ISPECTOR SDK- REST API

Introduction

Ispector use a JSON based REST API for control and configuration of the device. HTTP API allows to perform every operations supported by the web based GUI from a remote host. Ispector use GET/POST HTTP call to both retrieve spectrum/wave data and configure the instruments.

HTTP protocol, the same used to transport web page, has the big advantage to easy pass firewall and usually it does not require any special configuration on the network.

Several endpoints are available

Endpoint (WEB PAGE)	Operation	Description
/set_config.cgi	POST	Set configuration of HV/MCA/PSD
/status.cgi	GET	Read the status of the instruments and of all available processing channels (MCA/PSD)
/spectrum.cgi	GET	Get spectrum data
/wavedump.cgi	GET	Get waveform data
/psd.cgi	GET	Get PSD data (not available in current API version)
/get_mca_config.cgi	GET	Readback MCA/PSD configuration
/resetspectrum.cgi	GET	Reset spectrum
/mca_run.cgi	GET	Start MCA acquisition
/mca_stop.cgi	GET	Stop MCA acquisition
/fb_settings.cgi	GET	Retrieve fabric configuration
/get_sysx.cgi	GET	Retrieve firmware version and installed options

In order to access to the endpoint perform HTTP GET/POST to address: http://<ispector_ip>/set_config.cgi.
Default Ispector IP is: http://192.168.50.2

Ispector API support multiple user simultaneously connected to the instruments. Meanwhile your are developing, keep your browser open on the Ispector page in order see in live the performed operation. Please mind that configuration parameters are loaded on page load. If you change parameters from API you need to refresh browser page in order to see parameters changing on the GUI.

Set Configuration

In order to configure the Ispector the /set_config.cgi endpoint is available. This endpoint is a POST service. You need to perform a RAW post POST operation sending in the body of the POST the configuration JSON. Do not use form-data or x-www-form-urlencoded because they are not supported by ISPECTOR

There are two different possible configurations you can send to ISPECTOR

- HV CONFIGURATION
- MCA CONFIGURATION

HV CONFIGURATION

```
{"command" : "SET_CHANNEL_CONFIG","channel_config" :[{"id" : 0,"HV_STATUS" : true,"HV_VOLTAGE" :
41.5,"MaxV" : 46,"MaxI" : 5, "RAMP" : 20,"TCoeff" : -34,"HV_MODE" : "temperature","HV_PWRON" :
true}],"store_flash" : false}
```

All parameters are case sensitive

PARAMETERS	VALID VALUES	FUNCTION
HV_STATUS	true/false	Enable/Disable HV

HV_VOLTAGE	22..80	HV voltage. Please pay attention to do not destroy the sensor setting too high voltage
MaxV	22..80	Max output voltage
MaxI	0..9	Trip Current in mA
RAMP	1..100	Ramp speed of HV
TCoeff	-1000 .. 1000	[mV/°C] Temperature compensation coefficient
MODE	“digital” “temperature”	Enable / Disable temperature compensation on HV
HV_PWRON	true/false	Power on/off the HV on instrument boot

MCA CONFIGURATION

```
{"command" : "SET_CHANNEL_CONFIG","mca_config" :[{"id" : 0,"trigger_thrs" : 28,"trigger_inib" :
300,"int_pre" : 300,"int_val" : 10,"int_gain" : 80,"pileup_inib" : 30,"pileup_pen" :
30,"baseline_inib" : 24,"baseline_len" : 256,"taget_run" : 0,"taget_value" : 0,"reset_on_apply" :
true}],"store_flash" : false}}
```

All parameters are case sensitive

PARAMETERS	**VALID VALUES**	**FUNCTION**
trigger_thrs	10..1000 [int]	(LSB) Trigger threshold
trigger_inib	10..1000 [int]	(ns) Trigger inhibit after a trigger events. (set in in order to avoid double triggers)
int_pre	0..1000 [int]	(ns) Charge integrator pre-trigger integration extension
int_val	0..100 [float]	(us) Charge integrator integration time
int_gain	0..1000 [int]	Charge integrator GAIN
pileup_inib	0..100 [float]	(us) Pileup inhibition after a trigger
pileup_pen	0..100 [float]	(us) Pileup penalty if a pileup event occurs
baseline_inib	0..100 [float]	(us) Baseline inhibition after a trigger
baseline_len	1024,512,256, 128,64,32,16	Length is samples of the moving average used to calculate the baseline
taget_run	0,1,2	Acquisition run mode 0 – FREE 1 – TIME CONTRAINED (ms) 2 – TOTAL COUNTS ON SPECTRUM
taget_value	[int]	Referring to taget_run parameters, this field specify the run limit. For example to run for 10 seconds set taget_run=1 and taget_value=10000
reset_on_apply	true/false	Reset spectrum when one or more configuration parameters are changed

Get Instrument Status

In order to get the status of the inspector perform GET to the following page /status.cgi

```
{
 "command":"GET_SYSTEM_STATUS",
 "Result":"ok",
 "ErrorCode":0,
 "Reason":"",
 "current_status":{
    "system_status":{
       "temperature":0,
       "eth_status":0,
       "eth_ip":"192.168.50.2",
       "last_user_interact":-1,
       "power":"wall",
       "battery":false,
       "battery_life":0,
       "battery_charge":0,
       "battery_in_charge":false,
       "remaining_time":0,
       "battery_voltage":0,
       "battery_current":0,
       "battery_temperature":0,
       "alarm":0,
       "httpcloud":0,
       "loracloud":0
    },
    "channels":[
       {
          "id":0,
          "HV_STATUS":true,
          "HV_VOLTAGE":41.5,
          "HV_MODE":"temperature",
          "COMPL_V":false,
          "COMPL_I":false,
          "Vout":42.38652,
          "Vref":1.954313,
          "Iout":0.3540874,
          "IoutRAW":0.050250001,
          "Temp":50.199402,
          "SetPoint":42.356781,
          "ICR":1294,
          "OCR":1254,
          "runtime":4542,
          "livetime":4540,
          "sattime":0,
          "incnt":5856370,
          "outcnt":5646006,
            "live":0.969088,
            "dead":0.030912,
            "mca_running":1,
            "mca_status":0
         }
      ]
   }
}
```

Get Spectrum

In order to get the status of the inspector perform GET to the following page /spectrum.cgi

```
{
   "command":"GET_SPECTRUM",
   "Result":"ok",
   "ErrorCode":0,
   "Reason":"",
   "data":[
      3,
      2,
      4,
      7, ..,
      1883
   ]
}
```

The data field in the JSON contains the spectrum data in a 4096 bin array

Get Waveform

In order to get the status of the inspector perform GET to the following page /wavedump.cgi

```
{
   "command":"GET_WAVEDUMP",
   "Result":"ok",
   "ErrorCode":0,
   "Reason":"",
   "data":[
      [
         3421,
         0,
         0,
         0,
         1,
         0,
         0
      ],
      [
         3425,
         0,
         0,
         0,
         1,
         0,
         0
      ],
      [
         3425,
         0,
         0,
         0,
         1,
         0,
         0
      ],
      [
         3423,
         0,
         0,
         0,
         1,
         0,
         0
      ],
      …
   ]
}
```

The data field in the JSON contains the waveform data.
Each data element is and array of 7 elements:

1) Analog data
2) Trigger pulse
3) Charge Integration Window
4) PSD Tail integration Window
5) Baseline restorer status
6) Pile up rejector discard
7) Pile up inhibition

Readback MCA configuration

In order to get the status of the inspector perform GET to the following page /get_mca_config.cgi

Get_mca_config.cgi

```
{
   "command":"GET_CHANNEL_CONFIGURATION",
   "Result":"ok",
   "ErrorCode":0,
   "Reason":"",
   "mca_config":[
      {
         "id":0,
         "trigger_thrs":28,
         "trigger_inib":300.000000,
         "int_pre":300.000000,
         "int_val":10.000000,
         "int_gain":80.000000,
         "pileup_inib":30.000000,
         "pileup_pen":30.000000,
         "baseline_inib":24.000000,
         "baseline_len":256,
         "rebinnig":4096,
         "reset_on_apply":true,
         "taget_run":0,
         "taget_value":0,
         "psd_gain":1.000000,
         "psd_delay":0.500000,
         "psd_int":0.800000,
         "scaleTimeWave":0
      }
   ]
}
```

Testing API

Use CURL to POST/GET data

EXAMPLE: Send configuration for MCA

```
curl -d '{"command" : "SET_CHANNEL_CONFIG","mca_config" :[{"id" :
0,"trigger_thrs" : 100,"trigger_inib" : 100,"int_pre" : 300,"int_val" :
10,"int_gain" : 100,"pileup_inib" : 10,"pileup_pen" : 10,"baseline_inib"
: 10,"rebinnig" : 4096,"baseline_len" : 512,"taget_run" : 0,"taget_value"
: 0,"reset_on_apply" : true}],"store_flash" : false}}' -H "Content-Type:
application/json" -X POST http://192.168.50.2/set_config.cgi
```

EXAMPLE: Read system status

```
curl -X POST http://192.168.50.2/status.cgi
```

EXAMPLE: Download spectrum

```
curl -X POST http://192.168.50.2/spectrum.cgi
```

Python SDK

An SDK for Python language is available. The SDK use HTTP API communication to interface with the Ispector.

The SDK requires Python >3.2 and works both on x86/ia64/ARM processor. It could be used as well as on a standard PC and on a Raspberry PI.

REQUIRED MODULES FOR SDK:

The SDK requires the following module:

- `requests [pip install requests]`
- `enum [pip install enum]`

REQUIRED MODULES FOR SDK:

The Example file requires the following module:

- `pprint [pip install pprint]`
- `numpy [pip install numpy]`
- `matplotlib [pip install matplotlib]`

DOWNLOAD THE SDK:

Python SDK files can be download from Nuclear Instruments Github
https://github.com/NuclearInstruments/IspectorSDK-Python

or cloned with git:

```
git clone https://github.com/NuclearInstruments/IspectorSDK-Python.git
```

The SDK include library (inspector_sdk.py) and an example file (test_ispector_sdk.py)

LIBRARY USAGE

In order to use the library, import ispector_sdk and open a connection creating a new ispector_sdk object

```
from ispector_sdk import ispector_sdk
I1 = ispector_sdk("192.168.50.2")
```

LIBRARY FUNCTION

```
set_hv_basic(self, hv_on, hv_voltage):
```

Set HV basic function

Parameter	Type	Description
hv_on	bool	Enable/Disable HV
hv_voltage	float	HV voltage

Return: NONE

```
def set_hv_compensation(self, mode, temp_coeff):
```

Set HV basic temperature compensation parameters

Parameter	Type	Description

mode	HVCompensation	DISABLE_COMPENSATION: no active temperature compensation ENABLE_COMPENSATION: active temperature compensation
temp_coeff	int	SiPM temperature compensation in mV/°C

Return: NONE

```
set_hv_cfg(self, ramp, maxI, maxV, on_starup):
```

Set HV advanced parameter

Parameter	Type	Description
ramp	int	[V/s] HV ramp speed
maxI	int	[mA] HV trip current
maxV	int	[V] Protection maximum voltage
on_starup	bool	Power on/off the HV on instrument boot

Return: NONE

```
configureMCA(self, trigger_threshold,
                 trigger_inibit, pre_int_time,
                 int_time, int_gain, pileup_inib,
                 pileup_penality, baseline_inib,
                 baseline_len, target_run,
                 target_value):
```

Configure MCA parameters

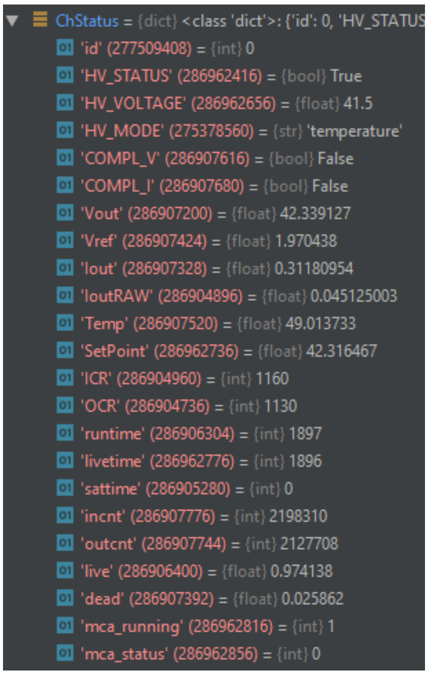
PARAMETERS	VALID VALUES	FUNCTION
trigger_threshold	10..1000 [int]	(LSB) Trigger threshold
trigger_inibit	10..1000 [int]	(ns) Trigger inhibit after a trigger events. (set in in order to avoid double triggers)
pre_int_time	0..1000 [int]	(ns) Charge integrator pre-trigger integration extension
int_time	0..100 [float]	(us) Charge integrator integration time
int_gain	0..1000 [int]	Charge integrator GAIN
pileup_inib	0..100 [float]	(us) Pileup inhibition after a trigger
pileup_pen	0..100 [float]	(us) Pileup penalty if a pileup event occurs
baseline_inib	0..100 [float]	(us) Baseline inhibition after a trigger
baseline_len	[BaselineLength]	Length is samples of the moving average used to calculate the baseline
target_run	[RunMode]	Acquisition run mode 0 – FREE 1 – TIME CONTRAINED (ms) 2 – TOTAL COUNTS ON SPECTRUM
target_value	[int]	Referring to taget_run parameters, this field specify the run limit. For example to run for 10 seconds set taget_run=1 and taget_value=10000

Return: NONE

```
getChannelStatus(self):
```

Read channel stats parameters

Return: Dictionary with channel status information



```
ChStatus = {dict} <class 'dict'>: {'id': 0, 'HV_STATUS
    'id' (277509408) = {int} 0
    'HV_STATUS' (286962416) = {bool} True
    'HV_VOLTAGE' (286962656) = {float} 41.5
    'HV_MODE' (275378560) = {str} 'temperature'
    'COMPL_V' (286907616) = {bool} False
    'COMPL_I' (286907680) = {bool} False
    'Vout' (286907200) = {float} 42.339127
    'Vref' (286907424) = {float} 1.970438
    'Iout' (286907328) = {float} 0.31180954
    'IoutRAW' (286904896) = {float} 0.045125003
    'Temp' (286907520) = {float} 49.013733
    'SetPoint' (286962736) = {float} 42.316467
    'ICR' (286904960) = {int} 1160
    'OCR' (286904736) = {int} 1130
    'runtime' (286906304) = {int} 1897
    'livetime' (286962776) = {int} 1896
    'sattime' (286905280) = {int} 0
    'incnt' (286907776) = {int} 2198310
    'outcnt' (286907744) = {int} 2127708
    'live' (286906400) = {float} 0.974138
    'dead' (286907392) = {float} 0.025862
    'mca_running' (286962816) = {int} 1
    'mca_status' (286962856) = {int} 0
```

In order to read particular value

```
I1 = ispector_sdk("192.168.50.2")

ChStatus = I1.getChannelStatus()
print(ChStatus["ICR"])
```

```
getSystemStatus(self):
```

Read system status

Return: Dictionary with system status

```
getWave(self):
```

Return list of array of array containing the waveform information

In order to extract a column of the matrix use numpy matrix and select one column (ie column 0 is analog values)

```
WaveMatrix = I1.getWave()

A = np.array(WaveMatrix)
wave_track = A[:,0]
```

Each data element is and array of 7 columns:

1) [0] Analog data
2) [1] Trigger pulse
3) [2] Charge Integration Window
4) [3] PSD Tail integration Window
5) [4] Baseline restorer status
6) [5] Pile up rejector discard
7) [6] Pile up inhibition

```
getSpectrum(self):
```

Read spectrum

Return: Array with 4096 spectrum bins

```
resetSpectrum(self):
```

Reset the spectrum in Ispector memory

Return: NONE

```
def runSpectrum(self):
```

Start spectrum acquisition

Return: NONE

```
def stopSpectrum(self):
```

Stop spectrum acquisition

Return: NONE

C# SDK

An SDK for C# language is available. The SDK use HTTP API communication to interface with the Ispector.

The SDK requires Newtonsoft JSON module. It will be automatically downloaded from NuGet at compiling time.

The SDK include a DLL library and a C# example.

The DLL can be imported in any programming language supporting C# (.NET) dll including VB.NET, Labview, Matlab

DOWNLOAD THE SDK:

Python SDK files can be download from Nuclear Instruments Github
https://github.com/NuclearInstruments/IspectorSDK-CSHARP

or cloned with git:

```
git clone https://github.com/NuclearInstruments/IspectorSDK-CSHARP.git
```

The function in C# SDK are the same of Python SDK. Refers to Python SDK for usage guide