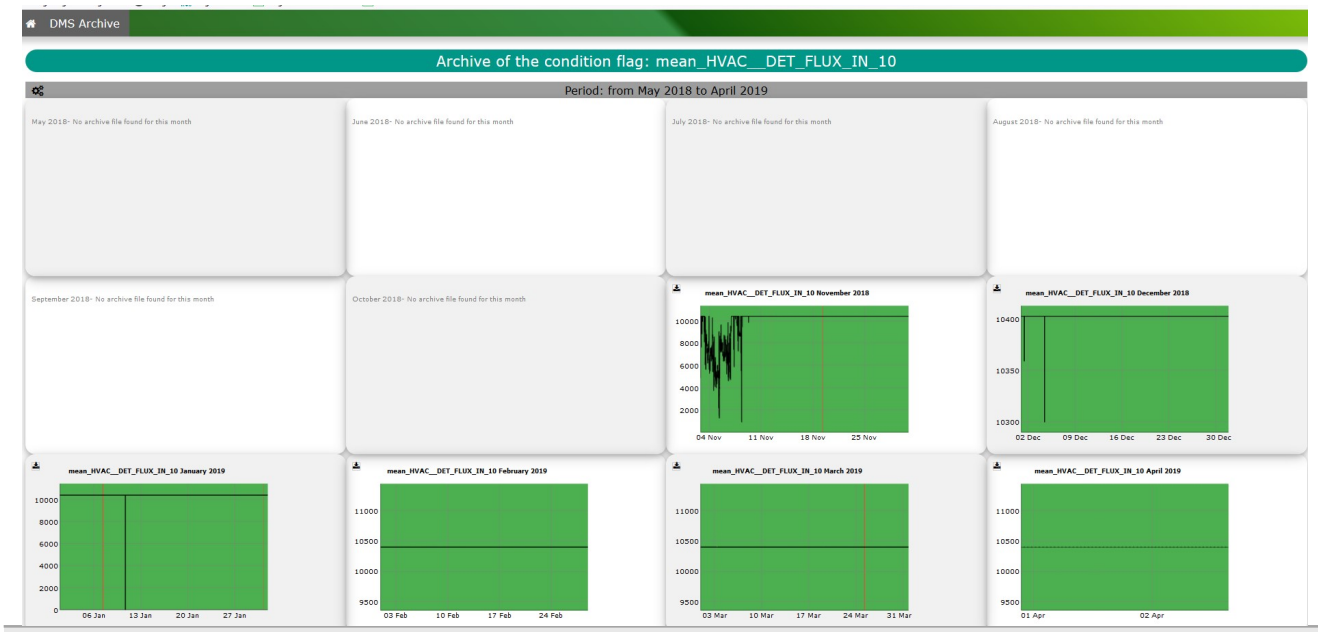


Figure 45 – DMS archive WUI

10 How-to

10.1 How-to for the Moni processes

The operation described below can be performed only by expert users. In case of doubt please contact the operation team.

10.1.1 How to configure a generic channel in the Moni process

- **Check that the channel is available in the DAQ**
  - To do this you can use Datadisplay and connect to online provider to search for the desired channel; if present in the list it can be defined.
- **Edit the Moni configuration file**
  - From VPM open the configuration file of the Moni process
  - Set the FDIN_TAG in the header section; you have to specific the prefix of the channel: V1_ENV, V1_HVAC, …
  - Define the channel; the generic syntax of the row is the following:

keyword	ITF_LOCK_state	Flag name	mathematical function of the channel and thresholds for red flag	comment associated for the red flag	mathematical function of the channel and thresholds for yellow flag	comment associated for the yellow flag

```
QC_MONITOR * TubePumpsNorth  "0<VAC_TUBE600N..PR1<0.3" "600 N Turbo could have a problem"
"*<0.1"   "TBD"
```

```
QC_MONITOR * TubePumpsWest  "VAC_TUBE600W..PR2<1e-7"          "600 W Turbo could have a problem"
"*<2e-8"  "TBD"
```

- **Stop and restart the process**

10.1.2 How to configure a channel computed with the mean() mathematical function

```
QC_MONITOR * TCS_zone  "18<mean(ENV..TCS_CHILROOM_TE,30)<28.0" "Temperature in TCS Chiller Room
is too High/Low"   "*<26.0"   "Temperature in TCS Chiller Room is quite High"
```

10.1.3 How to configure a channel computed with the rms() mathematical function

```
QC_MONITOR * DetectionLab  "rms(ENV_EDB_MIC,10)<0.1" "Microphone on External Detection Bench is too
high"
```

10.1.4 How to configure a channel computed with the brms() mathematical function

```
QC_MONITOR * MC_Hall  "brms(ENV_MC_ACC_Z,1,10,3,500)<0.004"     "Accelerometer at MC tower is too
High, brms(0.1Hz to 1000Hz)"
```

10.1.5 How to configure a channel computed with the delta() mathematical function

```
QC_MONITOR * EnvServers  "delta(ENV..EAB1_TE,1800)>1e-08"  "EnvServer frozen"
```

10.1.6 How to configure a channel computed with the exist() mathematical function

```
QC_MONITOR * ExistChannel  "exist(HVAC..CEB_HOT_CORR)>0"     "channel not aquired"
```

10.1.7 How to configure a channel with thresholds depending by the value of ITF_LOCK_STATE

```
QC_MONITOR *to124.5 B2_8MHz_DPHI   "-100<mean(LSC_B2_8MHz_DPHI,60)<100"                    "nn"
QC_MONITOR 125to*   B2_8MHz_DPHI   "-0.4<mean(LSC_B2_8MHz_DPHI,60)<0.4""B2_8MHz phase mistuned"
            "-0.2<*<0.2"  "B2_8MHz phase mistuned"
```

```
QC_MONITOR *to149    B1p_DC_rms"rms(LSC_B1p_DC,30)>0""B1p_DC is fluctuating too much"
QC_MONITOR 149.5to* B1p_DC_rms "rms(LSC_B1p_DC,30)<0.003" "B1p_DC is fluctuating too much
"rms(LSC_B1p_DC,30)<0.0015"   "B1p_DC is fluctuating"
```

```
QC_MONITOR *to40.5 ExtPLL  "SQZ_PLL_Ext_Status>-1" "ExtPLL unlocked!"
QC_MONITOR 41to* ExtPLL  "SQZ_PLL_Ext_Status>1.5" "ExtPLL unlocked!"
```

10.2 How to for the ComputingDMS process

The operation described below can be performed only by expert users. In case of doubt please contact the operation team.
To add a channel in this process you must edit the configuration file and then stop and restart the process from VPM.

10.2.1 How to configure a channel computed with the ping()function

```
CF    ping(olserver38)   olserver38  olserver38.virgo.infn.it   None   ping    None   0    0
```

10.2.2 How to configure a channel computed with the cpu_user()function

```
CF    cpu_user(olserver38) olserver38 olserver38.virgo.infn.it   New_Olservers  cpu_user max 70  90
```

10.2.3 How to configure a channel computed with the load()function

```
CF load(olserver137)  olserver137 olserver137.virgo.infn.it   DMS load_1-min   max  7    10
```

10.2.4 How to configure a channel computed with the mem_use ()function

```
CF  mem_use(olserver137)   olserver137  olserver137.virgo.infn.it  DMS mem_use  max 10e9 11e9
```

10.2.5 How to configure a channel computed with the mem_swap()function

```
CF mem_swap(olserver137)  olserver137 olserver137.virgo.infn.it    DMS  mem_swapmax1e9  2e9
```

10.3 How-to for DMSserver

The operation described below can be performed only by expert users. In case of doubt please contact the operation team.

10.3.1 How to stop and start the DMSserver

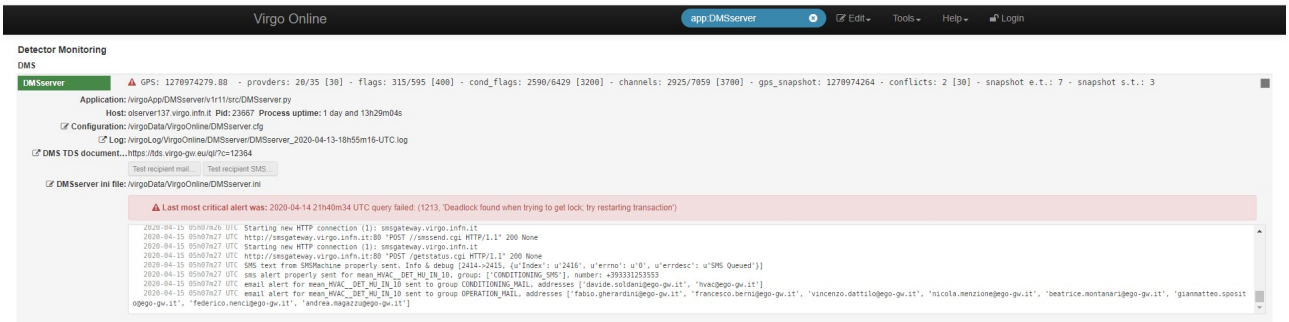


Figure 46 – DMSserver application in VPM

The DMSserver can be stopped and restarted from the VPM; the application name is: DMSserver.

10.3.2 How to configure a provider

- Check that the provider you are interested in is writing a JSON file;
- Get the path of the JSON file;
- Edit the DMS configuration file;

```
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcInfrastructuresAlarmedData.jsonInfraAlarmedMoni
0None    0    None00
```

- Stop and restart the server

10.3.2.1 How to configure a provider with "delay_before_unavailable_s"

```
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcInfrastructuresAlarmedData.jsonInfraAlarmedMoni
60None    0    None00
```

10.3.2.2 How to configure a provider with email notification

```
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcInfrastructuresAlarmedData.jsonInfraAlarmedMoni
60OPERATION_MAIL600 None00
```

```
ROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcSuspensionsData.json                    SuspMoni
120  OPERATION_MAIL,SUSP_MAIL 600 None 0 0
```

10.3.2.3 How to configure a provider with SMS notification

```
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcInfrastructuresAlarmedData.jsonInfraAlarmedMoni
60OPERATION_MAIL600 OPERATION_SMS6000
```

```
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcSuspensionsData.json                    SuspMoni
120  OPERATION_MAIL,SUSP_MAIL 600 OPERATION_SMS,SUSP_SMS 600 0
```

10.3.2.4 How to configure a provider with SOUND notification

```
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcInfrastructuresAlarmedData.jsonInfraAlarmedMoni
60OPERATION_MAIL600 OPERATION_SMS60060
```

```
PROVIDER  /opt/MonitoringWeb/buffer_dms/json_qcmoni/QcSuspensionsData.json                    SuspMoni
120  OPERATION_MAIL,SUSP_MAIL 600 OPERATION_SMS,SUSP_SMS 600 0
```

10.3.3 How to configure an alert recipient

The operation described below can be performed only by expert users. In case of doubt please contact the operation team.

10.3.3.1 How to configure an alert recipient for email notification

```
ALERT_RECIPIENT OPERATION_MAILuser1@ego-gw.it
```

```
ALERT_RECIPIENT SUSP_MAILuser2@ego-gw.it, user3@ego-gw.it
```

10.3.3.2 How to configure an alert recipient for sms notification

```
ALERT_RECIPIENT SUSPENSIONS_SMS 393381212345,338585987
```

```
ALERT_RECIPIENT OPERATION_SMS 393383535265
```

10.3.4 How to test an alert recipient

The test can be done by clicking on a dedicated buttons on the VPM after authentication. The buttons are in the DMSServer section.

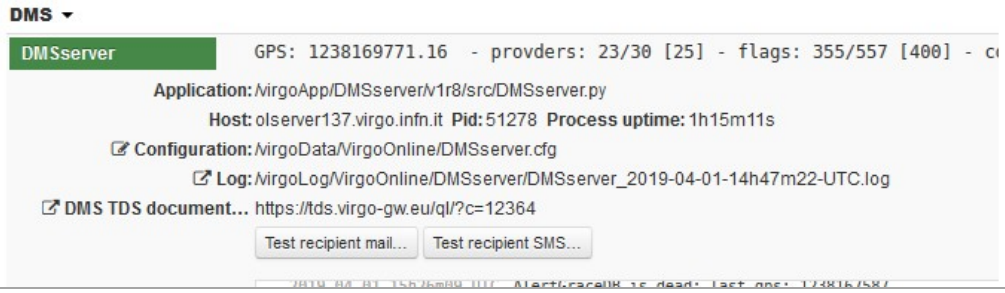


Figure 47 – DMSserver application in VPM, testing alert recipient

10.3.4.1 How to test an email recipient

- Click on the button “Test recipient mail”;
- In the modal window write the value of the recipient you are interested in to test in the input text;
- Click the button “Send Command and Close Window”;

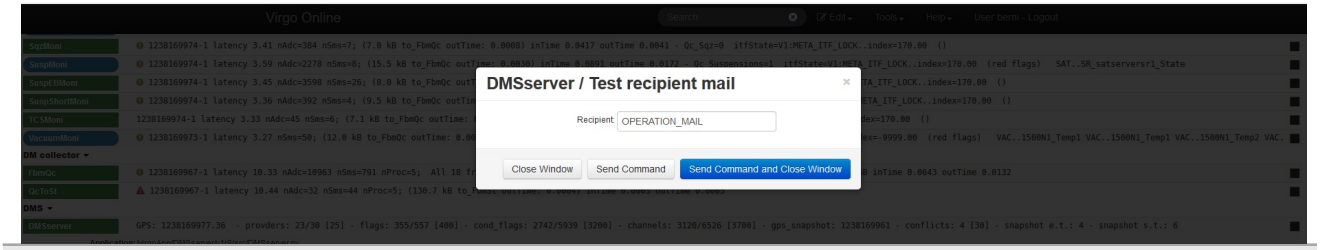


Figure 48 – DMSserver application in VPM, testing alert recipient.

10.3.4.2 How to test an SMS recipient

- Click on the button “Test recipient SMS”;
- In the modal window write the value of the recipient you are interested in to test in the input text;
- Click the button “Send Command and Close Window”;

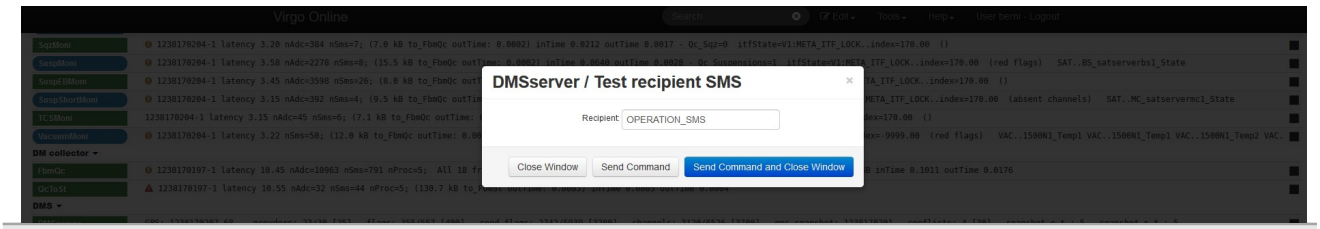


Figure 49 – DMSserver application in VPM, testing alert recipient.

10.3.5 How to configure a flag in the DMSserver

The operation described below can be performed only by expert users. In case of doubt please contact the operation team.

- Check that the flag and its condition flags are available in the JSON payload generated by one DMS provider; this operation can also done by looking at the status of the related Moni server in the VPM checking the “lost” signals.
- Check that the DMS provider is properly configured in the DMS configuration file;
- Edit the DMS configuration file;

```
FLAG NI_Guard_Trigger   Suspensions   2   None  0 None 0 None 0 0
```

- Stop and restart the server

10.3.5.1 How to configure a flag in the DMSserver with “group”

```
FLAG NI_Guard_Trigger   Suspensions   2   NI_Guard  0 None 0 None 0 0
```

10.3.5.2 How to configure a flag in the DMSserver with “delay_before_visualization”

```
FLAG NI_Guard_Trigger   Suspensions   2   NI_Guard30 None 0 None 0 0
```

```
FLAG NI_IP    Suspensions   2   None    30    None       0    None      0     0
```

10.3.5.3 How to configure a flag in the DMSserver with email notification

```
FLAG NI_Guard_Trigger    Suspensions   2   NI_Guard30 OPERATION_MAIL,SUSP_MAIL600  None 0 0
```

```
FLAG NI_IP    Suspensions   2   None    0SUSP_MAIL500    None      0     0
```

10.3.5.4 How to configure a flag in the DMSserver with sms notification

```
FLAG NI_Guard_Trigger    Suspensions   2   NI_Guard30 OPERATION_MAIL,SUSP_MAIL600 OPERATION_SMS
900 0
```

```
FLAG NI_IP    Suspensions   2   None    0None0SUSP_SMS600     0
```

```
FLAG NI_F7    Suspensions   2   None    0None0None01800
```

10.4 How-to for the homepage

10.4.1 How to open the DMS homepage

The link to open the DMS homepage is the following:

https://dms.virgo-gw.eu/

it can be open only from the internal network (or after firewall authentication).

10.4.2 How to open the individual-flag-information section

To open the information of the individual flag you have to click on the flag you are interested in at the level of the homepage.

10.4.3 How-to for Associated Condition Flag plots

10.4.3.1 How to open the associated condition flag plot

- Open the individual-flag-information section for the desired flag
- Click on the icon 

If you want to open a dedicated window containing the plot you can click inside box containing the plot just open.

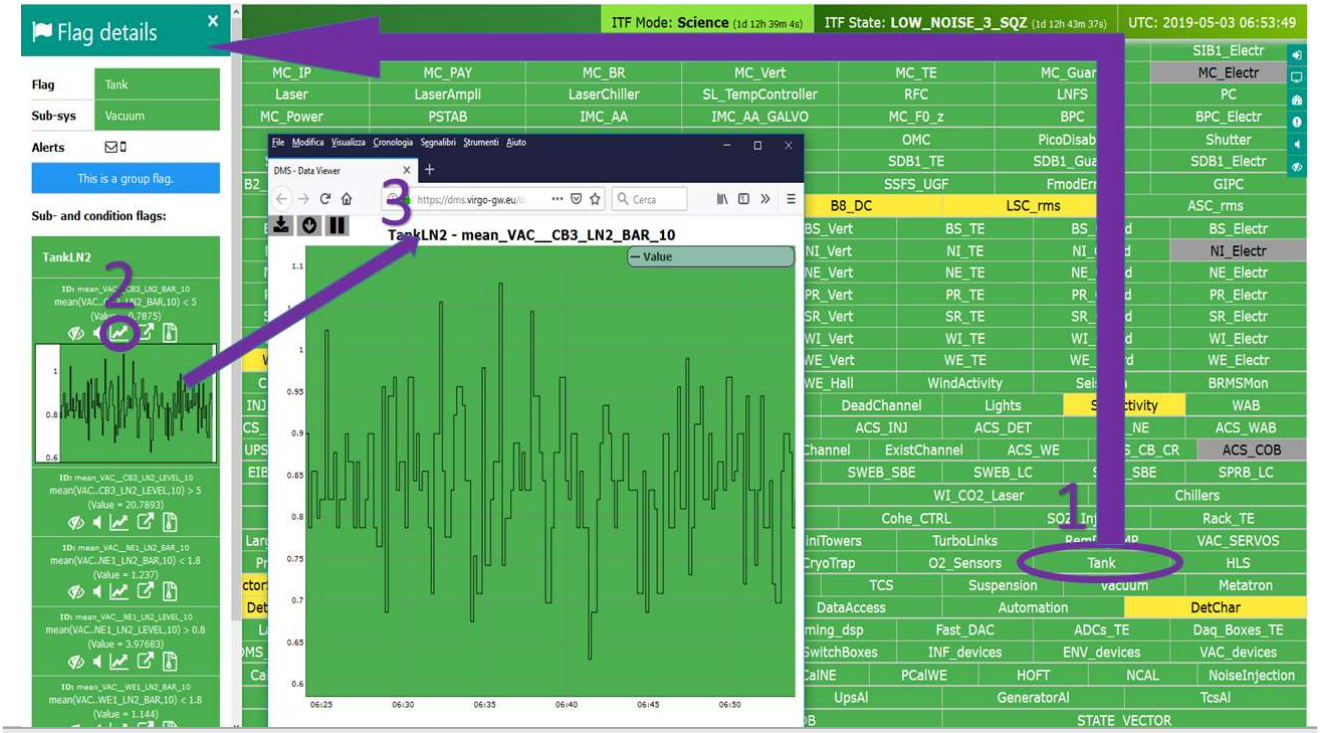


Figure 50 – how to open the associated condition flag plot

10.4.3.2 How to switch from up-sampled/down-sampled data

- Open the dedicated window containing the plot.

When you open the associated condition flag plot the default visualization is "up-sampled data" (30 minutes of data sampled every 10s).

To switch from "up-sampled data" to "down-sampled data" click on the icon

To switch from "down-sampled data" to "up-sampled data" click on the icon

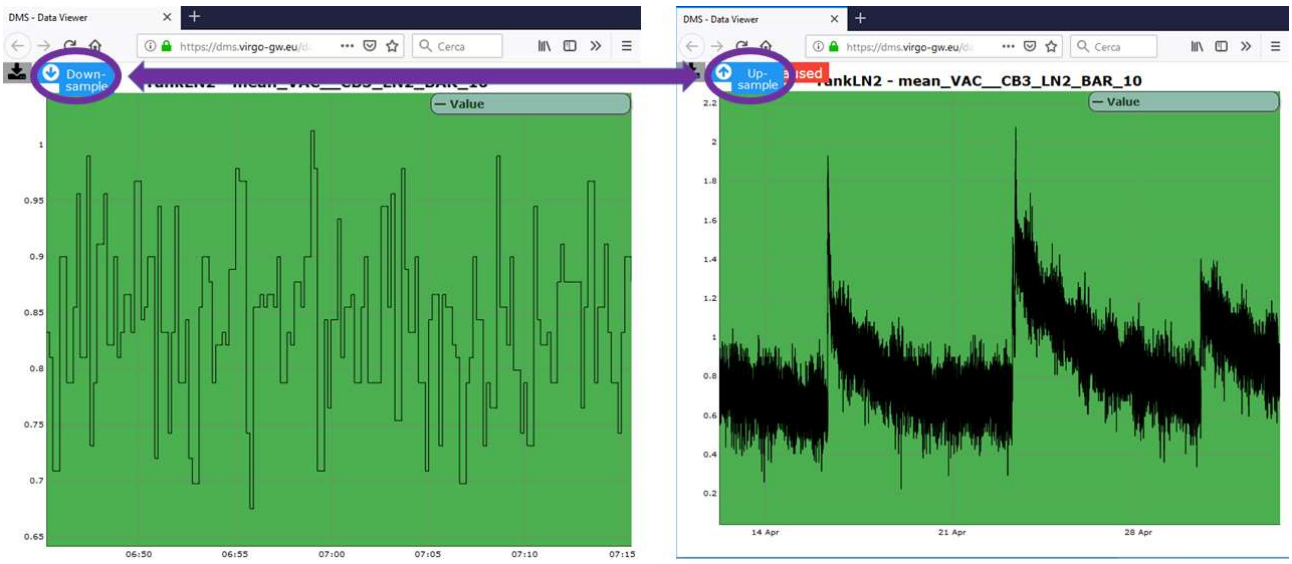


Figure 51 – how to switch from up-sampled/down-sampled data

10.4.3.3 How to zoom, un-zoom and pam a plot

The plot can be zoomed in both axis: X, Y.

- Open the dedicated window containing the plot.

To zoom the plot:

- click the left button of the mouse
- drag the mouse
- release the left button of the mouse

To un-zoom the plot:

- double-click of the left button of the mouse.

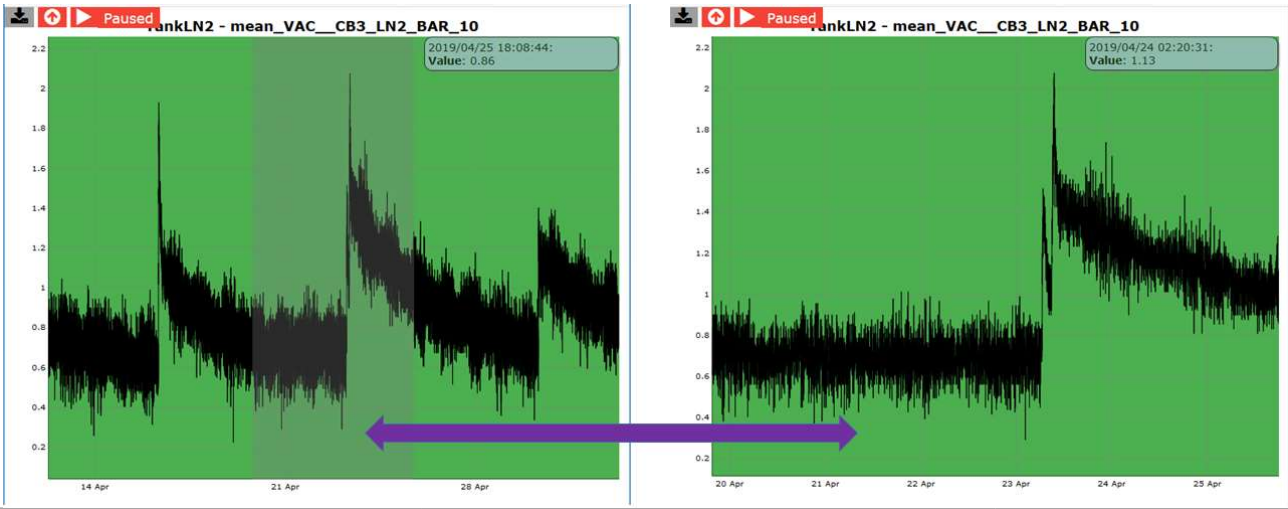


Figure 52 – how to zoom, un-zoom and pam a plot

To start pamming the plot:

- keep pressed shift+ctrl+left mouse button
- move the mouse

10.4.3.4 How to pause-play a plot

- Open the dedicated window containing the plot.

To pause a plot:

- Click on the icon ⏸

To play a plot:

- Click on the icon ▶

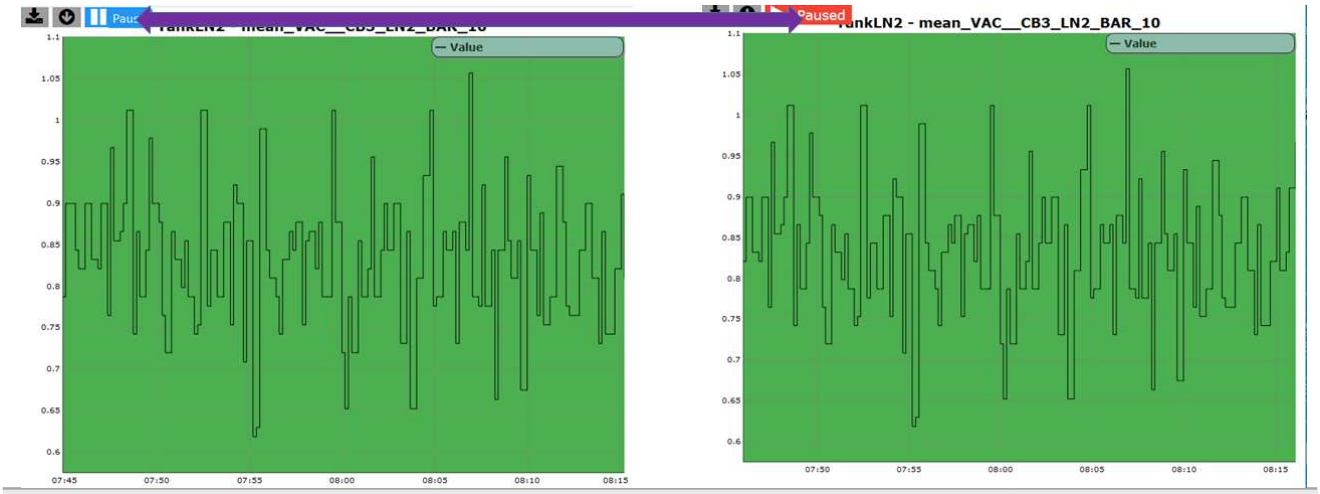


Figure 53 – how to pause-play a plot

10.4.3.5 How to download a plot as png

- Open the dedicated window containing the plot.

- Click on the icon

10.4.4 How to log into-out the system

To log into the system:

- Click on the icon on the top right side of the homepage to open a modal window to insert the user detail
- Enter your user details
- Click on the button Sign-in

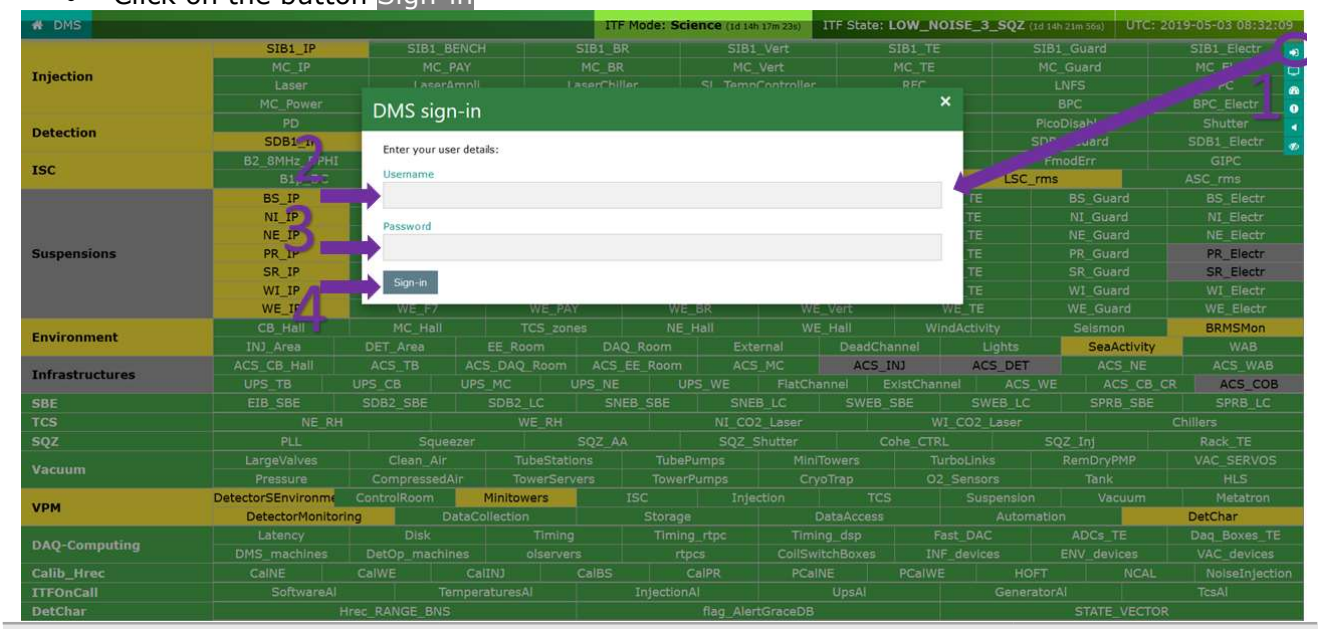


Figure 54 – how to log into-out the system

To know if the user is logged:

- Pass the mouse over the icons , on the top right side of the screen

To log out the system:

- Click the icon on the top right side of the screen

10.4.5 How to open the dashboard

- Click on the icon on the right side of the homepage;

10.4.6 How to open the alerts section

- Click on the icon on the right side of the homepage

10.4.7 How to open the shelving section

- Click on the icon on the right side of the homepage

10.4.8 How to open the muting section

- Click on the icon on the right side of the homepage

10.4.9 How to shelve-unshleve a condition flag

- Log into the system

To shelve a condition flag:

- Open the individual-flag-information section for the desired flag;
- Click on the icon to open the shelving section;
- Fill the form;
- Click on the button Shelve

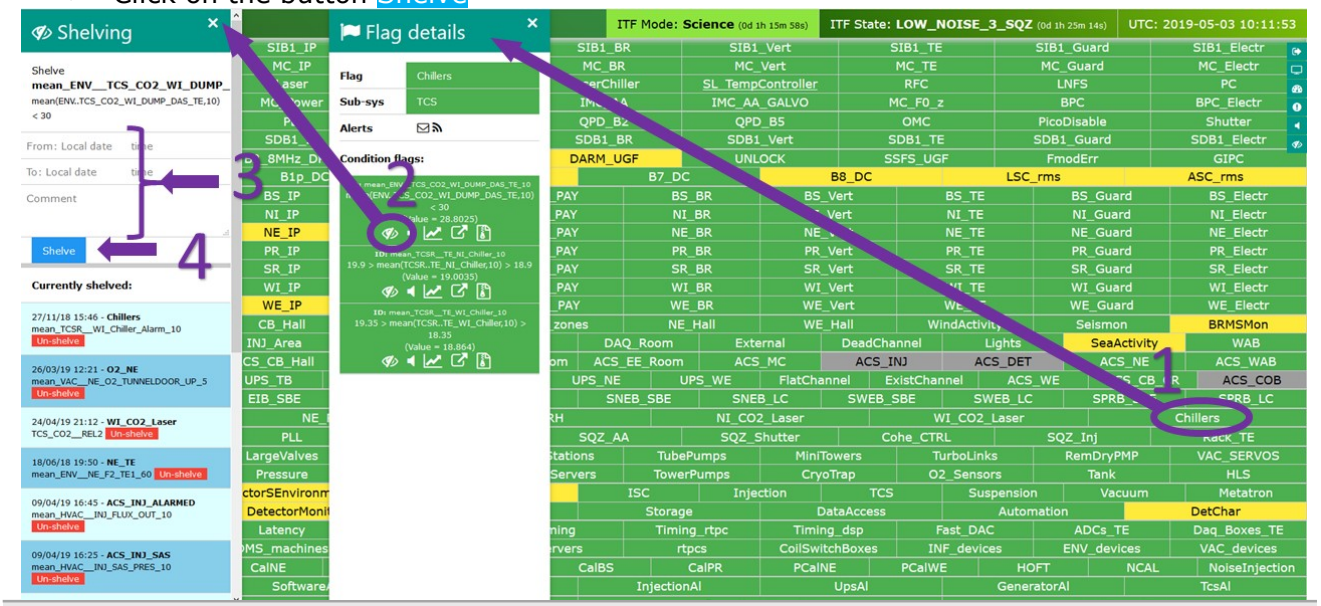


Figure 55 – how to shelve-unshleve a condition flag

To un-shelve a condition flag:

- Click on the icon on the right side of the homepage to open the shelving section;
- Search the condition flag you want to un-shelve;
- Click on the button Un-shelve.

Figure 56 – how to shelve-unshleve a condition flag

Another dedicated way to unshelve a condition-flag is to use the "DMS currently shelved condition flags", see [How to for the DMS currently shelved condition-flags.]

10.4.10 How to mute-unmute a condition flag

- Log into the system

To mute a condition flag:

- Open the individual-flag-information section for the desired flag;

To be noted that only alarmed flag can be muted.

- Click on the icon to open the mute section;
- Fill the form;
- Click on the button Mute

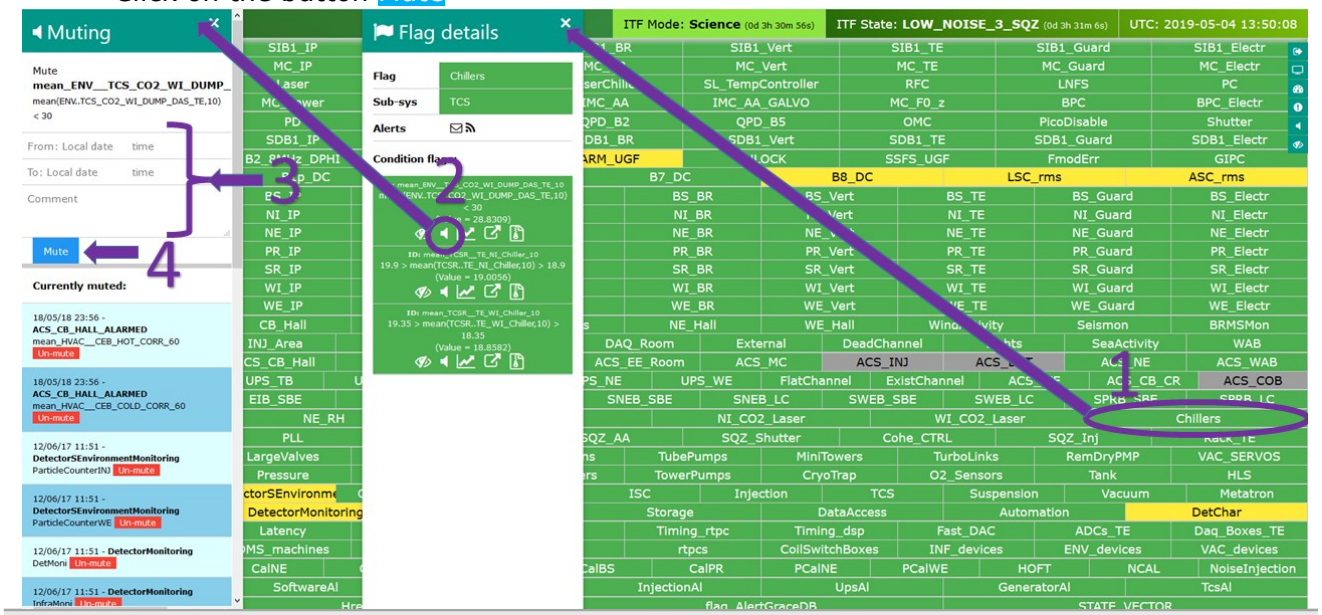


Figure 57 – how to mute-unmute a condition flag

To un-mute a condition flag:

- Click on the icon on the right side of the homepage to open the muting section;
- Search the condition flag you want to un-mute;
- Click on the button Un-mute.

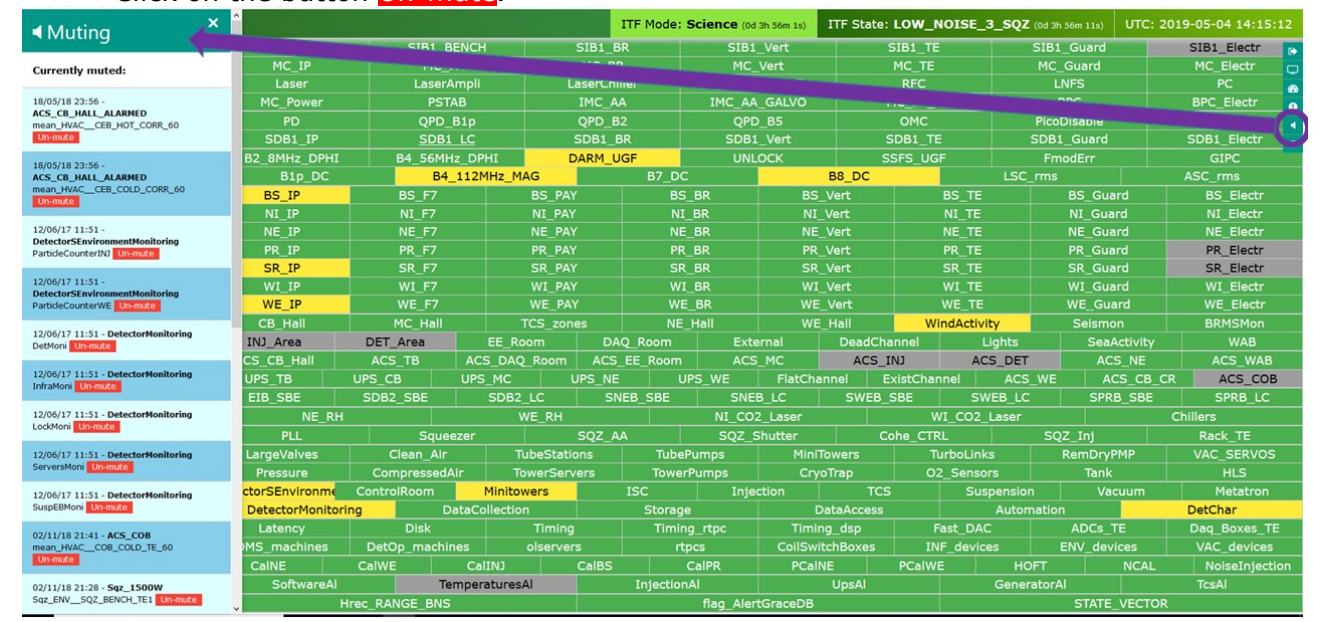


Figure 58 – how to mute-unmute a condition flag

10.5 How-to for DMS event monitor

10.5.1 How to open the DMS event monitor

The DMS event monitor can be open in two ways:

- By clicking on the link Go to Event Monitor from the dashboard;
- By clicking on the icon of the condition flag
- By clicking on Condition-Flag name in the table of the Events in the DMS Event monitor page.

The first option opens the homepage of the DMS event monitor with no filter criteria pre-configured.
The second and third options open the DMS event monitor with the events associated to the condition flag from which the application has been called.

10.5.2 How to know the last events in a specified time window

- Open the DMS event monitor from the dashboard
- Check all the events in the "Select event type" box inside the "Filter toolbar" section
- Check the option "Select all" in the "Select condition flag" box inside the "Filter toolbar" section
- Set the time window in the "Select time window" box inside the "Filter toolbar" section
- Click on the button Submit

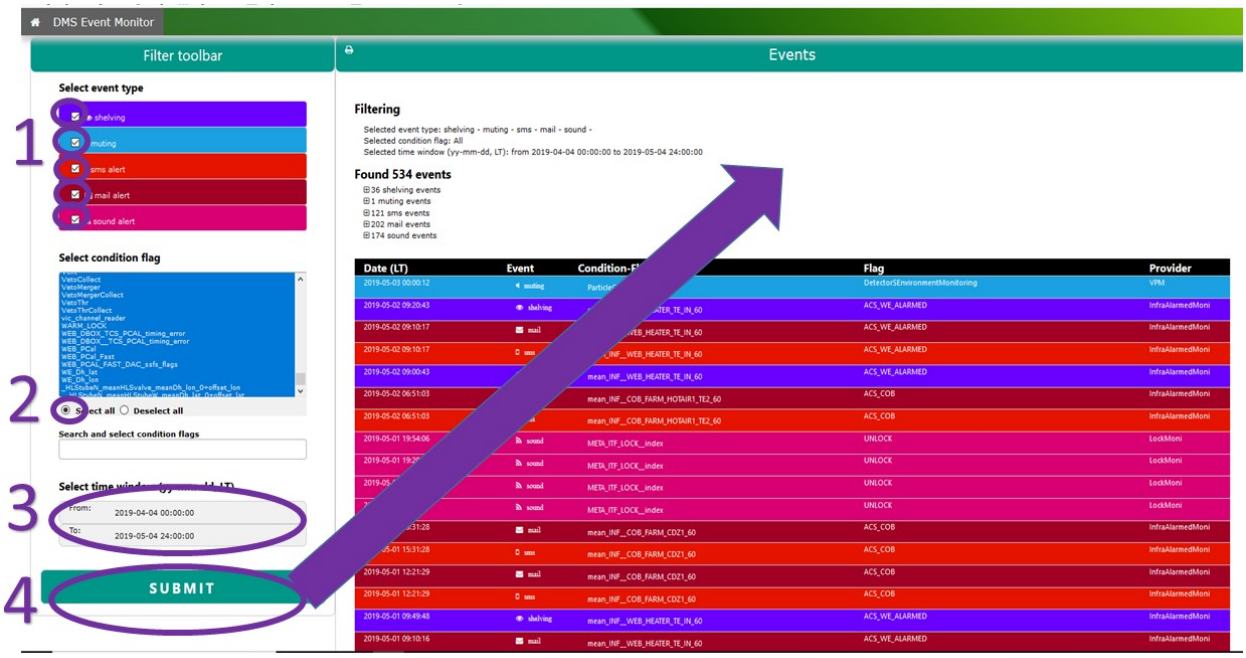


Figure 59 – how to know the last events in a specified time window

10.5.3 How to know a specific events for specific condition flags a specified time window

- Open the DMS event monitor from the dashboard
- Check only the events you are interested in the "Select event type" box inside the "Filter toolbar" section

- Select only the condition flags you are interested in the “Select condition flag” box inside the “Filter toolbar” section
- Set the time window in the “Select time window” box inside the “Filter toolbar” section
- Click on the button Submit

10.5.4 How to know details of the event

- Click on the icon of the event in column Events of the search result.

10.6 How-to for DMS playback

10.6.1 How to open the DMS playback

- By clicking on the icon on the right side of the DMS homepage.
- Coping the URLs provided by other API.

10.6.2 How to start-pause-stop the playback

- Open the DMS playback from DMS homepage

To start the playback:

- Set “UTC start” on the “Filter toolbar” section
- Set “Snapshot rate” on the “Filter toolbar” section
- Set “Snapshot pause” on the “Filter toolbar” section
- Click on the icon

To pause the playback:

- Click on the icon

To restart the playback:

- Click on the icon

To stop the playback:

- Click on the icon

Figure 60 – how to start-pause-stop the playback

10.6.3 How to download the snapshot JSON payload

- Stop or pause the playback
- Click the icon in the “Json File” box on the left section

10.6.4 How to upload the snapshot JSON payload

- Open the DMS playback from DMS homepage
- Click the icon in the "Json File" box on the left section

10.6.5 How to switch the display

- To switch from "Snapshot" to "info and debug" click the icon in the "Display" box on the left section
- To switch from "info and debug" to "Snapshot" click the icon in the "Display" box on the left section

10.7 How-to for DMS archive

10.7.1 How to open the DMS archive

- Open the individual-flag-information section for the desired condition-flag;
- Click on the icon to open the DSM archive for desired condition-flag

10.7.2 How to edit summary plots

- Open DSM archive for desired condition-flag
- Click on the icon on the top left side of the page
- Set the "Period"
- Set the "Y scale"
- Set the "Sampling"
- Click on the button SUBMIT

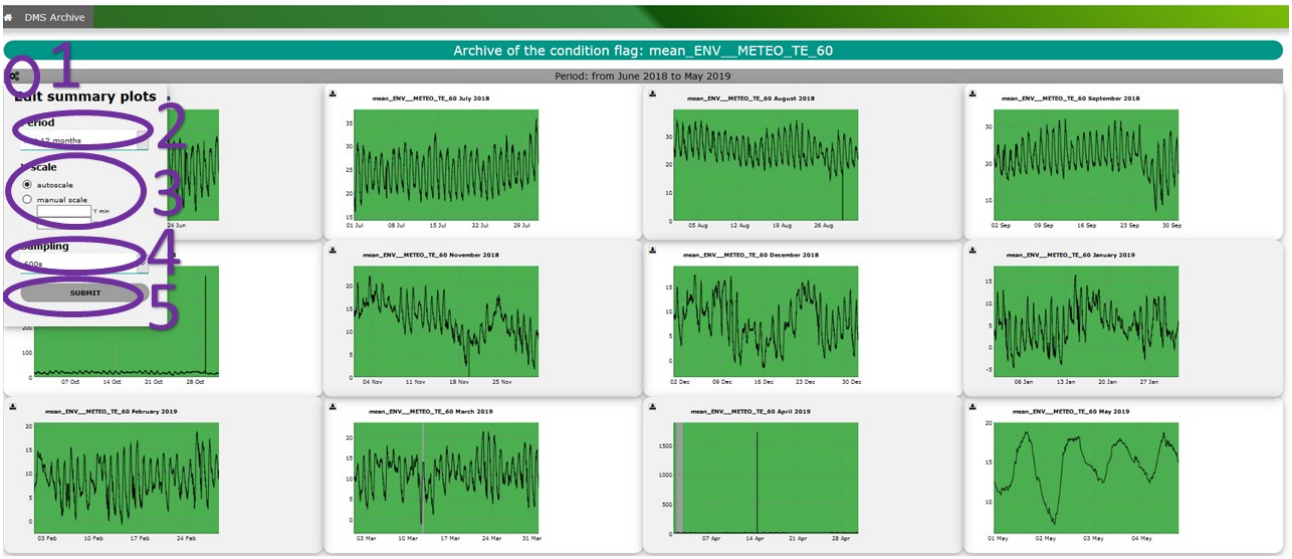


Figure 61 – how to edit summary plots

10.7.3 How to edit custom plots

- Open DSM archive for desired condition-flag
- Click on the icon on the bottom left side of the page
- Set the "Plot title"
- Set the "Y scale"
- Set the "Sampling"

- Set the “time window”
- Set the “plot dimension”
- Click on the button SUBMIT

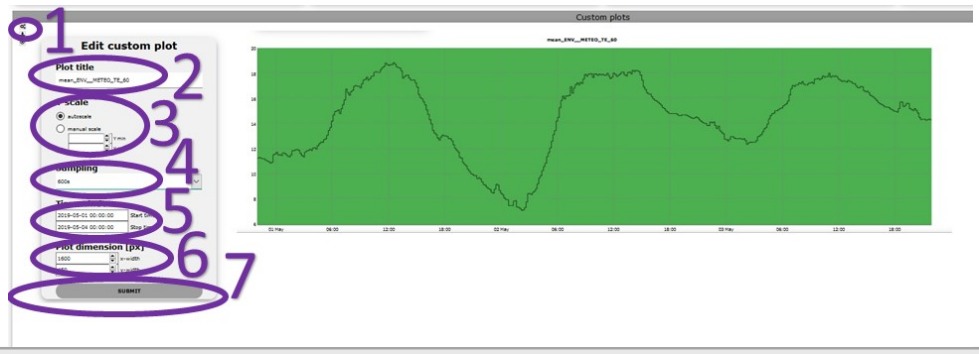


Figure 62 – how to edit custom plots

10.8 How to for the DMS currently shelved condition-flags.

10.8.1 How to open the DMS currently shelved condition-flags.

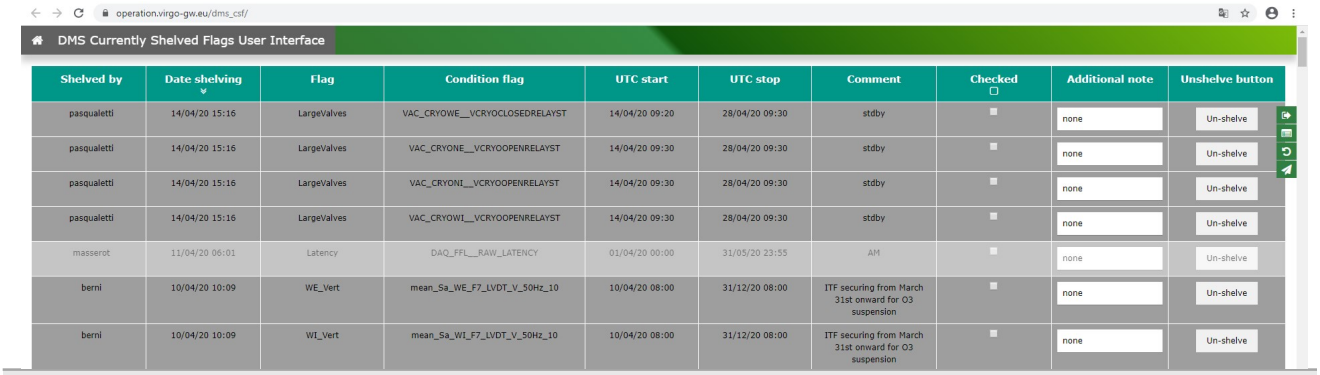
The DMS event monitor can be open by clicking on the link Go to shelved flags checklist from the dashboard.

10.8.2 How to order the results

The results can be order by:

- “Shelved by” ASC / DESC;
- “Date shelving” ASC / DESC;
- “Flag” ASC / DESC;
- “Condition-flag” ASC / DESC;
- “UTC start” ASC / DESC;
- “UTC stop” ASC / DESC;

This can be achieved by clicking on the related field on top of the table.



Shelved by	Date shelving	Flag	Condition flag	UTC start	UTC stop	Comment	Checked	Additional note	Unshelve button
pasqualetti	14/04/20 15:16	LargeValves	VAC_CRYOWE__VCRYOCLOSEDRELAYST	14/04/20 09:20	28/04/20 09:30	stdby		none	Un-shelve
pasqualetti	14/04/20 15:16	LargeValves	VAC_CRYONE__VCRYOOPENRELAYST	14/04/20 09:30	28/04/20 09:30	stdby		none	Un-shelve
pasqualetti	14/04/20 15:16	LargeValves	VAC_CRYONI__VCRYOOPENRELAYST	14/04/20 09:30	28/04/20 09:30	stdby		none	Un-shelve
pasqualetti	14/04/20 15:16	LargeValves	VAC_CRYOWI__VCRYOOPENRELAYST	14/04/20 09:30	28/04/20 09:30	stdby		none	Un-shelve
masserot	11/04/20 06:01	Latency	DAQ_FFL__RAW_LATENCY	01/04/20 00:00	31/05/20 23:55	AM		none	Un-shelve
berni	10/04/20 10:09	WE_Vert	mean_Sa_WE_F7_LVDT_V_50Hz_10	10/04/20 08:00	31/12/20 08:00	ITF securing from March 31st onward for O3 suspension		none	Un-shelve
berni	10/04/20 10:09	WI_Vert	mean_Sa_WI_F7_LVDT_V_50Hz_10	10/04/20 08:00	31/12/20 08:00	ITF securing from March 31st onward for O3 suspension		none	Un-shelve

Figure 63 – results orderd by “date shelving” DESC

10.8.3 How to log into the system

To log into the system:

- Click on the icon on the top right side of the homepage to open a modal window to insert the user detail
- Enter your user details
- Click on the button Sign-in

To know if the user is logged:

- Pass the mouse over the icons , on the top right side of the screen

To log out the system:

- Click the icon on the top right side of the screen

10.8.4 How to view the list of the reports

- Click on the icon on the right side of the homepage

10.8.5 How to build the report

- Log into the system
- Click on the icon on the right side of the homepage

10.8.6 How to un-shelve a flag

- Log into the system
- Click on the icon 

on the related condition flag

11 Documentation

- DMS - Snapshot of the system
- DMS - Presentation at Detchar meeting
- New Detector Monitoring System (DMS) Software Project
- New Detector Monitor System (DMS) User Requirements
- Meeting DAQ/Automation/DMS