FALCON

RESTful API Reference

Revision 24

Table of Contents

Changelog

Date	Author	Rev.	Log message
2017-01-19	Stanislav Ravas	12	- merge operation modes and their setpoints into single setpoint set - fix nomenclature - add luminaire current setpoint measurement - document SSR measurements and commands - update data model schema
2017-02-17	Stanislav Ravas	13	- drop command C_PNT***i***_BALLAST***j***_GROUP_SET - superseded with C_PNT***i***_BALLAST***j***_GROUP_ASSIGN (To assign exactly one group, call this command with group address as value. To resign all groups, call this command with value -1)
2017-02-26	Stanislav Ravas	14	Add alarm system API
2017-08-11	Stanislav Ravas	15	Add measurements and commands related to motion lighting settings
2018-02-15	Stanislav Ravas	16	Add Lumicharger API
2018-03-27	Stanislav Ravas	17	Add preliminary LUMiCOM API
2018-04-05	Stanislav Ravas	18	Add LUMiCOM serial parameters setup command
2018-06-12	Stanislav Ravas	19	- add controller-level alternative interrogation commands - add range interrogation commands - add lumicom group transmit command
2018-09-02	Stanislav Ravas	20	- add system measurements and commands section - add panter connector state PERMANENT_ERROR - add modulator measurement M_PNT***i***_POWER_CMD_QUEUE_LENGTH
2019-03-04	Stanislav Ravas	21	- add modulator measurement M_PNT***i***_CMD_SCHEDULE_LENGTH - add panter connector states SUSPEND, INROGEN_ALT_WAIT_READY, INROGEN_ALT_WAIT_STARTED, INROGEN_ALT_WAIT_COMPLETE, INROGEN_ALT_READ_LEVEL_INFO - drop panter connector state INIT_3
2019-04-16	Stanislav Ravas	22	Add digital I/O measurements & commands
2019-09-09	Stanislav Ravas	23	Alarm archive API
2020-07-29	Stanislav Ravas	24	Scene control API

Introduction

Falcon unit is built on the OpenDAF data acquisition framework. OpenDAF provides access to some of runtime entities via built-in HTTP server. Measurements and commands are of particular interest. HTTP server is available on port 80 and 443 (if https is configured). Only HTTP basic authentification is supported. URI hierarchy is sumarized in the table below.

URI hierarchy

Path	HTTP method	Description
MEASUREMENTS & COMMANDS		
/measurements/	GET	Retrieve full measurements image. Name filter can be applied via *names* parameter
/measurements/:name	GET	Retrieve single measurement image
/commands/	GET	Retrieve full commands image. Name filter can be applied via *names* parameter
/commands/:name	GET	Retrieve single measurement image
/commands/:name	PUT	Write command / setpoint **[public]**
ALARMS		
/alarms/	GET	Retrieve status of all alarms. Name filter can be applied via *names* parameter
/alarms/:name	GET	Retrieve single alarm status
/alarms/:name/activate	POST	Activate alarm
/alarms/:name/deactivate	POST	Deactivate alarm
/alarms/:name/acknowledge	POST	Acknowledge alarm
ALARM ARCHIVE		
/archive/alarms/:name/:ts/:te	GET	Retrieve history of alarm *name* since *ts* until *te*
SCHEDULER & ASTROCLOCK		
/kvds/falcon.scheduler.json	POST	Install new scheduler model **[public]**
/kvds/falcon.scheduler.json	GET	Retrieve current scheduler model **[public]**
/kvds/falcon.astroclock.json	POST	Install new astronomous clock model **[public]**
/kvds/falcon.astroclock.json	GET	Retrieve current astronomous clock model **[public]**
/astroclock/sunrise	GET	Retrieve today sunrise time (format HH:MM)
/astroclock/sunset	GET	Retrieve today sunset time (format HH:MM)
DATA MODEL & STATE		
/kvds/falcon.state.json	GET	Retrieve current data model state **[public]**
/kvds/falcon.state.json	POST	Set new data model state **[public]**

/kvds/falcon.model.json	GET	Retrieve current data model **[public]**
/kvds/falcon.model-copy.json	GET	Retrieve data model working copy **[public]**
/kvds/falcon.model-copy.json	POST	Overwrite data model working copy **[public]**
/kvds/falcon.model-copy.json	DELETE	Delete data model working copy **[public]**
/kvds/falcon.commit.json	POST	Write & commit new data model **[public]**

[public] - public authentification required
[privileged] - privileged authentification required

Measurements & Commands

System measurements

Name syntax	Data-type	Description
M_EST_CURRENT_L1	float	Estimated current on phase L1
M_EST_CURRENT_L2	float	Estimated current on phase L2
M_EST_CURRENT_L3	float	Estimated current on phase L3
M_N_POWER_CMDS_QUEUED	integer	Total number of power commands queued on all modulators

System commands

Name syntax	Datatype	Description
C_DISABLE_ALARMS	binary	Disable alarm activation

Modulator measurements

Name syntax	**Data-type**	**Description**
M_PNT***i***_SP	integer	Last broadcast setpoint on controller ***i***
M_PNT***i***_CONTROLLER_STATE	integer	State of the controller on controller ***i***, see Panter documentation
M_PNT***i***_CONTROLLER_VERSION	integer	Version of the controller on controller ***i***, see Panter documentation
M_PNT***i***_METER_COUNTER	double	Readout of the meter associated with controller ***i*** [kWh]
M_PNT***i***_ACTIVE_PHASES	integer	Number of active phases on controller ***i***
M_PNT***i***_MODULATOR1_TEMPERATURE	integer	Controller ***i*** modulator 1 temperature [°C]
M_PNT***i***_MODULATOR2_TEMPERATURE	integer	Controller ***i*** modulator 2 temperature [°C]
M_PNT***i***_MODULATOR3_TEMPERATURE	integer	Controller ***i*** modulator 3 temperature [°C]
M_PNT***i***_UPTIME	integer	Controller ***i*** uptime [s]
M_PNT***i***_VOLTAGE	integer	Controller ***i*** line voltage [V]
M_PNT***i***_MEASURED_CURRENT	float	Controller ***i*** measured current [A], updated on state transitions 4->0 and 4->5
M_PNT***i***_INROGEN_ID_RESULT	integer	Controller ***i*** ID interrogation result, updated on state transitions 4->0 and 4->5 (see Lumibox datasheet)
M_PNT***i***_INROGEN_ID_RESULT_ID	integer	Controller ***i*** ID interrogated during latest finished interrogation, updated on state transitions 4->0 and 4->5

M_PNT***i***_FSM_STATE	integer	Contoller ***i*** connector FSM state. Possible values:
		`0 (RESET)`
		`1 (SUSPEND)`
		`5 (PERMANENT_ERROR)`
		`10 (INIT_0)`
		`11 (INIT_1)`
		`12 (INIT_2)`
		`20 (READY)`
		`30 (INT_WAIT_COMPLETE)`
		`40 (MOD_WAIT_READY)`
		`41 (MOD_WAIT_STARTED)`
		`42 (MOD_WAIT_COMPLETE)`
		`50 (MEAS_WAIT_READY)`
		`51 (MEAS_WAIT_STARTED)`
		`52 (MEAS_WAIT_COMPLETE)`
		`53 (MEAS_READ_CURRENT)`
		`60 (INROGEN_WAIT_READY)`
		`61 (INROGEN_WAIT_STARTED)`
		`62 (INROGEN_WAIT_COMPLETE)`
		`63 (INROGEN_READ_ID_INFO)`
		`70 (INROGEN_ALT_WAIT_READY)`
		`71 (INROGEN_ALT_WAIT_STARTED)`
		`72 (INROGEN_ALT_WAIT_COMPLETE)`
		`73 (INROGEN_ALT_READ_LEVEL_INFO)`

M_PNT***i***_CMD_SCHEDULE_LENGTH	integer	Controller ***i*** number of commands scheduled to be queued
M_PNT***i***_CMD_QUEUE_LENGTH	integer	Controller ***i*** command queue length
M_PNT***i***_POWER_CMD_QUEUE_LENGTH	integer	Controller ***i*** number of power commands in the queue

M_PNT***i***_MOTION_ENABLE	binary	Latest motion enabled value as emitted
M_PNT***i***_MOTION_LEVEL	integer	Latest motion level value emitted [0-100 %]

M_PNT***i***_MOTION_LEVEL_TYPE	integer	Latest motion level type emitted. Possible values:
		`0 (direct)`
		`1 (additional)`

M_PNT***i***_MOTION_HOLD_ON_TIME	integer	Latest motion hold on time emitted [s]. Number is always dividable by 5.
M_PNT***i***_SCENE_ACT	integer	Current scene active, 0 = no scene active
M_PNT***i***_SCENE***k***_CONF	string	Current configuration of scene ***k***. See “Scene control”

Modulator commands

Name syntax	**Datatype**	**Description**
C_PNT***i***_SP_SET	integer	Broadcast setpoint on controller ***i***
C_PNT***i***_BALLAST_GROUPS_CLEAR	integer	Remove the ballast from all groups on controller ***i***
C_PNT***i***_ACTIVE_PHASES_SET	integer	Set number of active phases on controller ***i***
C_PNT***i***_METER_COUNTER_SET	double	Preload meter associated with controller ***i***
C_PNT***i***_CURRENT_TRIGGER	binary	Trigger current measurement on controller ***I*** by writing *true*
C_PNT***i***_INROGEN_ID_TRIGGER	integer	Interrogate ballast status on controller ***i.*** Ballast ID is passed as command value.
C_PNT***i***_INROGEN_IDS_TRIGGER	integer	Interrogate status from ID range. Command value to be written contains both first and last node ID and shall be calculated like this: $V = 65536 \times ID_{LAST} + ID_{FIRST}$
C_PNT***i***_INROGEN_ALT_ID_TRIGGER	integer	Interrogate ballast status and setpoint on controller ***i.*** See C_PNT***i***_INROGEN_ID_TRIGGER
C_PNT***i***_INROGEN_ALT_IDS_TRIGGER	integer	Interrogate status and level from ID range. See C_PNT***i***_INROGEN_IDS_TRIGGER
C_PNT***i***_INROGEN_AUTO_ID_TRIGGER	integer	Interrogate status or staus and level, depending on controller firmware version. See C_PNT***i***_INROGEN_ID_TRIGGER, C_PNT***i***_INROGEN_ALT_ID_TRIGGER
C_PNT***i***_INROGEN_AUTO_IDS_TRIGGER	integer	Interrogate status or status and level from ID range, depending on controller firmware version. See C_PNT***i***_INROGEN_IDS_TRIGGER, C_PNT***i***_INROGEN_ALT_IDS_TRIGGER
C_PNT***i***_INROGEN_IDS_ALL_TRIGGER	binary	Interrogate status or status and level depending on controller firmware version from all ballasts configured. Activate by writing value 1.
C_PNT***i***_MOTION_ENABLE	binary	Enable motion staging command
C_PNT***i***_MOTION_LEVEL	integer	Motion level 0-100 % staging command
C_PNT***i***_MOTION_LEVEL_TYPE	integer	Motion level type staging command. Possible values: `0 (direct)`

		`1 (additional)`
C_PNT***i***_MOTION_HOLD_ON_TIME	integer	Hold on time [s] staging command. Value will be rounded to the nearest greater integer dividable by 5.
C_PNT***i***_MOTION_SET	binary	When `true` is written, request is built from staging commands and executed.
C_PNT***i***_CHARGER_MEASURE_CURRENT	integer	Measure lumicharger node current (write node ID)
C_PNT***i***_CHARGER_QUERY_STATUS	integer	Query lumicharger node status (write node ID)
C_PNT***i***_CHARGER_QUERY_STATUSES	integer	Query multiple lumicharger nodes status. Command value to be written contains both first and last node ID and shall be calculated like this: $V = 65536 \times ID_{LAST} + ID_{FIRST}$
C_PNT***i***_CHARGER_QUERY_STATUS_ALL	boolean	Query all lumicharger nodes using range queries
C_PNT***i***_SCENE_TEST_SET	string	Test scene settings. See “Scene control”
C_PNT***i***_SCENE_ACT_SET	integer	Activate scene number written
C_PNT***i***_SCENE_REMOVE	integer	Remove all ballasts from scene number written
C_PNT***i***_SCENE***k***_CONF_SET	string	Setup scene ***k*** on all ballast. See “Scene control”

Group measurements

Name syntax	Datatype	Description
M_PNT***i***_GROUP***j***_SP	integer	Latest setpoint applied to controller ***i*** group ***j*** (includes setpoint chained from parent modulator)
M_PNT***i***_GROUP***j***_SCENE_ACT	integer	Current scene active, 0 = no scene active
M_PNT***i***_GROUP***j***_SCENE***k***_CONF	string	Current configuration of scene ***k***. See “Scene control”

Group commands

Name syntax	Datatype	Description
C_PNT***i***_GROUP***j***_ADD	integer	Add ballast to group ***j*** on controller ***i***
C_PNT***i***_GROUP***j***_REMOVE	integer	Remove ballast from group ***j*** on controller ***i***
C_PNT***i***_GROUP***j***_SP_SET	integer	Controller ***i*** group ***j*** setpoint
C_PNT***i***_GROUP***j***_COM_TX	string	Transmit string. String shall contain 2-digit hexadecimal ascii-formatted bytes. i.e. string “AA55102030” will cause bytes 0xAA, 0x55, 0x10, 0x20, 0x30 to be transmitted.
C_PNT***i***_GROUP***j***_SCENE_TEST_SET	string	Test scene settings. See “Scene control”
C_PNT***i***_GROUP***j***_SCENE_ACT_SET	integer	Activate scene number written
C_PNT***i***_GROUP***j***_SCENE_REMOVE	integer	Remove group ballasts from scene number written
C_PNT***i***_GROUP***j***_SCENE***k***_CONF_SET	string	Setup scene ***k*** on group ballast. See “Scene control”

Luminaire measurements

Name syntax	Datatype	Description
M_PNT***i***_BALLAST***j***_SP	integer	Latest setpoint applied to ballast ***j*** on controller ***i*** (includes setpoint chained from parent modulator)
M_PNT***i***_BALLAST***j***_GROUP	integer	Group assigned to ballast ***j*** on controller ***i***
M_PNT***i***_BALLAST***j***_STATUS	integer	Ballast ***j*** on controller ***i*** interrogated status, full and short byte combined (see Lumibox datasheet)
M_PNT***i***_BALLAST***j***_SCENE_ACT	integer	Current scene active, 0 = no scene active
M_PNT***i***_BALLAST***j***_SCENE***k***_CONF	string	Current configuration of scene ***k***. See “Scene control”

Scene control measurements are generated only if the luminaire has `scene` feature enabled.

Luminaire commands

Name syntax	Datatype	Description
C_PNT***i***_BALLAST***j***_SP_SET	integer	ballast ***j*** on controller ***i*** setpoint
C_PNT***i***_BALLAST***j***_GROUP_ASSIGN	integer	Assign and save exactly one group to ballast ***j*** on controller ***I***
C_PNT***i***_BALLAST***j***_SCENE_TEST_SET	string	Test scene settings. See “Scene control”
C_PNT***i***_BALLAST***j***_SCENE_ACT_SET	integer	Activate scene number written
C_PNT***i***_BALLAST***j***_SCENE_REMOVE	integer	Remove ballast from scene number written
C_PNT***i***_BALLAST***j***_SCENE***k***_CONF_SET	string	Setup scene ***k*** on ballast. See “Scene control”

Scene control commands are generated only if the luminaire has `scene` feature enabled.

Scene control

Scenes are configured and tested by writing scene configuration strings into scene configuration or testing commands.

On luminaires scene control measurements and commands are generated only if the luminaire has `scenes` feature enabled.

Scene configuration string syntax:

```
<level>%[,tc=<value>|xy=<x>,<y>|rgbwaf=<r>,<g>,<b>,<w>,<a>,<f>]
```

Examples:

1. Only set level to 50%:

```
50%
```

2. Set level to 75% and tunable-white color temperature to 4500K:

```
75%,tc=4500
```

3. Set level to 10% and color using CIE 1931 colorspace to 10000,25000:

```
10%,xy=10000,25000
```

4. Set level to 100% and color using RGBWAF colorspace:

```
100%,rgbwaf=255,128,55,50,150,150
```

Lumicharger node measurements

Name syntax	Datatype	Description
M_PNT***i***_CHARGER***j***_STATUS	integer	Charger status. Possible values: 0 - vehicle not connected 1 - receiving current limit 2 - vehicle connected 3 - charging 4 - charging disabled by control system 5 - overcurrent error
M_PNT***i***_CHARGER***j***_CURRENT_L1	float	Measured charger current on phase L1
M_PNT***i***_CHARGER***j***_CURRENT_L2	float	Measured charger current on phase L2
M_PNT***i***_CHARGER***j***_CURRENT_L3	float	Measured charger current on phase L3
M_PNT***i***_CHARGER***j***_SP_CURRENT	integer	Latest charging current setpoint
M_PNT***i***_CHARGER***j***_SP_LIMIT	integer	Latest current limit setpoint
M_PNT***i***_CHARGER***j***_SP_ENABLE	boolean	Latest enable setpoint

Lumicharger node commands

Name syntax	Datatype	Description
C_PNT***i***_CHARGER***j***_CURRENT_SET	integer	Set charging current regulator setpoint. Valid in charger states: 2, 3
C_PNT***i***_CHARGER***j***_LIMIT_SET	integer	Set charger current limit (according to circuit breaker) Valid in charger states: 1
C_PNT***i***_CHARGER***j***_ENABLE_SET	boolean	Enable/disable charging Valid in charger states: 0,1,2,3,4,5
C_PNT***i***_CHARGER***j***_CURRENT_MEASURE	boolean	Measure charging current
C_PNT***i***_CHARGER***j***_STATUS_QUERY	boolean	Query current status

LUMiCOM node measurements

LUMiCOM measurements are generated only if the luminaire has `com` feature enabled.

Name syntax	Datatype	Description
M_PNT***i***_COM***j***_STATUS	integer	Communication status. Possible values: 0 - OK 1 - transfer to LUMiCOM node error 2 - remote node receive error 3 - transfer from LUMiCOM node error
M_PNT***i***_COM***j***_RX	string	Data received. String contains bytes received represented as 2-digit hexadecimal ascii numbers. i.e. string “AA55102030” means bytes 0xAA, 0x55, 0x10, 0x20, 0x30 were received.

LUMiCOM node commands

LUMiCOM commands are generated only if the luminaire has `com` feature enabled.

Name syntax	Datatype	Description
C_PNT***i***_COM***j***_RX_LENGTH_SET	integer	Set expected response length
C_PNT***i***_COM***j***_TX	string	Transmit string. String shall contain 2-digit hexadecimal ascii-formatted bytes. i.e. string “AA55102030” will cause bytes 0xAA, 0x55, 0x10, 0x20, 0x30 to be transmitted.
C_PNT***i***_COM***j***_SETUP	integer	Configure serial transmission/reception parameters. Value is a 32-bit number composed of: bits 31:24 - baudrate 0 - 2400 1 - 9600 2 - 19200 3 - 38400 4 - 57600 5 - 115200 6 - 128000 7 - 250000 bits 23:16 - parity ? bits 15:8 - stopbits ? bits 7:0 - timeout 0 - 500ms 1 - 1s 2 - 5s 3 - 10s

Powermeter measurements

Name syntax	Datatype	Description
M_PM_E1	float	Active energy on L1 [Wh]
M_PM_E2	float	Active energy on L2 [Wh]
M_PM_E3	float	Active energy on L3 [Wh]
M_PM_EQ1	float	Reactive energy on L1 [Wh]
M_PM_EQ2	float	Reactive energy on L2 [Wh]
M_PM_EQ3	float	Reactive energy on L3 [Wh]
M_PM_ES1	float	Apparent energy on L1 [Wh]
M_PM_ES2	float	Apparent energy on L2 [Wh]
M_PM_ES3	float	Apparent energy on L3 [Wh]
M_PM_F1	float	Frequency on L1 [Hz]
M_PM_F2	float	Frequency on L2 [Hz]
M_PM_F3	float	Frequency on L3 [Hz]
M_PM_I1	float	Current on L1 [A]
M_PM_I2	float	Current on L2 [A]
M_PM_I3	float	Current on L3 [A]
M_PM_P1	float	Active power on L1 [W]
M_PM_P2	float	Active power on L2 [W]
M_PM_P3	float	Active power on L3 [W]
M_PM_P	float	Combined active power [W]
M_PM_Q1	float	Reactive power on L1 [W]
M_PM_Q2	float	Reactive power on L2 [W]
M_PM_Q3	float	Reactive power on L3 [W]
M_PM_Q	float	Combined reactive power [W]
M_PM_S1	float	Apparent power on L1 [W]
M_PM_S2	float	Apparent power on L2 [W]
M_PM_S3	float	Apparent power on L3 [W]
M_PM_S	float	Combined apparent power [W]
M_PM_PF1	float	Power factor on L1 [-]
M_PM_PF2	float	Power factor on L2 [-]
M_PM_PF3	float	Power factor on L3 [-]
M_PM_PF	float	Combined power factor [-]
M_PM_PFLC1	float	Power factor L&C on L1 [-]
M_PM_PFLC2	float	Power factor L&C on L2 [-]
M_PM_PFLC3	float	Power factor L&C on L3 [-]

M_PM_PFLC	float	Combined power factor L&C [-]
M_PM_V1	float	Voltage L1-N [V]
M_PM_V2	float	Voltage L2-N [V]
M_PM_V3	float	Voltage L3-N [V]
M_PM_V12	float	Voltage L1-L2 [V]
M_PM_V23	float	Voltage L2-L3 [V]
M_PM_V31	float	Voltage L3-L1 [V]

Digital I/O measurements

Name syntax	Datatype	Description
M_DI***i***	boolean	Digital input ***i*** state
M_SSR***i***_SP	boolean	Digital output ***i*** latest setpoint
M_SSR***i***_RB	boolean	Digital output ***i*** readback

Digital I/O commands

Name syntax	Datatype	Description
C_SSR***i***_SP_SET	boolean	Digital output ***i*** setpoint

Alarm archive access

Alarm archive access is performed on one alarm a time using following query:

GET http://<server-address>/opendaf/archive/alarms/*:name*/*:ts*/*:te*[?format=<csv|json>]

where:

Parameter	**Description**	**Examples**
:name	Alarm name, must conform to C identifier syntax	ALARM_0
:ts	Time interval start, expressed in seconds since UNIX epoch	1568057048
:te	Time interval end, expressed in seconds since UNIX epoch	1568067048
format	Requested result format	json csv

Result format is either csv or json, as selected by *format* parameter.

JSON schema:

```
{
  "title": "archive query alarm history results",
  "type": "object",
  "patternProperties": {
    "[_a-zA-Z][_a-zA-Z0-9]": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "t": {
            "title": "timestamp",
            "type": "array",
            "items": [
              {
                "title": "seconds since the epoch", "type": "integer",
                "minimum": 0
              },
              {
                "title": "microseconds", "type": "integer",
                "minimum": 0, "maximum": 999999
              }
            ]
          },
          "state": { "enum": ["INACT_ACK","ACT_UNACK","ACT_ACK","INACT_UNACK"] },
          "description": { "type": "string" },
          "authority": { "type": "string" }
        },
        "additionalProperties": false,
        "required": [ "t", "state", "description", "authority" ]
      }
    }
  }
}
```

CSV header:
timestamp,state,description,authority

Measurement & command examples

All GET requests return result in a JSON format. Here are the examples taken from a Falcon unit configured with Panter on address 32, with groups 1 and 2 and ballasts 1, 2 and 3:

Request	Response

GET /measurements/PNT32_METER_COUNTER	{
	"name" : "PNT1_METER_COUNTER",
	"address" : "PNT1_METER_COUNTER",
	"datatype" : "d",
	"raw-datatype" : "d",
	"eu-range" : [null,null],
	"raw-range" : [null,null],
	"vtq" : ["d30.5",[1442242212, 504232],192]
	}

GET /commands/C_PNT32_GROUP2_L_SETPOINT	{
	"name": "C_PNT32_GROUP2_L_SETPOINT",
	"address": "PNT32_GROUP2_L_SETPOINT",
	"datatype": "i",
	"raw-datatype": "i",
	"eu-range": [null, null],
	"raw-range": [null, null],
	"vt": ["i0", [0, 0]]
	}

GET /measurements/?names=PNT32_CONTROLLER_VERSION,PNT32_METER_READOUT	{
	"PNT32_CONTROLLER_VERSION": {
	"name": "PNT32_CONTROLLER_VERSION",
	"address": "32/mv_controller_version",
	"datatype": "i",
	"raw-datatype": "i",
	"eu-range": [null, null],
	"raw-range": [null, null],
	"vtq": ["i80", [1427372529, 70000], 192]
	},
	"PNT32_METER_READOUT": {
	"name": "PNT32_METER_READOUT",
	"address": "32/mv_meter_readout",
	"datatype": "l",
	"raw-datatype": "l",
	"eu-range": [null, null],
	"raw-range": [null, null],
	"vtq": ["l0", [1427372529, 103000], 192]
	}
	}

Formats of certain JSON elements:

JSON key	**JSON value datatype**	**JSON value description**
“datatype”	string	Object datatype – initial letter of **b**inary, **q**uaternary, **i**nteger, **l**ong, **f**loat, **d**ouble, **s**tring
“eu-range”	array [low, high]	Object range in engineering units
“raw-range”	array [low, high]	Object range in raw units
“vt”	array [value, timestamp]	Command value with timestamp. Value is a string containing datatype prefix and measurement value itself (i.e. i-10, f3.5, l20483038575). Timestamp is an array containing number of seconds and microseconds from the epoch (see

		gettimeofday() manpage).
“vtq”	array [value, timestamp, quality]	Measurement value with timestamp, value and quality. Value and timestamp are the same sa in “vt”. Quality is a number containing quality value according to OPC DA 2.05 standard. Basic measurement quality can be assessed this way: 1. mask quality value with 0xC0 2. when masked value equals to 0xC0 => quality is good 3. when masked value equals to 0x40 => quality is uncertain (i.e. out of range) 4. when masked value equals to 0x00 => quality is bad (i.e. not connected)

Scheduler

Scheduler jobs are configured via JSON configuration file. The schema is:

```
{
  "title": "Scheduler schema",
  "type": "object",
  "properties": {
    "jobs": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "schedule": {
            "type": "string",
            "description": "cron schedule"
          },
          "enabled": {
            "type": "boolean"
          },
          "action": {
            "type": "object",
            "oneOf": [
              {
                "properties": {
                  "class": {
                    "enum": ["PanterJobAction"]
                  },
                  "address": {
                    "type": "integer",
                    "minimum": 1,
                    "maximum": 254
                  },
                  "setpoint": {
                    "type": "integer",
                    "minimum": 0,
                    "maximum": 100
                  }
                },
                "required": ["class", "address", "setpoint"]
              },
              {
                "properties": {
                  "class": {
                    "enum": ["GroupJobAction"]
                  },
                  "panterAddress": {
                    "type": "integer",
                    "minimum": 1,
                    "maximum": 254
                  },
                  "address": {
                    "type": "integer",
                    "minimum": 1,
                    "maximum": 239
                  },
                  "setpoint": {
                    "type": "integer",
                    "minimum": 0,
                    "maximum": 100
                  }
                },
                "required": ["class", "panterAddress", "address", "setpoint"]
              },
              {
                "properties": {
                  "class": {
                    "enum": ["SSRJobAction"]
                  },
                  "index": {
                    "type": "integer",
                    "minimum": 1
                  },
                  "setpoint": {
                    "type": "integer",
                    "minimum": 0,
                    "maximum": 100
                  }
                },
                "required": ["class", "index", "setpoint"]
              }
            ]
```

```
          }
        },
        "required": ["schedule", "enabled", "action"]
      }
    }
  }
}
```

Astroclock

Astroclock jobs are configured via JSON configuration file. The schema is:

```
{
  "title": "Astroclock schema",
  "type": "object",
  "properties": {
    "jobs": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "event": {
            "type": "integer",
            "enum": [0, 1],
            "description": "originating astronomical event (0 = sunrise, 1 = sunset)"
          },
          "offset": {
            "type": "integer",
            "description": "offset from the origin event [m]"
          },
          "enabled": {
            "type": "boolean"
          },
          "action": {
            "type": "object",
            "oneOf": [
              {
                "properties": {
                  "class": {
                    "enum": ["PanterJobAction"]
                  },
                  "address": {
                    "type": "integer",
                    "minimum": 1,
                    "maximum": 254
                  },
                  "setpoint": {
                    "type": "integer",
                    "minimum": 0,
                    "maximum": 100
                  }
                },
                "required": ["class", "address", "setpoint"]
              },
              {
                "properties": {
                  "class": {
                    "enum": ["GroupJobAction"]
                  },
                  "panterAddress": {
                    "type": "integer",
                    "minimum": 1,
                    "maximum": 254
                  },
                  "address": {
                    "type": "integer",
                    "minimum": 1,
                    "maximum": 239
                  },
                  "setpoint": {
                    "type": "integer",
                    "minimum": 0,
                    "maximum": 100
                  }
                },
                "required": ["class", "panterAddress", "address", "setpoint"]
              },
              {
                "properties": {
                  "class": {
                    "enum": ["SSRJobAction"]
                  },
                  "index": {
                    "type": "integer",
                    "minimum": 1
                  },
                  "setpoint": {
                    "type": "integer",
                    "minimum": 0,
                    "maximum": 100
                  }
                },
                "required": ["class", "index", "setpoint"]
              }
```

```
            ]
          }
        },
        "required": ["schedule", "enabled", "action"]
      }
    }
  }
}
```

Data model & state

Data model is a JSON document with following schema:

```
{
  "title":  "Falcon data model schema",
  "type":    "object",
  "properties":  {
    "io": {
      "type": "object",
      "properties": {
        "di": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "id": {
                "enum": ["CASE_OPENED", "OVERRIDE", "LUX_SENSOR"]
              },
              "polarity": {
                "enum": ["NO","NC"]
              }
            },
            "required": ["id"]
          }
        }
      }
    },
    "ssrs": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "index" : {
            "type": "integer",
            "minimum": 1
          },
          "measurements": {
            "type": "object",
            "properties": {
              "sp" : { "type": "string" }
            },
            "required": [ "last_setpoint", "mode", "sp_man", "sp_auto", "sp_over", "sp_sched" ]
          },
          "commands": {
            "type": "object",
            "properties": {
              "sp_set" : { "type": "string" },
              "sp_exec" : { "type": "string" }
            },
            "required": [ "sp_set", "sp_exec", "mode_set",
              "sp_man_set", "sp_auto_set", "sp_over_set", "sp_sched_set" ]
          }
        },
        "required": [ "measurements", "commands" ]
      }
    },
    "panters": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "address" : {
            "type": "integer",
            "minimum": 1,
```

```
          "maximum": 254
        },
        "groups" : {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "address" : {
                "type": "integer",
                "minimum": 0,
                "maximum": 239
              },
              "name": {
                "type": "string"
              },
              "measurements": {
                "type": "object"
              },
              "commands": {
                "type": "object"
              }
            },
            "required": ["address"]
          }
        }
      },
      "required": ["address"]
    }
  }
}
```

State is stored in a JSON document with schema:

```
{
        "title":       "Falcon state schema",
        "type":        "object",
        "properties":  {
                "current": {
                        "type": "string",
                        "enum": ["RUN", "EDIT"]
                }
        },
        "required": ["current"]
}
```

cURL examples:

cURL commandline	**Description**
curl -X PUT -u op:compact http://192.168.11.116/commands/C_PNT32_BROADCAST?value=i10	Set all ballasts on controller 32 to 10%.
curl -X PUT -u op:compact http://192.168.11.116/commands/C_PNT32_GROUP1_SETPOINT?value=i25	Set ballasts in group 1 on controller 32 to 25%.
curl -X PUT -u op:compact http://192.168.11.116/commands/C_PNT32_BALLAST2_SETPOINT?value=i100	Set ballast 2 on controller 32 to 100%.
curl http://192.168.11.116/measurements/M_PNT32_GROUP1_L_SETPOINT	Get measurement containing last controller 32 group 1 setpoint.
curl http://192.168.11.116/measurements/M_PNT32_CONTROLLER_VERSION	Get version of controller 32.
curl -X PUT -u op:compact http://192.168.11.116/commands/C_PNT32_BALLAST_GROUPS_CLEAR?value=i2 curl -X PUT -u op:compact http://192.168.11.116/commands/C_PNT32_GROUP1_ADD?value=i2	Assign ballast 2 to and only to group 1 without remembering (association won't be shown in the web UI).
curl -X PUT -u op:compact http://192.168.11.116/commands/C_PNT32_BALLAST_GROUPS_CLEAR?value=i2 curl -X PUT -u op:compact http://192.168.11.116/commands/C_PNT32_GROUP1_ADD?value=i2 curl -X PUT -u op:compact http://192.168.11.116/commands/C_PNT32_BALLAST2_GROUP?value=i1	Assign ballast 2 to and only to group 1 with remembering (association would be shown in the web UI).