CNRS
Centre National de la Recherche Scientifique

INFN
Istituto Nazionale di Fisica Nucleare

Detector Monitoring System for Advanced Virgo

F. Berni, V. Dattilo, F. Gherardini, G. Hemming, D. Verkindt

Last update: July 17th, 2012
VIR-0XXXA-12

1. Introduction

The motivation of this document is to define the requirements and the needed upgraded of the various tools used up to now , in order to provide the Detector Monitoring System (DMS) of the Advanced Virgo detector. It contains mainly two parts: one dedicated to the production of the Data Quality (DQ) flags, one dedicated to the tools taking the DQ flags as input and providing visual information in control room with associated tools like alarm, shelving, etc... For each part, we define the requirements, we describe the main features of the tools already existing, the improvements needed and the new tools we may need to develop.

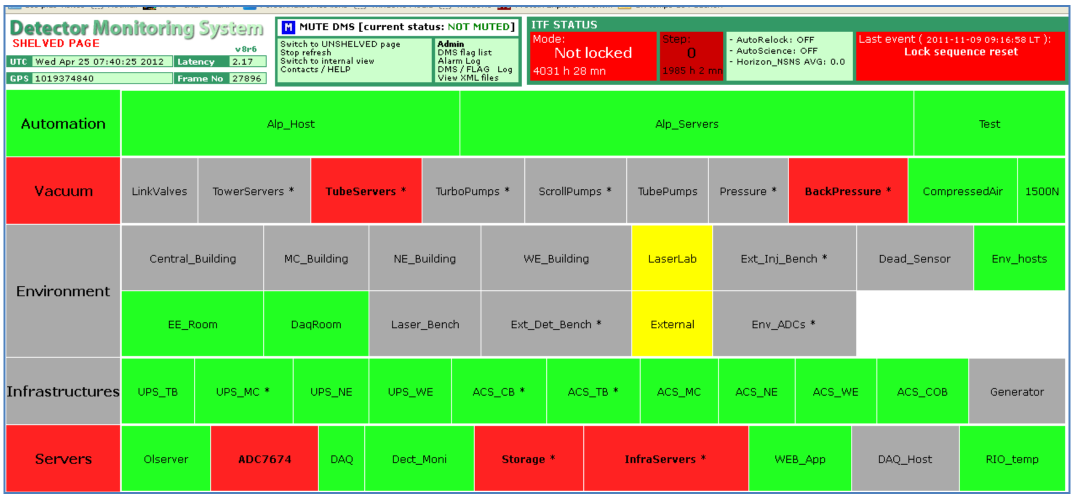
Since 2003, a set of tools has been set up and has been continuously improved for the DMS. Up to now, it is based mainly on two subsets:

- The Moni library (/virgoDev/Moni/v2r5p7) which provides an easy way to produce online the DQ flags through a set of keywords in a configuration file and the use of the Fd library (one of the main tools of the Virgo Data Acquisition System).
  Current documentation is available from
  https://wwwcascina.virgo.infn.it/cgi-bin/cvsweb/cvsweb.cgi/Moni/doc/

- The DMS php scripts which uses the XML files produced by the Moni server and, through a configuration file, organize the visualization of the DQ flags and the alarms attached to them.
  Current documentation is available from
  VIR-0191A-12 : The Detector Monitoring System (https://tds.ego-gw.it/ql/?c=9005)
  VIR-0192A-12 : The Detector Monitoring System : user manual (https://tds.ego-gw.it/ql/?c=9006)

  https://wwwcascina.virgo.infn.it/cgi-bin/cvsweb/cvsweb.cgi/MoniCfgUI
  https://wwwcascina.virgo.infn.it/cgi-bin/cvsweb/cvsweb.cgi/MoniSet

The information is provided by a set of Moni processes but also by other systems like Big Brother, IMMS, Eurotherm, UPS… The result is a central web page showing in the control room the status of most of the parts of the Virgo detector, with colored boxes associated to the DQ flags sent by the Moni processes or built by the DMS scripts. Each box can be clicked to show a more detailed information. An example is shown in the figure 1.



Detector Monitoring System
SHELVED PAGE
v8r6
UTC Wed Apr 25 07:40:25 2012 Latency 2.17
GPS 1019374840 Frame No 27896

MUTE DMS [current status: NOT MUTED]
Switch to UNSHELVED page
Stop refresh
Switch to internal view
Contacts / HELP

Admin
DMS flag list
Alarm Log
DMS / FLAG Log
View XML files

ITF STATUS
Mode: Not locked 4031 h 28 mn
Step: 0 1985 h 2 mn
- AutoRelock: OFF
- AutoScience: OFF
- Horizon_NSNS AVG: 0.0
Last event (2011-11-09 09:16:58 LT): Lock sequence reset

Automation: Alp_Host, Alp_Servers, Test
Vacuum: LinkValves, TowerServers *, TubeServers *, TurboPumps *, ScrollPumps *, TubePumps, Pressure *, BackPressure *, CompressedAir, 1500N
Environment: Central_Building, MC_Building, NE_Building, WE_Building, LaserLab, Ext_Inj_Bench *, Dead_Sensor, Env_hosts, EE_Room, DaqRoom, Laser_Bench, Ext_Det_Bench *, External, Env_ADCs *
Infrastructures: UPS_TB, UPS_MC *, UPS_NE, UPS_WE, ACS_CB *, ACS_TB *, ACS_MC, ACS_NE, ACS_WE, ACS_COB, Generator
Servers: Olserver, ADC7674, DAQ, Dect_Moni, Storage *, InfraServers *, WEB_App, DAQ_Host, RIO_temp

Fig. 1: Current typical display of the DMS in Virgo control room

2. Architecture

The online architecture will be similar to the one adopted up to now: one Moni Server per Virgo sub-system, each taking input frames from a shared memory connected to DAQ, each producing a XML file and sending in parallel the DQ flags (within frames) down to a frame collector FbmDMS linked to the DAQ. One or two olserver machines will be dedicated to those Moni servers.

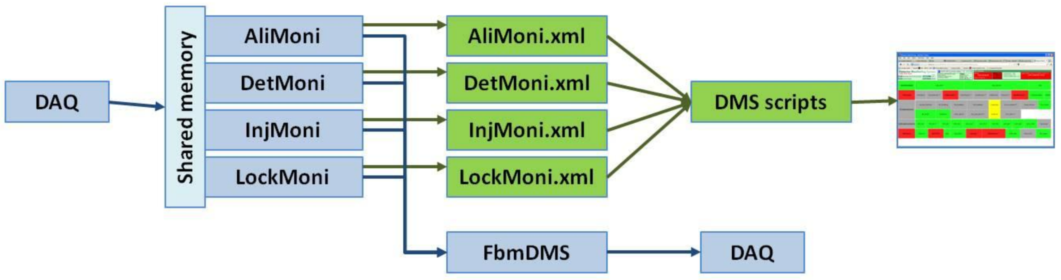


Fig.2 : Current architecture of the DMS (point of view of the Moni developer!)

The XML files contain the DQ flags values and some information about the DQ flags dependencies. Those files are read by a dedicated php script which uses in parallel a configuration which defines the organization of the flags dependencies, the way the flags will be shown on the web page, the vetoes or alert procedures associated to each flag…

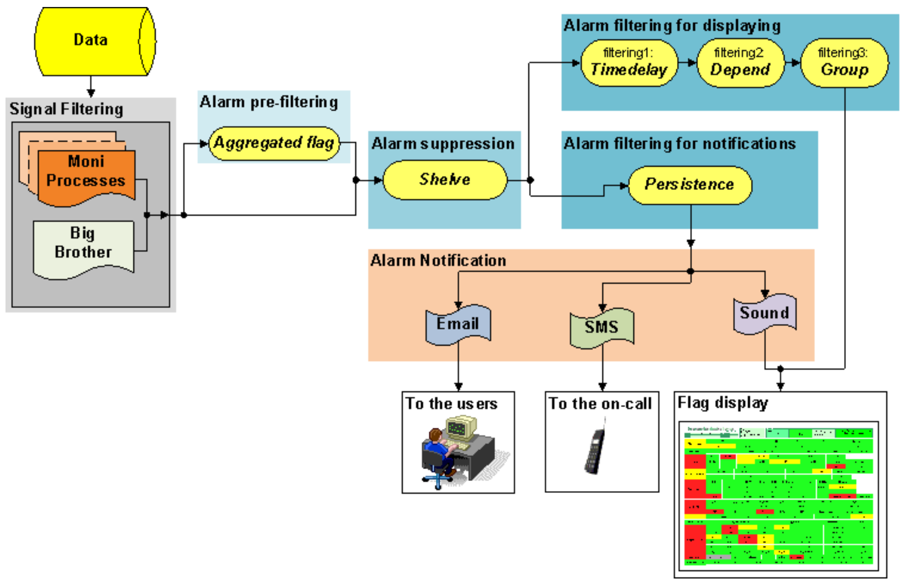


Fig. 3 : Current architecture of the DMS (point of view of the DMS scripts developers!)

2. The Moni library

This software package has been developed by D. Verkindt. The version used up to now is described in the document VIR-074A-08. A large upgrade is foreseen for this software and will follow a new set of requirements. Both requirements and upgrades are described below.

2.1 Requirements

The Moni Library and its associated executable called MoniServer will provide DQ flags based on a few set of functions used with one or several data channels. Those DQ flags will be produced according the following rules:

- Each DQ flag will have only 3 possible values: -1 (no information), 0 (no problem detected) or 1 (check or action is needed). Other negative or positive values can be foreseen for DMS purpose but will be considered equivalent to -1 and +1 for any data analysis purpose.
- Each DQ flag can be either the result of a combination of functions computed on one or several data channels or the result of the combination of several other DQ flags.
- Each DQ flag value will be sent to DAQ for raw data and trend data storage
- Each DQ flag value will be sent to the main online DQ chain so that it can be used also by online data analysis pipelines
- Each DQ flag value will be made available to the DMS scripts using XML files, with a format to be defined (should be quite similar to what is used now).
- The latency of the XML files production by the Moni server should be less than 30 sec. This means that XML files should be updated every 25 sec maximum. Current latency is about 10s. This proposed value seems too high to us. Better to discuss it together at our next meeting. I put 30s because I have never seen a problem raised by the DMS where operator action should be prompt with less than 1mn. But current latency of 10s is ok and we could take it as maximum value too.
- We experienced some difficult to retrieve the flag value from the value stored in the trenddata: indeed, every time it is necessary to know the cfg file and then decodify form decimal to binary. An improvement to this procedure (for instance including some functionality at level of DataDisplay) would be useful.

There is already in the dataDisplay the possibility to visualize the bits of the values. This is an option in the time plot. Anyway, with the new Moni, there is no more bit coding. Each flag will have only three values (+1, 0 or -1).

- Sometimes we reach the limit of 32 subflags while defining a flag. Any chance to pass to a 64 bit architecture?
  With the new Moni, there is no more bit coding. Each flag will have only three values (+1, 0 or -1). Thus no more 32 subflags limitation.

The Moni package will be able to read data channels from input frames and to output DQ flag values stored in FrSerData structures of output frames. The only other output will be XML files produced for the DMS scripts.

Each flag will be defined by a set of conditions. Each condition will be made by applying one or several operations on one or several data channels.

Output DQ values will be computed for each second. No sub-second result will be available. Values computed every n seconds with n>1 will be oversampled at 1 Hz.

2.2 Upgrades

The Moni package will be composed of the Moni library, the MoniServer executable and a set of test scripts. It will use the Fd library (that means Cm, CSet, Cfg and Fr libraries). ~~We could even think about a Moni package containing all the software needed to produce the DMS web page (so including also the php scripts, etc…).~~

We don't't' see any advantages in doing this, we would prefer to keep separated the two packages;
OK. It was just a proposition to regroup in a single package "DMS" all the scripts and codes used to provide DMS information.

The output XML files will have the format described below:

```
<Prefix defined by the keyword DMS_NAME in the config file>
<qcheader>
<qctitle>generic title formed with the prefix</qctitle>
<qcflag_value>int value of the flag defined by the keyword DMS_FLAG</qcflag_value>
<qc_oplink>html link containing explanations for operators. Based on prefix</qc_oplink>
<qcitfstate_value>value of the lock step when this xml file is produced</qcitfstate_value>
 <qcframe_number>number of the last data frame read</qcframe_number>
 <qcframe_time>UTC date of the frame</qcframe_time>
 <qcframe_gps>GPS time of the frame</qcframe_gps>
 <qcframe_latency>latency of the frame</qcframe_latency>
</qcheader>

<qcflag>
 <flag_name>Name of the flag defined by the keyword DMS_MONITOR </flag_name>
 <flag_value>Value of the flag</flag_value>
</qcflag>

<qcflag_condition>
 <flag_chname>Channel's name</flag_chname>
 <flag_name>Condition used to produce the flag</flag_name>
 <flag_section>Flag's name</flag_section>
 <flag_value>Flag's value</flag_value>
 <flag_status>NO MORE USED</flag_status>
 <flag_chvalue>Value computed in the condition</flag_chvalue>
 <flag_trigvalue>NO MORE USED</flag_trigvalue>
 <flag_time>Time elapsed since flag’s value last change</flag_time> (REPLACE flag_nabsent)
 <flag_ncorrupt>NO MORE USED</flag_ncorrupt>
 <flag_comment>Comment associated to flag’s value equal to 1</flag_comment>
</qcflag_condition>
```

Here is an example of XML file:

```
<?php
$xmlstr = <<<XML
<?xml version='1.0' standalone='yes'?>
<Qc_Environment>

<qcheader>
 <qctitle>Qc_Environment quality flags</qctitle>
 <qcflag_value>276825344</qcflag_value>
 <qcflag_color>#FF2222</qcflag_color>  ← not used anymore
 <qc_oplink>Qc_Environment_OP.html</qc_oplink>
 <qcitfstate_color>#EEFFEE</qcitfstate_color>  ← not used anymore
 <qcitfstate_value>-1</qcitfstate_value>
 <qcframe_number>30074</qcframe_number>
 <qcframe_time>Mon Apr 30 15:42:35 2012</qcframe_time>
 <qcframe_gps>1019835770</qcframe_gps>
 <qcframe_latency>2.17</qcframe_latency>
 <qcframe_overallflag>0x10800500</qcframe_overallflag>  ← not used anymore
</qcheader>

<qcsubflag>  ← becomes qcflag
 <flag_name>Ce_Building</flag_name>
 <flag_level>0</flag_level>  ← not used anymore
 <flag_section>Qc_Environment</flag_section>
 <flag_value>0</flag_value>
 <flag_status>0</flag_status>  ← not used anymore
</qcsubflag>

<qcsubflag>
 <flag_name>MC_Building</flag_name>
 <flag_level>0</flag_level>
 <flag_section>Qc_Environment</flag_section>
 <flag_value>0</flag_value>
 <flag_status>0</flag_status>
</qcsubflag>

<qcalgo_subflag>  ← becomes qcflag_condition
 <flag_chname>Em_ACTCSWI</flag_chname>
 <flag_name>"rms(Em_ACTCSWI,10) &lt; 0.75"</flag_name>
 <flag_section>WI_TCS_Bench</flag_section>
 <flag_value>-1</flag_value>
 <flag_chvalue>0</flag_chvalue>
 <flag_time>1292257</flag_time>
 <flag_comment>"Acoustic_noise_too_high"</flag_comment>
</qcalgo_subflag>
```

```
</Qc_Environment>
XML;
?>
```

Each flag will be defined by a set of conditions. Each condition will be made by applying one or several operations on one or several data channels. The following operations will be possible:

- Exist(chname): just put a flag to +1 if the channel is missing
- Mean(chname,duration) : compute the mean of chname over duration seconds
- Min(chname,duration) : compute the min of chname over duration seconds
- Max(chname,duration) : compute the max of chname over duration seconds
- Rms(chname,duration) : compute the rms of chname over duration seconds
- Delta(chname,duration) : compute the max of |s(i+1)-s(i)| of chname over duration seconds
- Brms(chname,duration,N,fmin,fmax) : compute the bandrms of chname from fmin to fmax, using FFTs of duration seconds averaged N times.

Conditions used for one flag can be a AND of conditions. For instance:
"18<mean(TELSLB02,30)<28 AND rms(AC_LB,10)<0.75 AND brms(SEBDCE07,50,4,0.2,1.)<1e-3"

The configuration file will contain a set of keywords allowing to define the flags and the parameters used to define them. The following keywords will be available:

- DMS_XML Name_of_the_xml_file Update_period
- DMS_NAME Prefix_of_the_flags
- DMS_FLAG Name_of_output_flag "Input_flag1 Input_flag2 …."
- DMS_REALUNITS "List of channels for which calibrated data will be used"
- DMS_MONITOR Lock_steps_concerned Flag_name CondNbr "Conditions" "Comment"

Here is an example of configuration file:

```
# Produce a xml file updated every 5s.
DMS_XML /opt/MonitoringWeb/olweb/html/itf/qcmoni/V5/data/QcEnvironmentData.php 5

# Set the prefix of the flags names of this Moni server
DMS_NAME Qc_Environment

#Set one overall flag as a OR of other flags
DMS_FLAG Overall  "Ce_Building  MC_Building  NE_Building   WE_Building"

# Use real units of all the signals monitored
DMS_REALUNITS  *

#Set the conditions whose OR determine the value of the flag Ce_Building
#As soon as one condition is not fulfilled, Ce_Building is set to +1.
#Third parameter after the keyword is a numerotation of the condition.
#To each condition is associated a flag: for instance Ce_Building_2 for the condition 2.
#First parameter after the keyword DMS_MONITOR is the itf_status range for which the condition is tested.
DMS_MONITOR * Ce_Building  1 "18.9<mean(Em..TEBDCE01,10)<20.9" "Temperature out of range"
DMS_MONITOR 1-11 Ce_Building 2  "delta(Em..TEBDCE01,300)>1e-8"  "CB ETN1 Box2 signals could be flat"
DMS_MONITOR * Ce_Building  3 "rms(Em_ACBDCE01,10)<0.3"   "Acoustic noise too high"

#Set the conditions whose OR determine the value of the flag MC_Building
DMS_MONITOR 3-6  MC_Building 1 "brms(Em_SEBDMC01,50,4,0.2,1.)<1e-3" "Seismic noise too high"
DMS_MONITOR 7-8  MC_Building  2 "brms(Em_SEBDMC01,50,4,1,4)<1e-3" "Seismic noise too high"
DMS_MONITOR * MC_Building   3 "21<mean(Em..TEBDMC01,10)<24"  "Temperature out of range"
```

~~The use of the keyword DMS_FLAG should be restricted (or even removed) so that only first level flags are produced by the Moni processes. Then, any combination (aggregated flags or groups) would be done only by the DMS scripts.~~

It is not completely clear to us; it could make sense but we need a clarification because it could imply a complete reshuffle of the PHP script.
The PHP script already regroup some flags with the "Group" feature. I propose that Moni does not regroup any flag and that the PHP script does all the regrouping job. The aim is to have only one place where the regrouping of flags is done. The Moni process would only do a first level of aggregation: for instance the flag Ce_Building will be a OR of the flags Ce_Building_1, Ce_Building_2, Ce_Building_3.

3. DMS scripts

This software package has been developed initially by Gary Hemming with the coordination work of Vincenzo Dattilo and is now under development with F. Berni and F. Gherardini. The version used up to now is described in the documents:
VIR-0201A-12 : Detector Monitoring Centralized Displaying (https://tds.ego-gw.it/ql/?c=9015)
VIR-0191A-12 : The Detector Monitoring System (https://tds.ego-gw.it/ql/?c=9005)
VIR-0192A-12 : The Detector Monitoring System : user manual (https://tds.ego-gw.it/ql/?c=9006)

Some upgrade is foreseen for this software and will follow a new set of requirements. Both requirements and upgrades are described below.

3.1 Requirements

The web page shown in control room will be managed by a set of scripts taking as input the XML files provided by the MoniServers, BigBrother, Mailbox Polling Agent, ALP.

The states and associated colors will be:

- DQ flag not available → grey
- DQ flag on (one of the condition not fulfilled) → red
- DQ flag off (all conditions fulfilled) → green
- Do we really need the yellow color associated to a pre-alarm set of conditions? yes we need: although it is not strictly necessary, it is useful.
- Do we really need the light red color associated to "problem induced by an other item"? yes we need, it is useful for making easier the diagnosis.

- Do we really need the dark grey associated to corrupted channel used in computing the flag? yes we need, it is useful for making easier the diagnosis.
-

Default state will be "no information" and corresponding color will be grey.

~~The web page will show only the second level flags which are an OR of the first level flags provided by the Moni processes. Clicking one of the second level flags will show the first level flags associated to it. Clicking one of the first level flags will show the conditions associated to it, plus the current values used in this condition and a link to a dynamical plot showing the channels used in this condition.~~

~~For any display level (aggregated flags, first level flags or conditions), all the flags or conditions should be shown, whatever their state.~~

It is not completely clear to us, better to discuss on it all together around a table
The idea is that Moni processes create level 0 flags (Ce_Building_1, Ce_Building_2, Ce_Building_3) and level 1 flags (Ce_Building). Then, the DMS script regroup Ce_Building and Mc_Building to create a level 2 flag (named by default Qc_Environment). Then, what is shown in control room is only the set of level2 flags. It allows to have a less crowded web page in control room. As soon as one those level 2 flags is red, the operator can click on it and see which level 1 flags are red. Clicking on the level 1 red flags, the operator can see the level 0 flags which are red.

3.2 Upgrades

~~The php scripts may be updated to deal with the DQ flags permitted values (-1,0,1).~~
~~The "Timedelay" and "Persistence" features should be mixed.~~
~~The "Group" feature will be used to produce what we call above "the second level flags"~~
~~And should not be different from the "Aggregated flags".~~
~~The "Shelve" feature will be indicated for each flag in the web page. It is a property of each flag.~~
~~The "Depend" feature will be no more used, except if really needed, if we decide for a new ServersMoni process (in charge of monitoring the servers themselves) using direct information from the servers instead of the existence of a channel in the data.~~

Concerning your proposals, we believe it is better to discuss on them all together around a table.

-

List of upgrades that we have in mind so far (actually we are still collecting desiderata/ideas for the users):

- Final debug and improvement of the plot zoom
- Improvement of performance by asking for a better machine: faster, higher HARD DISK, etc…
- Replicate the db, in read-only mode on a public machine (i.e. pub3) to avoid the problem of the locked tables, to make the system usable in the CDB application, …
- Ask a backup machine to implement a mechanism of master/slave and make the system more reliable;
- Interaction with FoxBox (TBD) (for Didier: FoxBox is our GSM server used to send SMS.

Did you collect any other /desiderata/ideas from the users?
What about the other inputs than Moni (Big Brother, IMMS, Eurotherm, UPS, Mailbox Polling Agent, ALP…)?
I do not understand the difference between the "TimeDelay" and "Persistence" features.
Concerning the "Depend" feature and the way ServersMoni is working, I think it desserves a specific discussion, whose basis is that we could remove the Depend features from the configuration of ServersMoni and develop a new ServersMoni which takes its information directly from the server (through a Cm request) or from a logfile written by the server (if the server is not Cm connected).