daqcontrol Documentation

Release 1

L.Mirabito

Jan 13, 2020

CONTENTS

Contents:

class daqcontrol.**fdaqClient**

analysis_histo(*h*)
Analysis: Get the JSON representation of a ROOT histogram

Warning Only FEBV1 is implemented yet

Parameters **h** – Complete path of histo name

Returns JSON answer

analysis_histolist()
Analysis: Get the JSON list of histogram available

Warning Only FEBV1 is implemented yet

Returns JSON answer

analysis_monitor(*run*)
Analysis: Starts to Monitor data stored in /dev/shm/monitor

Warning Only FEBV1 is implemented yet

:param run:run to analyse :return: JSON answer

analysis_process(*run*)
Analysis: Process one file

Warning Only FEBV1 is implemented yet

:param run:run to analyse :return: JSON answer

analysis_status()
Analysis: Get the status of the processing (run,evt)

Warning Only FEBV1 is implemented yet

Returns JSON answer

analysis_stop(*run*)
Analysis: Stop to Monitor data stored in /dev/shm/monitor

Histogram are written int /tmp

Warning Only FEBV1 is implemented yet

:param run:run to analyse :return: JSON answer

daq_configure()
Send a CONFIGURE transition to FDAQ :return: asnwer to the transition

daq_create()
Send a CREATE transition to the FDAQ process

daq_dbstatus()
Send an DBSTATUS command to FDAQ

Returns answer to the command, the status of the DBCACHE (run ,DB State)

daq_destroy()
Send a DESTROY transition to FDAQ :returN: answer to the transition

daq_difstatus()
Send A DIFSTATUS command to FDAQ

Returns answer to the command, a list of all DIF or GRIC with their status

daq_discover()
Send a DISCOVER transition to FDAQ :return: asnwer to the transition

daq_downloaddb(*state*, *version*)
Send a DOWNLOADDB command to FDAQ

Parameters

- **state** – State name
- **version** – State version if needed

Returns answer of the command

daq_evbstatus()
Send an EVBSTATUS command to FDAQ

Returns answer to the command, the status of the Event Builder (run , events built)

daq_forceState(*name*)
Send a FORCESTATE command to FDAQ

Parameters **name** – State name to be set

daq_getparameters()
Send a GETPARAM command to FDAQ

daq_info(*name*)
List all information about process named name

Parameters **name** – Name of the process

Returns A string with a pretty print of the informations

daq_initialise()
Send a INITIALISE transition to FDAQ :return: asnwer to the transition

daq_list()
List all information from all processes in the configuration

Returns A string with a pretty print of the informations

daq_normalstop()
Send a STOP transition to FDAQ and a MONITOR command to stop the monitoring :return: answer to the transition

daq_process()
Send a LISTPROCESS command :return: JSON answer :obsolete: Use jc_info instead

daq_services()
Send a PREPARE transition to FDAQ :return: asnwer to the transition

daq_setparameters()
Send a SETPARAM command to FDAQ

daq_setrunheader(*rtyp*, *value*, *mask=4294967295*)
Event Builder: Set the run header (used for calibration)

Parameters

- **rtyp** – Type of run (0=physic,1= VTHTIME scan,2= CAL6B scan)
- **value** – Value of VThTime or 6BDAC
- **mask** – Channel mask

Returns JSON answer

daq_start(*comment=None*)
Start a run: Reset trigger counter and calibration settings if MDCC is used Send a START transition to FDAQ Send a MONITOR command to FDAQ

Parameters **comment** – A comment of the run

Returns answer to the transition

daq_state()
Check The state of FDAQ

Returns A string with the state name

daq_stop()
Stop: If a FEBV1 Scurve is running, stop it Else Make a normalStop :return: the answer to the effective command

daq_tdcstatus()
Send A TDCSTATUS command to FDAQ

Returns answer to the command, a list of all FEBV1 with their status

dif_ctrlreg(*ctrl*)
DIF: Send a CTRLREG command to FDAQ

Parameters **ctr** – An hexadecimal string of the control register of DIF

Returns the answer to the command

dif_setgain(*gain*)
DIF: Send a SETGAIN command to FDAQ (DIF gain setting)

Parameters **gain** – the PAGAIN to be set on HR2

Returns answer to the command

dif_setthreshold(*b0*, *b1*, *b2*)
DIF: Send a SETTHRESHOLD command to FDAQ (DIF thresholds setting)

Parameters

- **b0** – B0 threshold of HR2
- **b1** – B1 threshold of HR2
- **b2** – B2 threshold of HR2

Returns answer to the command

ecal_pause()
MDCC: Send a ECALPAUSE command to FDAQ

Returns JSON answer

Obsolete Use for ECAL SDHCAL tests

ecal_resume()
MDCC: Send a ECALRESUME command to FDAQ

Returns JSON answer

Obsolete Use for ECAL SDHCAL tests

feb_cal6bdac(*mask*, *shift*)
FEBV1: Send a CAL6BDAC command to FDAQ, FEBV1 6BDAC calibration

Parameters

- **mask** – 32 Channel mask
- **shift** – 6bdac shift value to be added to loaded value

Returns JSON answer

feb_calibdac(*ntrg*, *ncon*, *ncoff*, *dacmin*, *dacmax*, *mask*, *step=5*, *vth=480*)
FEBV1: Acquisition loop for 6bdac scan

Parameters

- **ntrg** – Number of windows per point
- **ncon** – Number of clock active
- **ncoff** – Number of clock off
- **dacmin** – lowest dac value
- **dacmax** – highest dac value
- **step** – Loop step
- **vth** – Fixed Threshold

feb_fullcalib(*ch*, *spillon*, *spilloff*, *beg*, *las*, *step=2*, *asic=255*, *mode='FEB1'*)
FEBV1: Acquisition loop interface for 6BDAC scan

Parameters

- **ch** – 255 = according tomode channel/channel, 1023 = all channels, other value= channel(PETIROC)
- **spillon** – Number of clock active
- **spiloff** – Number of clock off
- **beg** – lowest DAC value
- **las** – highest DAC value
- **step** – Loop step
- **asic** – Asic mask (1,2,or 3)
- **mode** – OLD=WT board,COAX=coaxial PCB, FEBV0=with return line PCB,FEBV1

feb_fullscurve(*ch*, *spillon*, *spilloff*, *beg*, *las*, *step=2*, *asic=255*, *mode='FEB1'*)
FEBV1: Acquisition loop interface for VTHTIME scan

Parameters

- **ch** – 255 = according tomode channel/channel, 1023 = all channels, other value= channel(PETIROC)
- **spillon** – Number of clock active
- **spiloff** – Number of clock off
- **beg** – lowest VTHTIME value
- **las** – highest VTHTIME value
- **step** – Loop step
- **asic** – Asic mask (1,2,or 3)
- **mode** – OLD=WT board,COAX=coaxial PCB, FEBV0=with return line PCB,FEBV1

feb_injection_configure()
FEB Inj: Configure transition

Returns JSON answer

feb_injection_delay(*nb*)
FEB Inj: Injection delay in the windows command

Parameters **nb** – delay in 25 ns clocks

Returns JSON answer

feb_injection_destroy()
FEB Inj: DESTROY transition

Returns JSON answer

feb_injection_duration(*nb*)
FEB Inj: Injection duration command

Parameters **nb** – duration in 25 ns clocks

Returns JSON answer

feb_injection_height(*nb*)
FEB Inj: Injection height command

Parameters **nb** – pulse height (0-64) non linear

Returns JSON answer

feb_injection_internal()
FEB Inj: Injection internal trigger command

Returns JSON answer

feb_injection_mask(*side*, *mask*)
FEB Inj: Channel masking command

Parameters

- **side** – 0/1 for High Radius / Low radius
- **mask** – Channel mask

Returns JSON answer

feb_injection_number(*nb*)
FEB Inj: Injection number of trigger command

Parameters **nb** – Maximal number of trigger

Returns JSON answer

feb_injection_pause()
FEB Inj: Injection pause command

Returns JSON answer

feb_injection_resume()
FEB Inj: Injection resume command

Returns JSON answer

feb_injection_soft()
FEB Inj: Injection sof trigger command

Returns JSON answer

feb_injection_source(*source*)
FEB Inj: Injection source command

Parameters **source** – External/soft/Internal (to be precised)

Returns JSON answer

feb_lut_calib(*tdc*, *channel*)
FEB LUT: LUT calibration command

:param tdc:FEB instance number :param channel: FEB channel to be calibrated

feb_lut_draw(*tdc*, *channel*, *canvas=None*, *feb=15*)
FEB LUT: LUT Draw for one TD and one channel

:param tdc:FEB instance number :param channel: FEB channel :param canvas: ROOT TCanvas if already existing :param feb: FEB id, lowest byte of the IP address

feb_lut_dump(*tdc*, *feb=15*)
FEB LUT: LUT dump for one TDC and all channels to summary_LUT_febid.txt

:param tdc:FEB instance number :param feb: FEB id, lowest byte of the IP address

feb_resettdc()
FEBV1: Send a RESETTDC command to FDAQ. it resets the firmware of each FEB

Returns answer to the command

feb_scurve(*ntrg*, *ncon*, *ncoff*, *thmin*, *thmax*, *mask*, *step=5*)
FEBV1: Acquisition loop for VTHTIME scan

Parameters

- **ntrg** – Number of windows per point
- **ncon** – Number of clock active
- **ncoff** – Number of clock off
- **thmin** – lowest VTHTIME value
- **thmax** – highest VTHTIME value
- **step** – Loop step

feb_set6bdac(*value*)
FEBV1: Send a SET6BDAC command to FDAQ, FEBV1 6BDAC scan

Parameters **value** – 6bdac value to be set on all active channels

Returns JSON answer

feb_setmask(*value*, *asic*)
FEBV1: Send a SETMASK command to FDAQ, FEBV1 mask set

Parameters

- **value** – ams to be applied
- **asic** – 1,2 or 3 for ASIC 1 ,2 or both

Returns JSON answer

feb_settdcdelays(*active*, *dead*)
FEBV1: Send a SETTDCDELAYS command to FDAQ (FEBV1 latch settings)

Parameters

- **active** – Number of active latch clocks
- **dead** – Number of reset latch clocks

Returns answer to the command

feb_settdcmode(*mode*)
FEBV1: Send a SETTDCMODE command to FDAQ (FEBV1 mode)

Parameters **mode** – 0 all buffer, 1 only trigegr ones

feb_setvthtime(*value*)
FEBV1: Send a SETVTHTIME command to FDAQ, FEBV1 VTHTIME scan

Parameters **value** – VTHTIME value to be set on all Asics

Returns JSON answer

feb_testmask(*tdc*, *mask*)
FEB LUT: LUT Physic mask command

:param tdc:FEB instance number :param mask: FEB channel mask used for physic runs.

jc_appcreate()
Loop on all computers defined in the configuration and send an APPCREATE Command, it sends a transition CREATE to all baseApplication

Returns Dictionnary of answer to APPCREATE command

jc_create()
Loop on all computers defined in the configuration and initialise their job control with the loaded configuration

Returns Dictionnary of answer to INITIALISE transition

jc_destroy()
Loop on all computers defined in the configuration and clean the loaded configuration

Returns Dictionnary of answer to DESTROY transition

jc_info(*hostname*, *apname=None*)
Pretty print of processes status and application settings

Parameters

- **hostanem** – Host name where the jobcontrol is running
- **appname** – Application name , if None all application are dumped

jc_kill()
Loop on all computers defined in the configuration and kill the process defined with the loaded configuration

Returns Dictionnary of answer to KILL transition

jc_oldstatus(*apname=None*)
Same as jc_status

Obsolete use jc_status instead

jc_restart(*host*, *jobname*, *jobpid*)
Kill and restart one process

Parameters

- **host** – Host name

- **jobname** – Name of the process in the job control
- **pid** – Process id

jc_start()
Loop on all computers defined in the configuration and start the process defined with the loaded configuration

Returns Dictionnary of answer to START transition

jc_status(*apname=None*)
Pretty print of the status of all process and application informations defined in the configuration

Parameters **appname** – if set , only this application name is dump

loadConfig(*name=None*, *login=None*)
Load in memory the configuration of the DAQ to be used If the name and the login are undefined, the configuration is selected using 3 environnment variables

•DAQURL : the url to the ilcconfdb (Oracle)

•DAQFILE : the JSON file to use

•DAQMONGO: The CONFIG_NAME:VERSION in MongoDB (using MongoJob access)

Only one of the 3 must be set

Parameters

- **name** – name of the confihguration
- **login** – Oracle Web login

parseConfig()
Access and parse the configuration defined by one of the environnement variable (DAQURL,DAQFILE or DAQMONGO)

slow_clearalarm(*first*, *last*)
Slow Control: FSLOW application Command: HV clear alarm

It finds a HV controlling application and clear alarm of specified channels

Parameters

- **first** – First channel
- **last** – Last channel

Returns JSON HV status

slow_configure()
Slow Control: FSLOW application : send a CONFIGURE transition

It configures all Slow Control application previuosly discovered

slow_create()
Slow Control: Find FSLOW application and send a CREATE transition

slow_discover()
Slow Control: FSLOW application: send a DISCOVER transition

slow_humstatus()
Slow Control: FSLOW application Command: HIH8000 humidity and Temperature

It finds a HIH8000 controlling application and get it status

Returns JSON Humidity and temperature status

slow_hvoff (*first*, *last*)
Slow Control: FSLOW application Command: HV Off

It finds a HV controlling application and turn HV off of specified channels

Parameters

- **first** – First channel
- **last** – Last channel

Returns JSON HV status

slow_hvon (*first*, *last*)
Slow Control: FSLOW application Command: HV On

It finds a HV controlling application and turn HV on of specified channels

Parameters

- **first** – First channel
- **last** – Last channel

Returns JSON HV status

slow_hvstatus (*first*, *last*)
Slow Control: FSLOW application Command: HV status

It finds a HV controlling application and get the status of specified channels

Parameters

- **first** – First channel
- **last** – Last channel

Returns JSON HV status

slow_iset (*first*, *last*, *value*)
Slow Control: FSLOW application Command: HV current limit setting

It finds a HV controlling application and set the maximum current (microA) of specified channels

Parameters

- **first** – First channel
- **last** – Last channel
- **value** – I max (microA)

Returns JSON HV status

slow_lvoff ()
Slow Control: FSLOW application Command: Turns LV OFF

It finds a LV controlling application and turn it off

Returns JSON LV status

slow_lvon ()
Slow Control: FSLOW application Command: Turns LV ON

It finds a LV controlling application and turn it on

Returns JSON LV status

slow_lvstatus()
Slow Control: FSLOW application Command: LV status

It finds a LV controlling application and get it status

Returns JSON LV status

slow_ptstatus()
Slow Control: FSLOW application Command: BMP Pressure and Temperature

It finds a BMP controlling application and get it status

Returns JSON PT status

slow_rampup(*first*, *last*, *value*)
Slow Control: FSLOW application Command: HV ramp up

It finds a HV controlling application and set the rising ramp of specified channels

Parameters

- **first** – First channel
- **last** – Last channel
- **value** – ramp (V/s)

Returns JSON HV status

slow_start()
Slow Control: FSLOW application : send a START transition

It starts the monitoring loop of all monitorApplications. It triggers the storage in the monitoring process

slow_stop()
Slow Control: FSLOW application : send a STOP transition

It stops any monitoring publication and storage

slow_vset(*first*, *last*, *value*)
Slow Control: FSLOW application Command: HV voltage setting

It finds a HV controlling application and set the voltage of specified channels

Parameters

- **first** – First channel
- **last** – Last channel
- **value** – Volatge in V

Returns JSON HV status

trig_beam(*clock*)
MDCC: Send a BEAMON command to FDAQ, it sets the beamon delay

Parameters **clock** – number of clock length

Returns JSON answer

trig_calibcount(*value*)
MDCC: Send a CALIBCOUNT command to FDAQ, FEBV1 calibration

Parameters **value** – Number of calibration frame

Returns JSON answer

trig_calibon(*value*)
MDCC: Send a CALIBON command to FDAQ, FEBV1 calibration

Parameters **value** – Calibration frame width

Returns JSON answer

trig_getregister(*address*)
MDCC: Send a GETREGISTER command to FDAQ, it reads a register of the MDCC

Parameters **address** – Register Address

Returns JSON answer

trig_hardreset()
MDCC: Send a SETHARDRESET command to FDAQ

Returns JSON answer

Obsolete Used for FEBV0

trig_pause()
MDCC: Send a PAUSE command to FDAQ

Returns JSON answer

trig_reloadcalib()
MDCC: Send a RELOADCALIB command to FDAQ, FEBV1 calibration rearming

Returns JSON answer

trig_reset()
MDCC: Send a RESETCOUNTERS command to FDAQ

Returns JSON answer with MDCC status

trig_resume()
MDCC: Send a RESUME command to FDAQ

Returns JSON answer

trig_setregister(*address*, *value*)
MDCC: Send a SETREGISTER command to FDAQ, it sets a register of the MDCC

Parameters

- **address** – Register Address
- **value** – Register Value

Returns JSON answer

trig_spilloff(*clock*)
MDCC: Send a SPILLOFF command to FDAQ, it sets the Spill minimal delay

Parameters **clock** – number of clock length

Returns JSON answer

trig_spillon(*clock*)
MDCC: Send a SPILLON command to FDAQ, it sets the Spill length

Parameters **clock** – number of clock length

Returns JSON answer

trig_spillregister(*value*)
MDCC: Send a SPILLREGISTER command to FDAQ, it sets the MDCC mode

Parameters **value** – Spill register value

Returns JSON answer

trig_status()
MDCC: Send a TRIGGERSTATUS command to FDAQ

Returns JSON answer with MDCC status

trig_tdcreset()
MDCC: Send a RESETTDC command to FDAQ, it makes a full reset of FEBV1 firmware

Returns JSON answer

daqcontrol.**executeFSM**(*host*, *port*, *prefix*, *cmd*, *params*)
Access to the FSM of a zdaq::baseApplication

Parameters

- **host** – Host name
- **port** – Application port
- **prefix** – Prefix of the application , ie, http:://host:port/prefix/.....
- **cmd** – Transition
- **params** – Value of the content tag

Returns url answer

daqcontrol.**executeCMD**(*host*, *port*, *prefix*, *cmd*, *params*)
Access to the CoMmanDs of a zdaq::baseApplication

Parameters

- **host** – Host name
- **port** – Application port
- **prefix** – Prefix of the application , ie, http:://host:port/prefix/.....
- **cmd** – Command name
- **params** – CGI additional parameters

Returns url answer

daqcontrol.**executeRequest**(*surl*)

Acces to an url

Parameters **surl** – The url

Returns url answer

CHAPTER

ONE

INDICES AND TABLES

- genindex
- modindex
- search

d

A

D

E

F

J

L

P

S

T