GIF++ DAQ based on DIM

L.Mirabito

1 Slow Control

1.1 Processes configuration

The processes are driven via a unique daemon running on each connected PC called the job control. It's a dim server that has to be started once on each considered computer. If one computer crash it's the only needed process to restart.

1.1.1 DimJobControl

On each slow control computer the process is started by

```
sudo /etc/init.d/dimjcd start
```

The usual stop and status options are available.

1.1.2 Controlling slow control process

A DimJobInterface program allows to start and stop process on all connected computer via the DimJobControl daemon. It is based ona JSON description of the needed process.

Here is the one used at GIF++ in August 2015. The structure is simple and self-describing.

```
{
    "hosts" :
    {
        "lyosdhcal11":[
            {
            "Name":"CAENSERVER",
                "Arguments" : ["-f /data/NAS/toto.slcio","-c MONCOLLECTOR"],
            "Environnement" : [
                "DIM_DNS_NODE=lyosdhcal11",
                "SLOWDB=acqilc/RPC_2008@lyosdhcal11:GIFPP2015;GIFLYON2",
                "LD_LIBRARY_PATH=/usr/lib:/usr/local/lib:/opt/dhcal/lib:/opt/dhcal/DQM4HEP/l
            ],
            "Program" : "/opt/dhcal/bin/dimcaenhv"
```

```
            }
        ],
        "lyoilcrpi24":[
            {
            "Name":"BMP183SERVER",
                "Arguments" : ["-f /data/NAS/toto.slcio","-c MONCOLLECTOR"],
            "Environnement" : [
                "DIM_DNS_NODE=lyosdhcal11",
                "LD_LIBRARY_PATH=/usr/lib:/usr/local/lib:/opt/dhcal/lib:/opt/dhcal/DQM4HEP/l
            ],
            "Program" : "/opt/dhcal/bin/dimbmp183"
            }
        ],
// Test avec 2 programmes
        "lyoilcrpi02":[
            // GPIO
            {
                "Name":"GPIOSERVER",
                "Arguments" : [],
                "Environnement" : [
                    "DIM_DNS_NODE=lyosdhcal11",
                    "LD_LIBRARY_PATH=/usr/lib:/usr/local/lib:/opt/dhcal/lib:/opt/dhcal/DQM4H
                ],
                "Program" : "/opt/dhcal/bin/dimgpio"
            },
            // DS1820
            {
                "Name":"DS1820SERVER",
                "Arguments" : [],
                "Environnement" : [
                    "DIM_DNS_NODE=lyosdhcal11",
                    "LD_LIBRARY_PATH=/usr/lib:/usr/local/lib:/opt/dhcal/lib:/opt/dhcal/DQM4H
                ],
                "Program" : "/opt/dhcal/bin/dimds1820"
            }
        ]
    }
}
```

The DimJobInterface belongs to the python package Ldimjc. It has few methods to start, stop and list processes belonging to the description file.

1.1.3 Usage

On lyosdhcal11 with DIM_DNS_NODE correctly defined the DimJobInterface is instantiated in python with

```
>>>import Ldimjc
>>> dsc=Ldimjc.DimJobInterface()
```

Then a set of commands allows to control the process:

- **import Ldimjc** : imports the python package
- **dsc=Ldimjc.DimJobInterface()**: It creates the interface
- **dsc.loadJSON("slow_conf.json")**: It loads and parses the given files, scan the DNS to have the list of DimJobControl daemon
- **dsc.startJobs("lyoilcrpi02")**: it starts the jobs described in the configuration on the given host. If the host is "ALL", theb all the jobs desribed in the file are started.
- **dsc.clearHostJobs("lyoilcrpi02"**: it kills all the jobs started on lyoilcrpi02 or on any host given
- **dsc.clearAllJobs()**: it kills all jobs on all hosts
- **dsc.status()**: it queries the status of all the jobs on all hosts.
- **dsc.List()**: it prints the last queried status
- **dsc.killJob("lyoilcrpi02",2934,9)**: it kills the job with PID 2934 on host lyoilcrpi02 with signal 9. The host and PID is found with the previous List command
- **dsc.restartJob("lyoilcrpi02","DS1820SERVER",2934, 9)**: it sends a killJob message to PID 2934 on lyoilcrpi02 with signal 9 and restart the process described on this host as DS1820SERVER in the configuration file.

1.2 Slow control database

1.2.1 Construction tables

The geometry and connection of a given setup is described with the following tables. HVRACK, DIFs,ASUs and CHAMBERs are registered in separated tables, then DETECTORs are assembly of those devices with SETUP reference. Giving a SETUP name leads to a unique list of detectors and positions

```
mysql> desc SETUP;
+-------+-------------+------+-----+-------------------+-----------------------------+
| Field | Type        | Null | Key | Default           | Extra                       |
```

```
+-------+-------------+------+-----+------------------+-----------------------------+
| IDX   | int(11)     | NO   | PRI | NULL             | auto_increment              |
| NAME  | varchar(50) | YES  |     | NONE             |                             |
| START | timestamp   | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
| VALID | tinyint(1)  | YES  |     | 1                |                             |
+-------+-------------+------+-----+------------------+-----------------------------+
mysql> desc HVRACK;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| IDX      | int(11)     | NO   | PRI | NULL    | auto_increment |
| HVTYPE   | int(11)     | YES  |     | 0       |                |
| HOSTNAME | varchar(50) | YES  |     | NONE    |                |
| USERNAME | varchar(50) | YES  |     | NONE    |                |
| PWD      | varchar(50) | YES  |     | NONE    |                |
| VALID    | tinyint(1)  | YES  |     | 1       |                |
+----------+-------------+------+-----+---------+----------------+
mysql> desc DIF;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| IDX      | int(11)     | NO   | PRI | NULL    | auto_increment |
| NUMBER   | int(11)     | YES  |     | 0       |                |
| FIRMWARE | varchar(50) | YES  |     | NONE    |                |
| VALID    | tinyint(1)  | YES  |     | 1       |                |
+----------+-------------+------+-----+---------+----------------+
mysql> desc ASU;
+--------+-------------+------+-----+---------+----------------+
| Field  | Type        | Null | Key | Default | Extra          |
+--------+-------------+------+-----+---------+----------------+
| IDX    | int(11)     | NO   | PRI | NULL    | auto_increment |
| NAME   | varchar(50) | YES  |     | NONE    |                |
| TYPE   | int(11)     | YES  |     | 0       |                |
| NASICS | int(11)     | YES  |     | 0       |                |
| VALID  | tinyint(1)  | YES  |     | 1       |                |
+--------+-------------+------+-----+---------+----------------+
mysql> desc CHAMBER;
+-------+-------------+------+-----+------------------+-----------------------------+
| Field | Type        | Null | Key | Default          | Extra                       |
+-------+-------------+------+-----+------------------+-----------------------------+
| IDX   | int(11)     | NO   | PRI | NULL             | auto_increment              |
| NAME  | varchar(50) | YES  |     | NONE             |                             |
| TYPE  | int(11)     | YES  |     | 0                |                             |
| X0    | double      | YES  |     | 0                |                             |
| X1    | double      | YES  |     | 0                |                             |
| Y0    | double      | YES  |     | 0                |                             |
```

```
| Y1    | double      | YES  |     | 0                 |                               |
| START | timestamp   | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
| VALID | tinyint(1)  | YES  |     | 1                 |                               |
+-------+-------------+------+-----+-------------------+-------------------------------+
```

```
mysql> desc DETECTOR;
+-----------+------------+------+-----+-------------------+-------------------------------+
| Field     | Type       | Null | Key | Default           | Extra                         |
+-----------+------------+------+-----+-------------------+-------------------------------+
| IDX       | int(11)    | NO   | PRI | NULL              | auto_increment                |
| CHAMBERID | int(11)    | YES  |     | 0                 |                               |
| SETUPID   | int(11)    | YES  |     | 0                 |                               |
| HVRACKID  | int(11)    | YES  |     | 0                 |                               |
| HVCHANNEL | int(11)    | YES  |     | 0                 |                               |
| DIFID0    | int(11)    | YES  |     | 0                 |                               |
| DIFID1    | int(11)    | YES  |     | 0                 |                               |
| DIFID2    | int(11)    | YES  |     | 0                 |                               |
| ASUID00   | int(11)    | YES  |     | 0                 |                               |
| ASUID10   | int(11)    | YES  |     | 0                 |                               |
| ASUID20   | int(11)    | YES  |     | 0                 |                               |
| ASUID01   | int(11)    | YES  |     | 0                 |                               |
| ASUID11   | int(11)    | YES  |     | 0                 |                               |
| ASUID21   | int(11)    | YES  |     | 0                 |                               |
| PREF      | double     | YES  |     | 0                 |                               |
| TREF      | double     | YES  |     | 0                 |                               |
| VREF      | double     | YES  |     | 0                 |                               |
| Z0        | double     | YES  |     | 0                 |                               |
| Z1        | double     | YES  |     | 0                 |                               |
| HEURE     | timestamp  | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
| VALID     | tinyint(1) | YES  |     | 1                 |                               |
+-----------+------------+------+-----+-------------------+-------------------------------+
```

One has to notice in the DETECTOR table, 3 files PREF,TREF,VREF where the reference pressure, temperature and voltage are stored. They will be used in the regulation procedure.

1.2.2 Monitoring tables

Each monitoring devices has its own monitoring table where measurements are stored with a time stamp:

```
mysql> desc HVMON;
+-----------+------------+------+-----+-------------------+-------------------------------+
| Field     | Type       | Null | Key | Default           | Extra                         |
+-----------+------------+------+-----+-------------------+-------------------------------+
| IDX       | int(11)    | NO   | PRI | NULL              | auto_increment                |
```

```
| HVRACKID  | int(11)    | YES  |     | 0                 |                               |
| HVCHANNEL | int(11)    | YES  |     | 0                 |                               |
| VSET      | double     | YES  |     | 0                 |                               |
| VMON      | double     | YES  |     | 0                 |                               |
| IMON      | double     | YES  |     | 0                 |                               |
| STATUS    | int(11)    | YES  |     | 0                 |                               |
| HEURE     | timestamp  | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
| VALID     | tinyint(1) | YES  |     | 1                 |                               |
+-----------+------------+------+-----+-------------------+-------------------------------+
mysql> desc DS1820MON;
+-------+------------+------+-----+-------------------+-----------------------------+
| Field | Type       | Null | Key | Default           | Extra                       |
+-------+------------+------+-----+-------------------+-----------------------------+
| IDX   | int(11)    | NO   | PRI | NULL              | auto_increment              |
| TIN   | double     | YES  |     | 0                 |                             |
| TOUT  | double     | YES  |     | 0                 |                             |
| HEURE | timestamp  | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
| VALID | tinyint(1) | YES  |     | 1                 |                             |
+-------+------------+------+-----+-------------------+-----------------------------+
mysql> desc PTMON;
+-------------+------------+------+-----+-------------------+-----------------------------+
| Field       | Type       | Null | Key | Default           | Extra                       |
+-------------+------------+------+-----+-------------------+-----------------------------+
| IDX         | int(11)    | NO   | PRI | NULL              | auto_increment              |
| PRESSURE    | double     | YES  |     | 0                 |                             |
| TEMPERATURE | double     | YES  |     | 0                 |                             |
| HEURE       | timestamp  | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
| VALID       | tinyint(1) | YES  |     | 1                 |                             |
+-------------+------------+------+-----+-------------------+-----------------------------+
```

1.3 Control and Measure

Each of the process described in the slow_conf.json configuration file is controlling a different hardware:

- **CAENSERVER** : it runs on lyosdhcal11 and connects the mysql database specified in the SLOWDB variable. The SETUP used in the GIFPP2015 database is GIFLYON2 in this case. On initialise it queries the detector connected, the corresponding HVRACK and connect to the CAEN SY1527 specified there. It uses the SETUP to monitor and/or regulate the detector channels. It provides also single channel access to change manually high voltage and current limit.
- **GPIOSERVER**: it runs on lyoilcrpi02 and controls the switch of the low voltage power supplies of the SDCC VME card and of all the DIFs

- **DS1820SERVER**: it runs on lyoilcrpi02 adn read 2 DS1820 temperature probes. The first one is in the box, the second is on the detector rack. The periodicity of the readout is available and data can be store in the mysql database in the table DS1820MON
- **BMP183SERVER**: it runs on lyoilcrpi24 that is connected outside the zone near the SY1527 rack. It measure the pressure and the temperature with a BMP183 Bosch device. Once again the periodicty of the readout can be controlled and the data can be stored in the mysql database in the PTMON table

Once again a python package LMainSlowControl is available to control all those processes:

- **import LMainSlowControl** : imports the python package
- d=LMainSlowControl.DimSlowControl("lyosdhcal11.cern.ch"): it creates the interface with the DNS lyosdhcal11.cern.ch
- d.scandns() : it scans the DNS and discover services
- d.CAENHVInitialise("test") : it uses the SLOWDB variable of the CAENSERVER program to access the database and query it for the given SETUP.
- d.CAENHVRead(48): it reads channel 48
- d.getCurrentHVChannel(): it returns last channel read
- d.getCurrentHVVoltageRead(): it returns last VMON read
- d.getCurrentHVCurrentRead(): it returns last IMON read
- d.getCurrentHVVoltageSet(): it returns last VSET read
- d.CAENHVSetOn(49) : turns HV on channel 49
- d.CAENHVSetOff(49): tuns HV Off on channel 49
- d.CAENHVSetVoltage(48,3300): it sets VSET to 3300 V on channel 48
- d.CAENHVSetCurrent(48,50): it sets I0 to 50 microA on channel 48
- d.CAENHVStartRegulation(60): it starts the regulation with a period of 60 s. It queries the database for each detector of the setup, looks for current Voltage, makes a mean of the last measure pressure measured by the BMP183 (PMEAN) and last TOUT temperature measured by the DS1820 probe (TMEAN). It compares the effective voltage to the VREF recalculated from (PREF,TREF) to (PMEAN,TMEAN) and correct the voltage VSET if the differenc is bigger than 20 V and bellow 200 V
- d.CAENHVStartMonitor(120): it reads all detector channel sepcified in the SETUP and store the in HVMON every 120 s

- d.CAENHVStopRegulation(): it stops the Regulation thread
- d.CAENHVStopMonitor(): it stops the monitroing thread
- d.BMP183SetPeriod(30): it set the period of the BMP183 reading
- d.BMP183Store("acqilc/RPC_2008@lyoac30.cern.ch:GIFPP2015"): it stores BMP183 data in PTMON table on the given account
- d.DS1820SetPeriod(30): it sets the period of the DS1820 reading
- d.DS1820Store("acqilc/RPC_2008@lyoac30.cern.ch:GIFPP2015"): it stores the DS1820 data in the DS1820MON table on the given account.
- d.OpenGPIO(): it opens the GPIO access
- d.VMEOn(): turns VME LV On
- d.VMEOff(): turns VME LV Off
- d.DIFOn(): turns DIF LV On
- d.DIFOff() turns DIF LV Off
- d.CloseGPIO(): closes the GPIO access

2 DAQ Architecture

2.1 PC names and role

The acquisition is based on 1 RaspberryPI micro computer controlling and reading up to 9 DIFs connected on each of the 7 planes: **lyoilcrpi02**. Data are collected and written to disk on the linux PC *lyosdhcal11*.

The RPI is responsible for chip configurations, DIF control and data readout. **lyoilcrpi02** is also handling the control of the SDCC. The central PC is in charged of the configuration database access, the data collection, the event building and writing and the human interface instance

Since DIM is the network framework used in this data acquisition, the DIM DNS must be stated once **lyosdhcal11** with

```
killall -9 dns
export DIM_DNS_NODE=lyosdhcal11
/opt/xdaq/bin/dns &>/dev/null &
```

2.2 DIF control

On each RPI should run a DIM daemon that controls the DIF connected to it. It is started with

sudo /etc/init.d/dimdifd start
its status is obtained with
sudo /etc/init.d/dimdifd status
and it can be stoped with
sudo /etc/init.d/dimdifd stop
The log file is **/var/log/dimdif.log**
It is a DIM server acknowledging the following commands:

- **SCANDEVICES** : List all DIFs connected on USB and publish a DIM Service DEVICES of 255 integers with the identifiers of the found DIFs
- **INITIALISE**: requires the DIF identifier, it opens the DIF devices and tries to write and read the test register. It updates the Dim Services STATE and INFO and returns an INITIALISED state in case of success or INIT_FAILED
- **REGISTERSTATE**: requires the control register (integer) and the current Database state (string). It stores the control register to be used at the next configuration and subscribes to the database DIM server services publishing DIF parameters for each DIF separately (/DB/*STATE_NAME*/DIF*id*). By this way it obtains from the dimdb service the configuration parameter of each DIF.
- **CONFIGURE**: it configures all connected DIF and their asoociated ASICs. It updates the STATE and the INFO of each DIF. The STATE contains the whole decoding of the slow control status.
- **START**: it starts the acquisition of each DIF. The STATE of each DIF is updated to STARTED.
- **STOP**: it stops the acquisition of each DIF. The STATE of each DIF is updated to STOPPED.

It also publishes 3 services per DIF:

- **STATE**: a string containing the current DIF state (FOUND, INITIALISED, PRECONFIGURED, CONFIGURED, STARTED, STOPPED)
- **INFO**: a structure with *dif id, status, slow control status, bcid, gtc, bytes read and pc name*
- **DATA**: the last DIF data buffer read.

2.3 SDCC control

A DIM daemon is running on the RPI (*clrtorp2-prive*) where the SDCC is connected.It is started with
sudo /etc/init.d/dimcccd start
its status is obtained with
sudo /etc/init.d/dimcccd status
and it can be stoped with
sudo /etc/init.d/dimcccd stop
The log file is **/var/log/dimccc.log**
It acknowlegdes the following commands:

- **INITIALISE**: requires the CCC FTDI identifier ("DCCCCC01"), it opens the SDCC devices and tries to write and read the test register.
- **CONFIGURE**: it sends DIFRESET and CCCRESET before the start of the acquisition
- **DIFRESET**: it sends a DIFRESET command on the DCC bus
- **CCCRESET**: it send a CCCRESET command on the DCC bus
- **PAUSE**: it send a PAUSE command on the DCC bus
- **RESUME**: it send a RESUME command on the DCC bus
- **START**: it send a START_ACQUISITION command on the DCC bus
- **STOP**: it send a STOP_ACQUISITION command on the DCC bus
- **TESTREGISTERREAD** : read the test register and update the REGISTER DIM service
- **TESTREGISTERWRITE** : requires an integer value that is written in the test register

It publishes 2 DIM services,**REGISTER** that is filled when TESTREGISTERREAD is called and an integer **STATUS** which value is changed depending on the last command received.

2.3.1 Possible issue

At power on, the USB control may be faulty. It is recommended to INITIALISE and use the TESTREGISTERWRITE/READ to check the correct access of the FPGA.

2.4 Database control

The database control is the responsability of one DIM daemon server running on the central PC (*clrtodaq0-prive*.It is started with
sudo /etc/init.d/dimdbd start
its status is obtained with
sudo /etc/init.d/dimdbd status
and it can be stoped with
sudo /etc/init.d/dimdbd stop
The log file is **/var/log/dimdb.log**
It acknowlegdes the following commands:

- **DOWNLOAD**: requires the name of the needed DB state ("LPCC_230"), it downloads the given state from the configuration databas and creates for each DIF a DIM service withe name /DB/*STATE_NAME*/DIF*id* containing the configuration parameters for all the ASICs of the given DIF
- **DELETE**: it deletes all the DIM services of the current state
- **NEWRUN**: it asks the database for a new run number and update /DB/RUNFROMDB service

2.5 Data writing

The data writing is achieved by one DIM daemon server running on the central PC (*lyosdhcal11*.It is started with
sudo /etc/init.d/dimwriterd start
its status is obtained with
sudo /etc/init.d/dimwriterd status
and it can be stoped with
sudo /etc/init.d/dimwriterd stop
The log file is **/var/log/dimwriter.log**
It acknowlegdes the following commands:

- **INITIALISE**: it subscribes to all possible DIF DATA and INFO services connected to the DIM DNS. It also subscribes to the RUNFROMDB service from the dimdb server. It creates an *ShmProxy* instance that is spying the */dev/shm* directory for new data collected. The *infoHandler* method of the server received data from the *dimdif* daemon DATA services and write them in */dev/shm*
- **START**: it opens an LCIO file in */data/online/Results* using the current value of run obtained by the RUNFROMDB service. It starts the */dev/shm* spying, collects data and writes them to disk when all publishing DIF have completed the same BCID buffer.
- **STOP**: it stops the spying and closes the data file.

2.5.1 Possible issue

This server is still partially in development. It has to be started after the other daemons in order to be able to collect run number and DIF data. In case of failure, please exit cleanly the daq control (see next section), restart this daemon and restart the daq control

3 DAQ control

The DAQ control is done on the main console loggin on lyosdhcal11 . There is a graphical interface and it will be desribe later on but the underlying control is done by the DimDaqControl package.

3.1 The DimDaqControl package

The control of the DAQ is done via *DimDaqControl* a standalone DIM program interfaced to python. The python package LSDHCALDimCtrl is in */data/online/SDHCAL/DimCtrl*. The environment needs to be set properly with:

```
cd /data/online/SDHCAL/DimCtrl
export LD_LIBRARY_PATH=.:./lib:$LD_LIBRARY_PATH
python
```

Inside the python shell launched you can do the following:

1. **import LSDHCALDimCtrl** : imports the python package
2. **s=LSDHCALDimCtrl.DimDaqControl("gifpp")**: creates an instance of DimDaqControl
3. **s.scandns()**: lists all DIMServer, services and commands available on the DIM DNS
4. **s.initialiseWriter("/data/online/Results")**: Initialise the data writing on the given directory. It has to be done only once if the dimwriter daemon is not stopped.
5. **s.scan()**: sends a SCANDEVICES command to all dimdif servers and wait for DEVICES service update. It must be called once or after a destroy()
6. **s.initialise()**: sends an INITIALISE command to all DIF found previously
7. **s._print()**: prints the status of each DIF on each dimdif server from the INFO service of each given DIF. Like the scan() command it must be called only after a scan() command

8. **s.download("LPCC_230")**: triggers the DELETE of the previous state in the DB server and the DOWNLOAD of the specified state. Since no semaphore mechanism is used in this command a 20 s delay may be needed before the data are available. It has to be called only when a new DB state needs to be downloaded or when the dimdb daemon is restarted

9. **s.registerstate(0x815A1B00,"LPCC_230")**: send a REGISTERSTATE command to all dimdif server. The state specified should be identical to the last downloaded state in dimdb. It must be called before configure to also set the control register.

10. **s.configure()**: sends a CONFIGURE command to all dimdif servers

11. **s.start()**: starts a run, sends a START command to the dimccc, the dimdif and the dimwriter servers

12. **s.stop()**: stops a run,sends a STOP command to the dimccc, the dimdif and the dimwriter servers

13. **s.destroy()**: send a DESTROY command to the dimdif servers. All the DIF handling are cleared locally so the only possible next command is scan()

The red commands needs to be used only once per session. The blue ones should be used once or if any changes has occurred on dimdif servers (power cycle for example). The purple ones are used to reconfigure chips. The green ones are the normal one to start and stop a run. Eventually the black commands are optional, the **download** one must be called at least ones to access the DB but is not needed in a new session if the dimdb server is alive and already contains the state

3.2 The StartDaq package

The *StartDaq* pacakge embedded the *DimDaqControl* and add hosts managment, LV and HV control. It can be started directly in python on lyosdhcal11. It uses a configuration file to handle the host managment and set the DB and trigger parameter. An exemple is found in SDHCAL/python/config_box.py:

```
host=['lyoilcrpi02']
ccc='lyoilcrpi02'
db='lyosdhcal11'
writer='lyosdhcal11'
register=0x815A1B00
# 115 + petit seup
state="GIFSPS_57"
# Novermber 2012 state="Dome_42chambres_Reference_v4_115"
# November 2012 + masks state="Dome_42chambres_Reference_v4_144"
```

```
directory="/data/NAS/GIF2015"
```

Inside the python shell launched you can do the following:

1. **import StartDaq** : imports the python package
2. **s=StartDaq.StartDaq("config_m3")**: creates an instance of StartDaq withc the config_m3.py files
3. **s.host_start()**: starts DIM daemon on all hosts list in config.host,config.ccc,config.db and config.writer names
4. **s.host_stop()**: stops DIM daemon on all hosts list in config.host,config.ccc,config.db and config.writer names
5. **s.rpi_start()**: starts DIM daemon on all hosts list in config.host
6. **s.rpi_stop()**: stops DIM daemon on all hosts list in config.host
7. **s.initialiseWriter("/data/online/Results")**: Initialise the data writing on the given directory. It has to be done only once if the dimwriter daemon is not stopped.
8. **s.scan()**: sends a SCANDEVICES command to all dimdif servers and wait for DEVICES service update. It must be called once or after a destroy()
9. **s.initialise()**: sends an INITIALISE command to all DIF found previously
10. **s._print()**: prints the status of each DIF on each dimdif server from the INFO service of each given DIF. Like the scan() command it must be called only after a scan() command
11. **s.download("LPCC_230")**: triggers the DELETE of the previous state in the DB server and the DOWNLOAD of the specified state. Since no semaphore mechanism is used in this command a 20 s delay may be needed before the data are available. It has to be called only when a new DB state needs to be downloaded or when the dimdb daemon is restarted
12. **s.registerstate(0x815A1B00,"GIFSPS_57")**: send a REGISTERSTATE command to all dimdif server. The state specified should be identical to the last downloaded state in dimdb. It must be called before configure to also set the control register.
13. **s.configure()**: sends a CONFIGURE command to all dimdif servers
14. **s.start()**: starts a run, sends a START command to the dimccc, the dimdif and the dimwriter servers

15. **s.stop()**: stops a run,sends a STOP command to the dimccc, the dimdif and the dimwriter servers

16. **s.destroy()**: send a DESTROY command to the dimdif servers. All the DIF handling are cleared locally so the only possible next command is scan()

The red commands needs to be used only once per session. The blue ones should be used once or if any changes has occurred on dimdif servers (power cycle for example). The purple ones are used to reconfigure chips. The green ones are the normal one to start and stop a run. Eventually the black commands are optional, the **download** one must be called at least ones to access the DB but is not needed in a new session if the dimdb server is alive and already contains the state

3.3 Spying with webDid

The latest versions of DIM provide a web interface to all DIM servers and clients. The server is started with:

```
export DIM_DNS_NODE=lyosdhcal11
/opt/dhcal/dimWebDID/webDid
```

Then the interface can be accessed via a web browser at address http://lyosdhcal11:2500 as seen on figure **??**

Commands can be sent manually but spying of dynamiclly allocated infos (like the DID DATA service) should be avoided since it crashes the program.

3.4 Graphical interface

3.4.1 Starting

```
cd SDHCAL/python
python SDaqImpl.py
```

This graphical interface is just a user friendly interface to the StartDaq package, each button corresponds to the call of one method. The figure **??**,**??**, and **??** shows the various panel with the sequence of call of each method.

4 How to

4.1 Lost connection to lyoilcrpi02

If the connection is lost to lyoilcrpi02, first check with a ping that the TCP/IP is not answering at all:

```
ping lyoilcrpi02
```

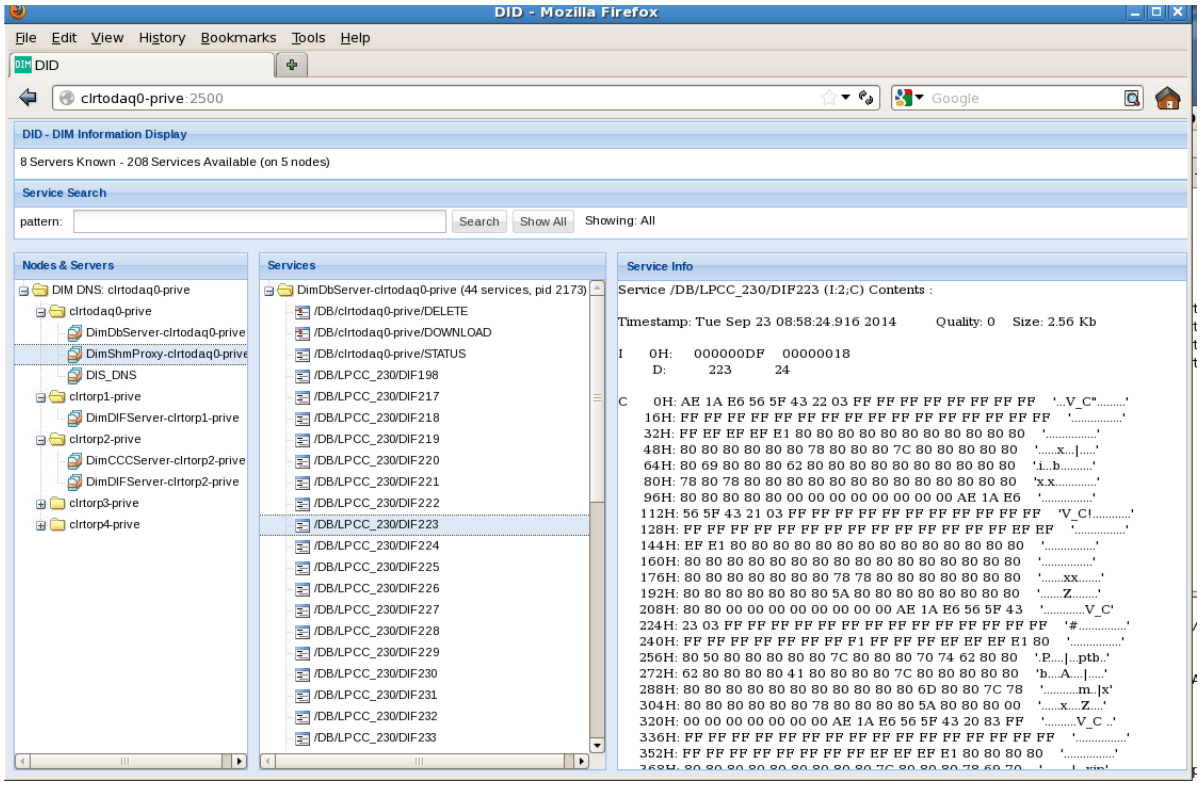

Figure 1: The web application webDID used to spy DIM services

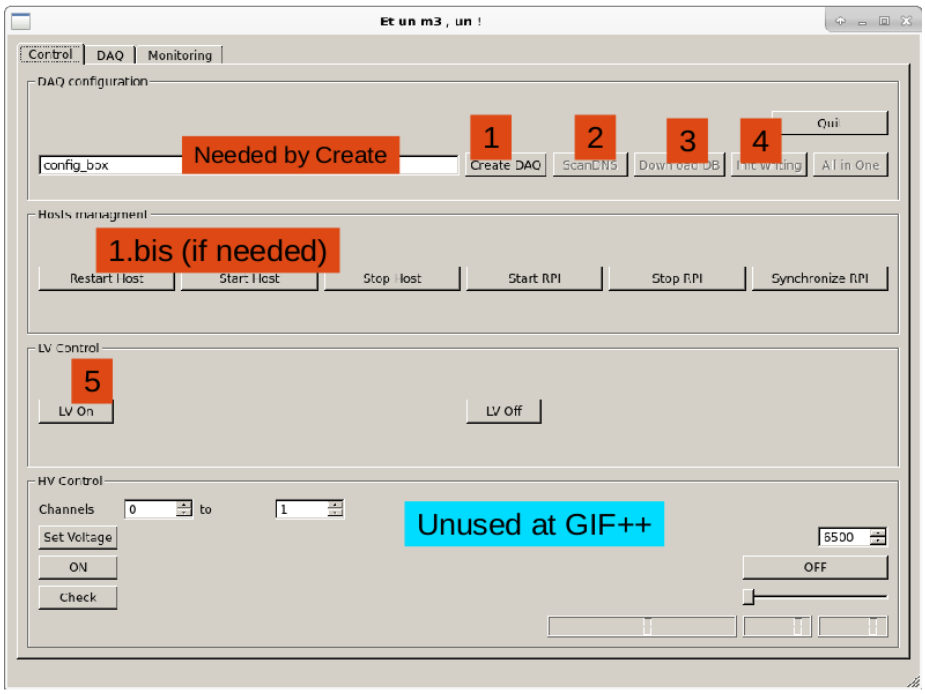

Figure 2: Process control panel. Restart host can be called after createDaq and can be called again after a DAQ destroy but must be followed by the step 2,3 and 4

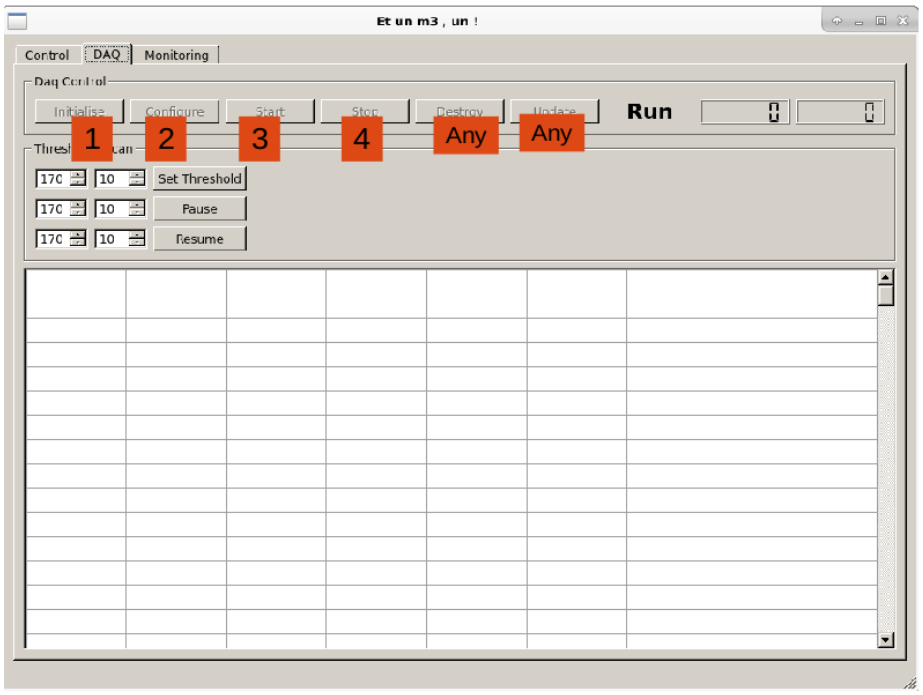

Figure 3: Daq control panel. Initialise must be called after Destroy. After a Stop, Configure (parameter reload) or Start can be called

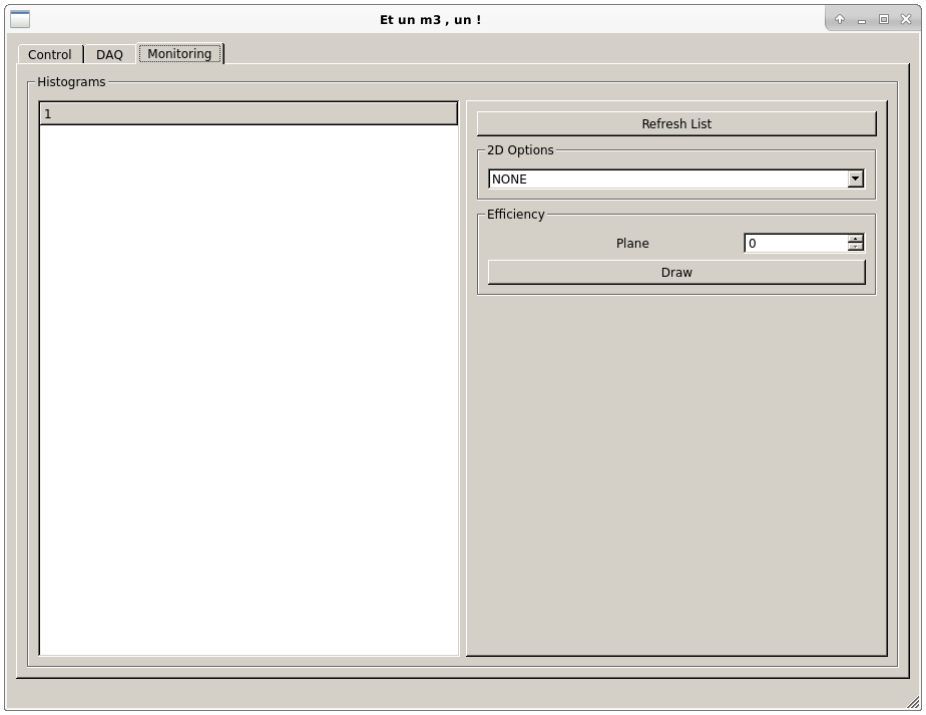

Figure 4: Monitoring Panel. RefreshList gives the list of histograms and access to an expanding tree of histograms. Doble-clicked on a histogram to display

If no communication is possible go in a browser to http://lyoilcpwr01 (admin/admin) and power cycle the box. Five minutes later you should be able to connect to lyoilcrpi02

```
ssh pi@lyoilcrpi02
```

then restart the job control on it

```
sudo /etc/init.d/dimjcd start
```

Now in the home of lyosdhcal11 in python

```
import Ldimjc
dsc=Ldimjc.DimJobInterface()
dsc.loadJSON("slow_conf.json")
dsc.clearHostJobs("llyoilcrpi02")
dsc.startJobs("lyoilcrpi02")
dsc.status()
dsc.List()
# you should see the list of process
import LMainSlowControl
l=LMainSlowControl.DimSlowControl("lyosdhcal11")
l.scandns()
l.DS1820SetPeriod(30)
l.DS1820Store("acqilc/RPC_2008@lyosdhcal11:GIFPP2015")
l.OpenGPIO()
l.VMEOn()
l.DIFOn()
# wait 1 minute
l.VMEOff()
l.VMEOn()
```

Keep this window open.
Now you can restart the DAQ graphical interface:

1. Create DAQ
2. Restart Hosts
3. Scan DNS
4. Download DB (wait 1 min)
5. Init Writing
6. LV On
7. Initialise

After Initialise, log on lyoilcrpi02 and check /var/log/dimccc.log if the last return code is -1 that means that the USB communication to the SDCC failed and the program exited. Go in the slow control window and repeat the VME Off/On step ,then repeat the sequence

1. Destroy
2. Restart Hosts
3. Scan DNS
4. Download DB (wait 1 min)
5. Init Writing
6. LV On
7. Initialise

The /var/log/dimccc.log should have return code 0 and you can now do your run.

4.2 One DIF is in INIT_FAILED

That means that the USB communication cannot be established. It might be due to a connection problem (LV or HDMI) but also a SDCC issue (check again /var/log/dimccc.log)

4.3 One DIF cannot configured

Check:

- HDMI connection
- LV voltage on the DIF (it must be greater than 5 V)
- DIF-ASU connector (LV Off, replug it)

If none of this fix works, it means one ASIC is faulty. There is few alternatives

1. Try to run: You're not sure all chips are configured and it may block the trigger
2. Keep the chamber in the setup but disable it. Unplug the LV of the DIF
3. Dismount and debug the chamber