EDS

Enterprise Data Server

REST API Documentation

TRANSITION TECHNOLOGIES ADVANCED SOLUTIONS

Contents

General information

This document presents practical usage of EDS REST API in python 3.7+.

In all examples python requests module is used.

/login endpoint creates session and returns session id. Client should provide session token using Authorization header in the form of *Authorization: Bearer <token>*

api_url variable contains a core url in which there are specified parameters like server ip, REST API port and version. An Example:
api_url = 'http://127.0.0.1:43084/api/v1/'
Request url is api_url + 'endpointName'

Errors handling

API returns HTTP status codes in responses for various errors.

Responses with "application/json" content type header will contain application specific error codes listed below. Response body will contain JSON with following structure:

Error body:

```
{
  "error": 1,                  // error code
  "message": "License invalid"  // error message
}
```

Error codes:

- 0 -
- 1 - License invalid
- 2 - License expired
- 3 - Mobile license invalid
- 4 - Mobile license expired
- 5 - License max session reached
- 6 - Object server not connected
- 7 - Authentication error
- 8 - Authentication timout

- 9 - User not in 'webapi' nor 'admin' group
- 10 - Cannot remove own permission
- 11 - Invalid request ID
- 30 - Points' data not synchronized yet.
- 31 - Point not found
- 32 - Invalid point type
- 33 - Invalid point value
- 34 - No point alter permission
- 35 - Cannot operate point
- 36 - No operate permission
- 37 - Operate point error
- 400 - Bad request
- 401 - Unauthorized
- 403 - Access denied
- 404 - Not found
- 500 - Internal server error

Authorization functions

Authenticate

Checks user name and password. Does not create session.

POST REQUEST

```
>>> import requests
>>> api_url = 'http://localhost:43084/api/v1/'
>>> request_url = api_url + 'authenticate'
>>> data = {'username' : 'admin', 'password' : ''}
>>> request = requests.post(request_url, json = data)
>>> print(response.json())
```

RESPONSE

```
{"authenticated": true}
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "username": {"type": "string"},
    "password": {"type": "string"}
  },
  "required": ["username", "password"]
}
```

Login function

Creates new session.

/login endpoint creates session and returns session id. Client should provide session token using Authorization header in the form of *Authorization: Bearer <token>*

POST REQUEST

```
>>> import requests
>>> api_url = 'http://localhost:43084/api/v1/'
>>> request_url = api_url + 'login'
>>> data = {'username' : 'admin', 'password' : '', 'type' : 'rest client'}
>>> response = requests.post(request_url, json = data)
>>> print(response.json())
```

RESPONSE

```
{'sessionId': '18258da8-5919-54fe-8145-90baf59c5922'}
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "username": {"type": "string"},
    "password": {"type": "string"},
    "type": {"type": "string"}
  },
  "required": ["username", "password"]
}
```

Logout function

Closes session.

POST REQUEST

```
>>> import requests
>>> api_url = 'http://localhost:43084/api/v1/'
>>> request_url = api_url + 'login'
>>> data = {'username' : 'admin', 'password' : '', 'type' : 'rest client'}
>>> response = requests.post(request_url, json = data)
>>> token = json.loads(response.text)

>>> request_url = api_url + 'logout'
>>> headers = {'Authorization' : 'Bearer {}'.format(token['sessionId'])}
>>> response = requests.post(request_url, headers = headers)
```

RESPONSE

```
<Response [200]>
```

Ping function

Checks session and updates its timeout.

GET REQUEST

```
>>> import requests
>>> api_url = 'http://localhost:43084/api/v1/'
>>> request_url = api_url + 'login'
>>> data = {'username' : 'admin', 'password' : '', 'type' : 'rest client'}
>>> response = requests.post(request_url, json = data)
>>> token = json.loads(response.text)
>>> request_url = api_url + 'ping'
>>> headers = {'Authorization' : 'Bearer {}'.format(token['sessionId'])}
>>> response = requests.get(request_url, headers = headers)
```

RESPONSE

```
<Response [200]>
```

Points functions

Points query

Query points matching selected criteria or a pre-defined filter.

GET - query points matching a pre-defined filter.
Parameters:

- "source" - source name
- "order" - semicolon-separated list of point field names e.g. "zd;iess;sid" or "aux;-sid". Adding "-" before a field name reverses the order. Full list of fields that may appear in the order parameter: sid, value, dts, tss, at, atss, quality, iess, desc, rt, zd, idcs, ar, ap, aux, un, dp, artd, ard, tb, bb, hl, ll, dun, ddp, dartd, dard, dtb, dbb, dhl, dll, sd, rd.
- "fields" - (optional) comma-separated list of fields returned in response
- "page" - (optional) page number
- "pagesize" - (optional) points per page (default: 50, max: 1000)

GET REQUEST

```
>>> import requests
>>> api_url = 'http://localhost:43084/api/v1/'
>>> request_url = api_url + 'login'
>>> data = {'username' : 'admin', 'password' : '', 'type' : 'rest client'}
>>> response = requests.post(request_url, json = data)
>>> token = json.loads(response.text)

>>> headers = {'Authorization' : 'Bearer {}'.format(token['sessionId'])}
>>> source = 'zd1'
>>> filter = 'myfilter1'
>>> order = 'iess'
>>> query = '?source={}&filter={}&order={}'.format(source, filter, order)
>>> request_url = api_url + 'points/query' + query
>>> request = requests.get(request_url, headers=headers)
```

RESPONSE

```
see "points/query" ...
```

POST - query points matching the filters.

Point value for ANALOG and DOUBLE points can be a string with "Inf", "-Inf", "NaN" value.
The "order" parameter should be a list of point field names e.g. ["zd", "iess", "sid"] or ["aux", "-sid"]. Adding "-" before a field name reverses the order. Full list of fields that may appear in the order parameter: sid, value, dts, tss, at, atss, quality, iess, desc, rt, zd, idcs, ar, ap, aux, un, dp, artd, ard, tb, bb, hl, ll sd, rd. }

POST REQUEST

```
>>> import requests
>>> api_url = 'http://localhost:43084/api/v1/'
>>> request_url = api_url + 'login'
>>> data = {'username' : 'admin', 'password' : '', 'type' : 'rest client'}
>>> response = requests.post(request_url, json = data)
>>> token = json.loads(response.text)
>>> headers = {'Authorization' : 'Bearer {}'.format(token['sessionId'])}

>>> request_url = api_url + 'points/query'
>>> query = {
...     'filters' : [{
...         'zd' : ['Your_ZD'],
...         'tg' : [0, 1]
...     }],
...     'order' : ['iess']
... }
>>> request = requests.post(request_url, headers = headers, json = query)
```

RESPONSE

```
{
  "points": [{
    "sid": 1,                 // (optional) sid
    "iess": "iess1",          // (optional) iess
    "idcs": "idcs1",          // (optional) idcs
    "zd": "zd1",              // (optional) source
    "rt": "ANALOG",           // (optional) record type
    "value": 1.23,            // (optional) value
    "quality": "GOOD",        // (optional) quality
    "ts": 1603305959,         // (optional) last read timestamp
    "lts": 1603305959,        // (optional) long term modification timestamp
    "tss": 0,                 // (optional) timestamp shift
    "at": 0,                  // (optional) alarm timestamp
    "atss": 0,                // (optional) alarm timestamp shift
    "desc": "desc1",          // (optional) description
    "st": 0,                  // (optional) status bits
    "xst1": 0,                // (optional) external status 1 bits
    "xst2": 0,                // (optional) external status 2 bits
    "xst3": 0,                // (optional) external status 3 bits
    "ar": "LONG_TERM",        // (optional) archiving type
    "artd": "PCT_RANGE",      // (optional) archiving deadband type
    "ard": 1.0,               // (optional) archiving deadband
    "sg": [0, 1],             // (optional) security groups
    "tg": [],                 // (optional) technological groups
    "df": 0,                  // (optional) definition flags
    "ap": 0,                  // (optional) alarm priority
    "aux": "",                // (optional) aux
    "un": "m",                // (optional) unit
    "dp": 2,                  // (optional) display precision
    "tb": 2,                  // (optional) top bar
    "bb": 0,                  // (optional) bottom bar
    "hl": 2,                  // (optional) high limit
    "ll": 0,                  // (optional) low limit
    "sd": "NORMAL",           // (optional) set description (BINARY)
    "rd": "ALARM",            // (optional) reset description (BINARY)
    "foreground": 123,        // (optional) foreground color
    "background": 123         // (optional) background color
  }],
  "matchCount": 1,         // matched elements count
  "totalCount": 100000,   // total point count
}
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "filters": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "sid": {"type": "array", "items": {"type": "integer", "minimum":
0}},
          "iess": {"type": "array", "items": {"type": "string"}},
          "iessRe": {"type": "string"},
          "idcs": {"type": "array", "items": {"type": "string"}},
          "idcsRe": {"type": "string"},
          "zd": {"type": "array", "items": {"type": "string"}},
          "zdRe": {"type": "string"},
          "descRe": {"type": "string"},
          "auxRe": {"type": "string"},
          "rt": {"type": "array", "items": {"type": "string", "enum":
["ANALOG", "DOUBLE", "BINARY", "PACKED", "INT64", "TEXT"]}},
          "ts": {
            "type": "object",
            "properties": {"from": {"type": "integer"},"till": {"type":
"integer"}},
            "required": ["from", "till"]
          },
          "quality": {"type": "array", "items": {"type": "string", "enum":
["GOOD", "FAIR", "POOR", "BAD", "NONE"]}},
          "stSet": {"type": "integer", "minimum": 0},
          "stUnset": {"type": "integer", "minimum": 0},
          "dfSet": {"type": "integer", "minimum": 0},
          "dfUnset": {"type": "integer", "minimum": 0},
          "xst1Set": {"type": "integer", "minimum": 0},
          "xst1Unset": {"type": "integer", "minimum": 0},
          "xst2Set": {"type": "integer", "minimum": 0},
          "xst2Unset": {"type": "integer", "minimum": 0},
          "xst3Set": {"type": "integer", "minimum": 0},
          "xst3Unset": {"type": "integer", "minimum": 0},
          "ar": {"type": "array", "items": {"type": "string", "enum":
["NONE", "LONG_TERM", "EXTERNAL", "FILLIN"]}},
          "artd": {
            "type": "array",
            "items": {
              "type": "string",
              "enum": ["NONE", "FLOW", "LOG", "PCT_RANGE", "POWER",
"RADIATION", "RATIO", "STANDARD",
                       "GEOMETRIC1", "GEOMETRIC3", "TRAPEZOIDAL",
"LTS_BASED", "TIME_INTERVAL", "TEST"]
            }
          },
          "sg": {"type": "array", "items": {"type": "integer", "minimum":
0}},
          "tg": {"type": "array", "items": {"type": "integer", "minimum":
0}},
          "ap": {"type": "array", "items": {"type": "integer", "minimum":
0}}
        }
      }
    },
    "order": {"type": "array", "items": {"type": "string"}},
    "page": {"type": "integer", "minimum": 1},
    "pagesize": {"type": "integer", "minimum": 1},
    "fields": {
      "type": "array",
      "items": {
        "type": "string",
        "enum": ["sid", "iess", "idcs", "zd", "rt", "value", "quality",
"ts", "lts", "tss", "at", "ats",
                 "desc", "st", "xst1", "xst2", "xst3", "ar", "artd", "sg",
"tg", "df", "ap", "aux", "un",
                 "dp", "ard", "tb", "bb", "hl", "ll", "sd", "rd",
"foreground", "background"]
      }
    }
  }
}
```

Points export

Export points matching selected criteria or a pre-defined filter.

GET - export points matching a pre-defined filter.

Parameters:

- "zd" - zd name
- "iess" - iess simple regular expression
- "idcs" - idcs simple regular expression
- "desc" - desc simple regular expression
- "aux" - aux simple regular expression
- "ac" - ac simple regular expression
- "rt" - record type list: ANALOG, DOUBLE, BINARY, PACKED, INT64, TEXT
- "order" - semicolon-separated list of point field names e.g. "zd;iess;sid" or "aux;-sid". Adding "-" before a field name reverses the order. Full list of fields that may appear in the order parameter: sid, value, dts, tss, at, atss, quality, iess, desc, rt, zd, idcs, ar, ap, aux, un, dp, artd, ard, tb, bb, hl, ll, dun, ddp, dartd, dard, dtb, dbb, dhl, dll, sd, rd.
- "separator" - space is the default separator
- "encoding" - one of: iso-8859-1, iso8859-1, iso-latin-1, latin-1, latin1, iso-8859-2, iso8859-2, iso-latin-2, latin-2, latin2, windows-1250, windows1250, windows-1251, windows1251, windows-1252, windows1252, utf-8, utf8, utf-16be, utf16be, utf-16le, utf16le, unicode
- "flags" - (optional) 0x0002 - export with SIDs
- "page" - (optional) page number
- "pagesize" - (optional) points per page (default: 1000000, max: 1000000)

GET REQUEST

```
>>> zd = 'Your_ZD'
>>> iess = '^A'
>>> order = 'iess'
>>> query = '?zd={}&iess={}&order={}'.format(zd, iess, order)
>>> request_url = api_url + 'points/export' + query
>>> request = requests.get(request_url, headers=header)
```

RESPONSE

```
see "points/export" ...
```

POST - export points matching the filters.

Point value for ANALOG and DOUBLE points can be a string with "Inf", "-Inf", "NaN" value.
The "order" parameter should be a list of point field names e.g. ["zd", "iess", "sid"] or ["aux", "-sid"]. Adding "-" before a field name reverses the order. Full list of fields that may appear in the order parameter: sid, value, dts, tss, at, atss, quality, iess, desc, rt, zd, idcs, ar, ap, aux, un, dp, artd, ard, tb, bb, hl, ll sd, rd. }

POST REQUEST

```
>>> request_url = api_url + 'points/export'
>>> query = {
...     'filters' : [{
...         'zd' : ['Your_ZD'],
...         'tg' : [0, 1]
...     }],
...     'order' : ['iess']
... }
>>> request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

```
# ENCODING='utf-8'
# SAVE_TIMESTAMP=1680275316.0 SAVE_TIMESTAMP_LOCAL=2023-03-31_17:08:36
# POINTS_AND_CONFIGS_MD5=1f410743b28acf59bf94a3b6f10a3d06
# POINTS_TIMESTAMP=1677784798.4 POINTS_TIMESTAMP_LOCAL=2023-03-02_20:19:58
# CONFIGS_TIMESTAMP=1647328961.7 CONFIGS_TIMESTAMP_LOCAL=2022-03-
15_08:22:42
CONFIG NAME='DEFAULT' AR='F' ARTD='V' ARD=0 ARP='
SELECTIVE_SOURCE_MERGE=\'0\' TREND_FORCEABLE=\'0\' USE_IDCS=\'1\''
SOURCE='L'
POINT RT=ANALOG SID=45631 DF=0x0124 IESS='UHN17CF201_M' ZD='Ovation'
IDCS='UHN17CF201_M' DESC='F.spalin w kominie D_MODYF' AUX='T=sine'
AC='DEFAULT' AP=0x88 TG='0;1' SG='0;3' UN='kNm3/h' DP=0 TB=5500 BB=0
HL=5500 LL=0
POINT RT=ANALOG SID=45632 DF=0x0124 IESS='UHN18CQ203' ZD='Ovation'
IDCS='UHN18CQ203' DESC='NOx (6%) w spalinach - kom.C' AUX='T=sine'
AC='DEFAULT' AP=0x88 TG='0;1' SG='0;3' UN='mg/m3' DP=0 TB=600 BB=0 HL=600
LL=0
POINT RT=ANALOG SID=45633 DF=0x0124 IESS='UHN18CT201' ZD='Ovation'
IDCS='UHN18CT201' DESC='TEMPERATURA SPALIN KOMIN C' AUX='T=sine'
AC='DEFAULT' AP=0x88 TG='0;1' SG='0;3' UN='ST.C' DP=1 TB=200 BB=0 HL=200
LL=0
```

```
POINT RT=ANALOG SID=45634 DF=0x0124 IESS='UHN19CQ203' ZD='Ovation'
IDCS='UHN19CQ203' DESC='NOx (6%) w spalinach - kom.D' AUX='T=sine'
AC='DEFAULT' AP=0x88 TG='0;1' SG='0;3' UN='mg/m3' DP=0 TB=600 BB=0 HL=600
LL=0
POINT RT=ANALOG SID=45635 DF=0x0124 IESS='UHN19CT201' ZD='Ovation'
IDCS='UHN19CT201' DESC='TEMPERATURA SPALIN KOMIN D' AUX='T=sine'
AC='DEFAULT' AP=0x88 TG='0;1' SG='0;3' UN='ST.C' DP=1 TB=200 BB=0 HL=200
LL=0
# END
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "filters": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "sid": {"type": "array", "items": {"type": "integer", "minimum":
0}},
          "iess": {"type": "array", "items": {"type": "string"}},
          "iessRe": {"type": "string"},
          "idcs": {"type": "array", "items": {"type": "string"}},
          "idcsRe": {"type": "string"},
          "zd": {"type": "array", "items": {"type": "string"}},
          "zdRe": {"type": "string"},
          "descRe": {"type": "string"},
          "auxRe": {"type": "string"},
          "rt": {"type": "array", "items": {"type": "string", "enum":
["ANALOG", "DOUBLE", "BINARY", "PACKED", "INT64", "TEXT"]}},
          "ts": {
            "type": "object",
            "properties": {"from": {"type": "integer"},"till": {"type":
"integer"}},
            "required": ["from", "till"]
          },
          "quality": {"type": "array", "items": {"type": "string", "enum":
["GOOD", "FAIR", "POOR", "BAD", "NONE"]}},
          "stSet": {"type": "integer", "minimum": 0},
          "stUnset": {"type": "integer", "minimum": 0},
          "dfSet": {"type": "integer", "minimum": 0},
          "dfUnset": {"type": "integer", "minimum": 0},
          "xst1Set": {"type": "integer", "minimum": 0},
          "xst1Unset": {"type": "integer", "minimum": 0},
          "xst2Set": {"type": "integer", "minimum": 0},
          "xst2Unset": {"type": "integer", "minimum": 0},
          "xst3Set": {"type": "integer", "minimum": 0},
          "xst3Unset": {"type": "integer", "minimum": 0},
          "ac": {"type": "string"},
          "acRe": {"type": "string"},
          "ar": {"type": "array", "items": {"type": "string", "enum":
["NONE", "LONG_TERM", "EXTERNAL", "FILLIN"]}},
          "artd": {
            "type": "array",
            "items": {
              "type": "string",
              "enum": ["NONE", "FLOW", "LOG", "PCT_RANGE", "POWER",
"RADIATION", "RATIO", "STANDARD",
                       "GEOMETRIC1", "GEOMETRIC3", "TRAPEZOIDAL",
"LTS_BASED", "TIME_INTERVAL", "TEST"]
            }
          },
          "sg": {"type": "array", "items": {"type": "integer", "minimum":
0}},
          "tg": {"type": "array", "items": {"type": "integer", "minimum":
0}},
          "ap": {"type": "array", "items": {"type": "integer", "minimum":
0}}
        }
      }
    },
    "order": {"type": "array", "items": {"type": "string"}},
    "separator": {"type": "string"},
    "encoding": {"type": "string"},
    "flags": {"type": "integer", "minimum": 0},
    "page": {"type": "integer", "minimum": 1},
    "pagesize": {"type": "integer", "minimum": 1},
    "fields": {
      "type": "array",
      "items": {
        "type": "string",
        "enum": ["sid", "iess", "idcs", "zd", "rt", "value", "quality",
"ts", "lts", "tss", "at", "ats",
                 "desc", "st", "xst1", "xst2", "xst3", "ar", "artd", "sg",
"tg", "df", "ap", "aux", "un",
                 "dp", "ard", "tb", "bb", "hl", "ll", "sd", "rd",
"foreground", "background"]
      }
    }
  }
}
```

Points sources

Returns a list of all point sources.
Note: point sources and object sources might be completely different, even though in many cases they have similar names.

GET REQUEST

```
>>> request_url = api_url + 'points/sources'
>>> request = requests.get(request_url, headers=header)
```

RESPONSE

```
["zd1", "zd2"]
```

Points operate

Changes current points values.

POST REQUEST

```
>>> request_url = api_url + 'points/operate'
>>> query = [{
...     'sid' : 1,
...     'iess' : 'iess1',
...     'idcs' : 'idcs1',
...     'zd' : 'zd1',
...     'value' : 1.1
...     'quality' : 'GOOD',
... }]
>>> request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

```
<Response [200]>
```

SCHEMA

```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "sid": {"type": "integer", "minimum": 0},
      "iess": {"type": "string"},
      "idcs": {"type": "string"},
      "zd": {"type": "string"},
      "value": {"type": ["number", "string", "boolean"]},
      "quality": {"type": "string", "enum": ["GOOD", "FAIR", "POOR", "BAD",
"NONE"]}
    }
  }
}
```

Points publish

Automatically refresh point values for 'duration' time after the call.

POST REQUEST

```
>>> request_url = api_url + 'points/publish'
>>> query = [{
...     'sid' : 1,
...     'iess' : 'iess1',
...     'iess' : 'idcs1',
...     'zd' : 'zd1',
...     'value' : 1.1,
...     'quality' : 'GOOD',
...     'duration' : 60
... }]
>>> request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

<Response [200]>

SCHEMA

```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "sid": {"type": "integer", "minimum": 0},
      "iess": {"type": "string"},
      "idcs": {"type": "string"},
      "zd": {"type": "string"},
      "value": {"type": ["number", "string", "boolean"]},
      "quality": {"type": "string", "enum": ["GOOD", "FAIR", "POOR", "BAD",
"NONE"]},
      "duration": {"type": "integer", "minimum": 1}
    }
  }
}
```

Points unpublish

Automatically refresh point values for 'duration' time after the call.

POST REQUEST

```
>>> request_url = api_url + 'points/unpublish'
>>> query = [{
...     'sid' : 1,
...     'iess' : 'iess1'
...     'idcs' : 'idcs1'
...     'zd' : 'zd1'
... }]
>>> request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

<Response [200]>

SCHEMA

```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "sid": {"type": "integer", "minimum": 0},
      "iess": {"type": "string"},
      "idcs": {"type": "string"},
      "zd": {"type": "string"}
    }
  }
}
```

Objects functions

Objects query

Query objects metadata matching selected criteria.
The "order" parameter should be a list of fields names, for example: ["file", "-name"]. Adding "-" before a field name reverses the order.
Full list of fields that may appear in the order parameter: file, name, sourceName, sourceId, modified, sg, tg, md5.

POST REQUEST

```
>>> request_url = api_url + 'objects/query'
>>> query = {
...     'filters' : [{
...         'id' : [1, 2],
...         'fileRe' : '^abc',
...         'nameRe' : '^abc',
...         'sourceNameRe' : '^abc',
...         'sourceId' : [1],
...         'modified' : {
...             'from' : 1603305950,
...             'till' : 1603305959
},
...         'sg' : [0, 1]
...         'tg' : [0, 1]
...         'md5' : 'abcdefg'
...     }],
...     'order' : ['file'],
...     'fields' : ['file', 'name'],
...     'page' : 2,
...     'pagesize' : 50
... }
>>> request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

```
{
  "objects": [{
    "id": 1,                // (optional) id
    "file": "file1.edf",    // (optional) file
    "name": "name1",        // (optional) name
    "sourceName": "src1",   // (optional) source name
    "sourceId": 3,          // (optional) source id
    "modified": 1603305959, // (optional) modification timestamp
    "sg": [0, 1],           // (optional) security groups
    "tg": [],               // (optional) technological groups
    "md5": "abcd"           // (optional) md5 sum
  }],
  "matchCount": 1,          // matched count
  "totalCount": 100000,      // total count
}
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "filters": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "id": {"type": "array", "items": {"type": "integer", "minimum":
0}},
          "fileRe": {"type": "string"},
          "nameRe": {"type": "string"},
          "sourceNameRe": {"type": "string"},
          "sourceId": {"type": "array", "items": {"type": "integer",
"minimum": 0}},
          "modified": {
            "type": "object",
            "properties": {"from": {"type": "integer"}, "till": {"type":
"integer"}},
            "required": ["from", "till"]
          },
          "sg": {"type": "array", "items": {"type": "integer", "minimum":
0}},
          "tg": {"type": "array", "items": {"type": "integer", "minimum":
0}},
          "md5": {"type": "array", "items": {"type": "string"}}
        }
      }
    },
    "order": {"type": "array", "items": {"type": "string"}},
    "page": {"type": "integer", "minimum": 1},
    "pagesize": {"type": "integer", "minimum": 1},
    "fields": {
      "type": "array",
      "items": {
        "type": "string",
        "enum": ["id", "file", "name", "sourceName", "sourceId",
"modified", "sg", "tg", "md5"]
      }
    }
  }
}
```

Objects

POST - create object

PUT - update object

DELETE - delete object

POST REQUEST

```
>>> request_url = api_url + 'objects'
>>> query = [{
...     'sourceId' : 123,
...     'file' : 'myfile.edf',
...     'name' : 'myobject',
...     'sg' : [0, 1],
...     'tg' : []
... }]
>>> request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

```
[{"id":new_object_id}]
```

SCHEMA

```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "sourceId": {"type": "integer", "minimum": 0},
      "file": {"type": "string"},
      "name": {"type": "string"},
      "sg": {"type": "array", "items": {"type": "integer", "minimum": 0}},
      "tg": {"type": "array", "items": {"type": "integer", "minimum": 0}}
    },
    "required": ["sourceId", "file"]
  }
}
```

PUT REQUEST

```
>>> request_url = api_url + 'objects'
>>> query = [{
...     'id' : 5,
...     'name' : 'myobject'
...     'sg' : [0, 1],
...     'tg' : [],
...     'force' : false
... }]
>>> request = requests.put(request_url, headers = header, json = query)
```

RESPONSE

```
<Response [200]>
```

SCHEMA

```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "id": {"type": "integer", "minimum": 0},
      "name": {"type": "string"},
      "sg": {"type": "array", "items": {"type": "integer", "minimum": 0}},
      "tg": {"type": "array", "items": {"type": "integer", "minimum": 0}},
      "force": {"type": "boolean"}
    },
    "required": ["id"]
  }
}
```

DELETE REQUEST

```
>>> request_url = api_url + 'objects'
>>> query = [{
...     'id' : 1
... }]
>>> request = requests.delete(request_url, headers = header, json = query)
```

RESPONSE

```
<Response [200]>
```

SCHEMA

```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "id": {"type": "integer", "minimum": 0}
    },
    "required": ["id"]
  }
}
```

Objects sources query

Returns a list of all object sources if filter is not set or all object sources matching filter. Note: point sources and object sources might be completely different, even though in many cases they have similar names.
The order parameter should be a list of field names, in example: ["id"] or ["location", "-name"]. Adding "-" before a field name reverses the order.
Full list of fields that may appear in the order parameter: id, name, desc, sg, tg, location, host, prefix, suffix, options.

POST REQUEST

```
>>> request_url = api_url + 'objects/sources/query'
>>> query = {
...     'filters' : [{
...         'id' : '^abc',
...         'nameRe' : '^abc',
...         'descRe' : '^abc',
...         'modified' : '^abc',
...         'sg' : [0, 1],
...         'tg' : [0, 1],
...         'location' : 'DB',
...         'hostRe' : '^abc',
...         'prefixRe' : '^abc',
...         'suffixRe' : '^abc',
...         'optionsSet' : 0,
...         'optionsUnset' : 0
...     }],
...     'order' : ['id', 'name'],
...     'fields' : ['name', 'desc'],
...     'page' : 2,
...     'pagesize' : 50
... }
>>> request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

```
{
  "sources": [{
    "id": 1,
    "name": "iess1",
    "desc": "desc1",
    "sg": [0, 1],
    "tg": [],
    "location": "DB",
    "host": "",
    "prefix": "",
    "suffix": "",
    "options": 0
  }],
  "matchCount": 1,
  "totalCount": 100000,
}
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "filters": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "id": {"type": "array", "items": {"type": "integer", "minimum": 0}},
          "nameRe": {"type": "string"},
          "descRe": {"type": "string"},
          "modified": {"type": "integer", "minimum": 0},
          "sg": {"type": "array", "items": {"type": "integer", "minimum": 0}},
          "tg": {"type": "array", "items": {"type": "integer", "minimum": 0}},
          "location": {"type": "string", "enum": ["DISK", "FTP", "DB"]},
          "hostRe": {"type": "string"},
          "prefixRe": {"type": "string"},
          "suffixRe": {"type": "string"},
          "optionsSet": {"type": "integer", "minimum": 0},
          "optionsUnset": {"type": "integer", "minimum": 0}
        }
      }
    },
    "order": {"type": "array", "items": {"type": "string"}},
    "page": {"type": "integer", "minimum": 1},
    "pagesize": {"type": "integer", "minimum": 1},
    "fields": {
      "type": "array",
      "items": {
        "type": "string",
        "enum": ["id", "name", "desc", "modified", "location", "host", "prefix", "suffix"]
      }
    }
  }
}
```

Objects sources

POST - create object sources

PUT - update object sources

DELETE - delete object sources

POST REQUEST

```
>>> request_url = api_url + 'objects/sources'
>>> query = [{
...     'name' : 'mysource',
...     'desc' : '',
...     'sg' : [0, 1]
...     'tg' : [],
...     'location' : 'DISK',
...     'host' : '',
...     'prefix' : '',
...     'suffix' : 'a@NET1',
... }]
>>> request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

```
[{"id":new_object_source_id}]
```

SCHEMA

```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "name": {"type": "string"},
      "desc": {"type": "string"},
      "sg": {"type": "array", "items": {"type": "integer", "minimum": 0}},
      "tg": {"type": "array", "items": {"type": "integer", "minimum": 0}},
      "location": {"type": "string", "enum": ["DISK", "FTP", "DB"]},
      "host": {"type": "string"},
      "prefix": {"type": "string"},
      "suffix": {"type": "string"}
    },
    "required": ["name"]
  }
}
```

PUT REQUEST

```
>>> request_url = api_url + 'objects/sources'
>>> query = [{
...     'id' : 123,
...     'name' : 'myobject',
...     'desc' : '',
...     'sg' : [0, 1],
...     'tg' : [],
...     'location' : 'DISK',
...     'host' : '',
...     'prefix' : '',
...     'suffix' : '.a@NET1',
...     'force' : False,
... }]
>>> request = requests.put(request_url, headers = header, json = query)
```

RESPONSE

<Response [200]>

SCHEMA

```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "id": {"type": "integer", "minimum": 0},
      "name": {"type": "string"},
      "desc": {"type": "string"},
      "sg": {"type": "array", "items": {"type": "integer", "minimum": 0}},
      "tg": {"type": "array", "items": {"type": "integer", "minimum": 0}},
      "location": {"type": "string", "enum": ["DISK", "FTP", "DB"]},
      "host": {"type": "string"},
      "prefix": {"type": "string"},
      "suffix": {"type": "string"},
      "force": {"type": "boolean"}
    },
    "required": ["id"]
  }
}
```

DELETE REQUEST

```
>>> request_url = api_url + 'objects/sources'
>>> query = [{
...     'id' : 5
... }]
>>> request = requests.delete(request_url, headers = header, json = query)
```

RESPONSE

<Response [200]>

SCHEMA

```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "id": {"type": "integer", "minimum": 0}
    },
    "required": ["id"]
  }
}
```

Diagrams functions and GFX API

Graphics server diagram rendering API v1.

/diagram/open endpoint creates diagram rendering session and returns session id. It returns root URL in form http://GFX-HOST:GFX-PORT/UUID/. The UUID should be extracted from this link and a new link for communication with webapi should be created using it. *An Example:*
diagram_url = 'http://127.0.0.1:43090/e9d5d1fa-25a8-48f1-b653-37d7435e289d/'
gfx_api_url = 'http://127.0.0.1:43090/api/v1/e9d5d1fa-25a8-48f1-b653-37d7435e289d/'

Request url is gfx_api_url + 'GFXendpointName'

Diagram open

Starts diagram rendering session on Graphics Server (GfxSrv). Returns root URL in form http://GFX-HOST:GFX-PORT/UUID/. The UUID should be extracted from this link and a new link for communication with webapi should be created using it.

POST REQUEST

```
>>> open_diagram_url = api_url + 'diagram/open'
>>> query = {
...     'source' : 'YourSource',
...     'file' : '1000.edf',
...     'httpUrl' : 'http://127.0.0.1:43090/'
... }
>>> open_diagram = requests.post(open_diagram_url, headers = header, json = query)
>>> diagram_url = json.loads(open_diagram.text)['url']
>>> session_id = diagram_url.split('/')[-2]
>>> gfx_api_url = 'http://127.0.0.1:43090/' + session_id + '/'
```

RESPONSE

```
{"url":"http://127.0.0.1:43090/e9d5d1fa-25a8-48f1-b653-37d7435e289d/"}
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "source": {"type": "string"},
    "file": {"type": "string"},
    "redrawInterval": {"type": "integer", "minimum": 1, "maximum": 60},
    "httpUrl": {"type": "string"},
    "previousUrl": {"type": "string"},
    "pointGroup": {"type": "string"}
  },
  "required": ["source", "file"]
}
```

Sessions status

Returns multiple sessions tag number.

Tag number will be necessary to use other functions.

POST REQUEST

```
>>> request_url = 'http://192.168.49.89:43090/api/v1/' + 'sessions/status'
>>> query = [{
...     'id' : session_id
... }]
>>> request = requests.post(request_url, headers = header, json = query)
>>> tag = json.loads(request.text)[0]['tag']
```

RESPONSE

```
[{'tag': tag_number}]
```

SCHEMA

```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "id": {"type": "string"},
      "skipNext": {"type": "boolean"}
    },
    "required": ["id"]
  }
}
```

Diagram

meta.json

Returns metadata for the current tag

GET REQUEST

```
>>> request_url = gfx_api_url + 'meta.json?tag=' + str(tag)
>>> request = requests.get(request_url, headers=header)
```

RESPONSE

```
{
  "history": { "nextCount": 0, "previousCount": 0 },
  "main": {
    "error": "",
    "source": "src1",
    "fileName": "diag1234.edf",
    "groupName": "",
    "currentView": { "x": 0, "y": 0, "width": 1, "height": 1 },
    "resolution": { "width": 1904, "height": 537 },
    "diagramLoading": false,
    "sequenceNo": 0,
    "repaintNo": 0,
    "actions": [],
    "activeAreas": [
      { "action": { "type": "diagram", "target": "MAIN", "source": "Unit1",
"file": "2303.edf","groupName": "0" }, "x": 509,"y": 168, "width": 81,
"height": 41 },
      { "action": { "type": "internal", "areaId": "262487344" }, "x": 796,
"y": 258, "width": 142, "height": 22 },
      { "action": { "type": "points", "areaId": "262470576", "points":
["pt1", "pt2"] }, "x": 996, "y": 478, "width": 242, "height": 52 }
    ]
  },
  "subwindow": {},
  "window": {},
  "extraWindows": []
}
```

<role>.png

Returns diagram image for requested tag.

Supported files:

- "main.png" - main image
- "subwindow.png" - subwindow image
- "window.png" - active window
- "window[idx].png" - window at index

GET REQUEST

```
>>> request_url = gfx_api_url + 'main.png?tag=' + str(tag)
>>> request = requests.get(request_url, headers=header)
```

RESPONSE

PNG image content

Actions

Click

Handle diagram click

POST REQUEST

```
>>> request_url = gfx_api_url + 'click'
>>> query = {
...     'role' : 'MAIN',
...     'areaId' : '586813307248'
... }
>>> request = requests.post(request_url, json = query)
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "role": {"type": "string", "enum": ["MAIN", "WINDOW", "SUBWINDOW"]},
    "areaId": {"type": "string"},
    "windowId": {"type": "integer", "minimum": 0}
  },
  "required": ["role", "areaId"]
}
```

Close

Close diagram

POST REQUEST

```
>>> request_url = gfx_api_url + 'close'
>>> query = {
...     'role' : 'MAIN'
... }
>>> request = requests.post(request_url, json = query)
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "role": {"type": "string", "enum": ["MAIN", "WINDOW", "SUBWINDOW"]},
    "windowId": {"type": "integer", "minimum": 0}
  },
  "required": ["role"]
}
```

Entry

Set entry field value

POST REQUEST

```
>>> request_url = gfx_api_url + 'entry'
>>> query = {
...     'role' : 'MAIN',
...     'areaId' : '586791398784',
...     'value' : '5'
... }
>>> request = requests.post(request_url, json = query)
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "role": {"type": "string", "enum": ["MAIN", "WINDOW", "SUBWINDOW"]},
    "areaId": {"type": "string"},
    "value": {"type": "string"},
    "windowId": {"type": "integer", "minimum": 0}
  },
  "required": ["role", "areaId", "value"]
}
```

Navigate

Navigate the diagram

POST REQUEST

```
>>> request_url = gfx_api_url + 'navigate'
>>> query = {
...     'navigation' : 'HOME'
... }
>>> request = requests.post(request_url, json = query)
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "navigation": {"type": "string", "enum": ["HOME", "PREV", "NEXT", "UP",
"DOWN", "LEFT", "RIGHT"]}
  },
  "required": ["navigation"]
}
```

Resize

Resize diagram resolution

POST REQUEST

```
>>> request_url = gfx_api_url + 'resize'
>>> query = {
...     'role' : 'MAIN',
...     'width' : 1280,
...     'height' : 720
... }
>>> request = requests.post(request_url, json = query)
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "role": {"type": "string", "enum": ["MAIN", "WINDOW", "SUBWINDOW"]},
    "width": {"type": "integer", "minimum": 1},
    "height": {"type": "integer", "minimum": 1},
    "windowId": {"type": "integer", "minimum": 0}
  },
  "required": ["role", "width", "height"]
}
```

Viewport

Set diagram viewport.
'x' and 'y' represent relative position on diagram where [0.0, 0.0] is a top left diagram coordinate and [1.0, 1.0] is a bottom right diagram corner.

POST REQUEST

```
>>> request_url = gfx_api_url + 'viewport'
>>> query = {
...     'role' : 'WINDOW',
...     'topLeft' : {'x' : 0.1, 'y' : 0.9},
...     'bottomRight' : {'x' : 0.2, 'y' : 0.8}
... }
>>> request = requests.post(request_url, json = query)
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "role": {"type": "string", "enum": ["MAIN", "WINDOW", "SUBWINDOW"]},
    "topLeft": {
      "type": "object",
      "properties": {
        "x": {"type": "number", "minimum": 0, "maximum": 1},
        "y": {"type": "number", "minimum": 0, "maximum": 1}
      },
      "required": ["x", "y"]
    },
    "bottomRight": {
      "type": "object",
      "properties": {
        "x": {"type": "number", "minimum": 0, "maximum": 1},
        "y": {"type": "number", "minimum": 0, "maximum": 1}
      },
      "required": ["x", "y"]
    },
    "windowId": {"type": "integer", "minimum": 0}
  },
  "required": ["role", "topLeft", "bottomRight"]
}
```

Window Active

Set active window

POST REQUEST

```
>>> request_url = gfx_api_url + 'window/active'
>>> query = {
...     'windowId' : 2
... }
>>> request = requests.post(request_url, json = query)
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "windowId": {"type": "integer", "minimum": 0}
  },
  "required": ["windowId"]
}
```

Window Lock

Lock diagram window

POST REQUEST

```
>>> request_url = gfx_api_url + 'window/lock'
>>> query = {
...     'windowId' : 2,
...     "locked" : True
... }
>>> request = requests.post(request_url, json = query)
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "windowId": {"type": "integer", "minimum": 0},
    "locked": {"type": "boolean"}
  },
  "required": ["windowId", "locked"]
}
```

Window Open

Open window diagram

POST REQUEST

```
>>> request_url = gfx_api_url + 'window/open'
>>> query = {
...     'source' : 'src1',
...     'file' : '1000.edf'
... }
>>> request = requests.post(request_url, json = query)
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "source": {"type": "string"},
    "file": {"type": "string"},
    "pointGroup": {"type": "string"}
  },
  "required": ["source", "file"]
}
```

Requests

Requests

Long running requests status.

GET - get requests status

DELETE - drop requests

GET REQUEST

```
>>> query = '?ids=3214'
>>> request_url = api_url + 'requests' + query
>>> request = requests.get(request_url, headers=header)
```

RESPONSE

```
{
  "3214": {
    "status": "EXECUTING",   // status: "QUEUED", "EXECUTING", "SUCCESS" or
"FAILURE"
    "progress": 78.12,       // (optional) progress 0 - 100
    "message": ""             // (optional) error message
  },
}
```

DELETE REQUEST

```
>>> request_url = api_url + 'requests'
>>> query = [{
...     'id' : 1
... }]
>>> request = requests.delete(request_url, headers = header, json = query)
```

RESPONSE

```
<Response [200]>
```

SCHEMA

```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "id": {"type": "integer", "minimum": 1}
    },
    "required": ["id"]
  }
}
```

Trends

Trend

GET - Retrieves trend result

POST - Requests trend

Executes graphical trend. "pixelCount" is a trend width resolution. When requesting trend for long time range 1 pixel can correspond to e.g. 1 day, that's why trend request may return up to 5 samples per pixel (value at the start of the range, maximum, minimum, value at the end, hole marker). See also /api/v1/requests.

GET REQUEST

```
>>> query = '?id=3214'
>>> request_url = api_url + 'trend' + query
>>> request = requests.get(request_url, headers=header)
```

RESPONSE

```
[{
  "items": [  // list of items results
    [         // list of samples
      [
        1603305950,  // timestamp
        2.1,         // value
        "G",         // quality
        0,           // (optional) timestamp shift
        0,           // (optional) sample origin (default: DataSource):
                     //   0 - DataSource
                     //   1 - Filter
                     //   2 - Function
                     //   3 - Processor
                     //   4 - Test
                     //   5 - Other
        1,           // (optional) sample type (default: Regular):
                     //   0 - Regular
                     //   1 - RegularZeroOrder
                     //   2 - ShadeBegin
                     //   3 - ShadeEnd
      ],
      [1603305951, 2.2, "G"],
      [1603305952, 2.3, "G"]
    ],
    [
      [1603305950, 0.5, "G"]
    ]
  ],
  "averages": [2.15, 0.5], // (optional) averages available with "LAST"
chunk
  "status": "LAST"
}]
```

POST REQUEST

```
>>> request_url = api_url + 'trend'
>>> query = [{
...     'pointId' : {
...         'sid' : 42809,
...         'iess' : 'point42809.UNIT1@NET1'
...     },
...     'period' : {
...         'from' : 1603305950,
...         'till' : 1603305959
...     },
...     'pixelCount' : 1024,
...     'shadePriority' : 'DEFAULT'
... }]
>>> request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

```
{"id": 1234  // request id}
```

SCHEMA

```
{
  "type": "array",
  "items":  {
    "type": "object",
    "properties": {
      "pointId": {
        "type": "object",
        "properties": {
          "sid": {"type": "integer", "minimum": 0},
          "iess": {"type": "string"}
        }
      },
      "period": {
        "type": "object",
        "properties": {"from": {"type": "integer"}, "till": {"type":
"integer"}},
        "required": ["from", "till"]
      },
      "pixelCount": {"type": "integer", "minimum": 1},
      "shadePriority": {"type": "string", "enum": ["DEFAULT",
"REGULAR_OVER_SHADE", "SHADE_OVER_REGULAR", "REGULAR_ONLY", "SHADE_ONLY"]}
    },
    "required": ["pointId", "period", "pixelCount"]
  }
}
```

Tabular

GET - Retrieves tabular trend result

POST - Requests tabular trend

GET REQUEST

```
>>> query = '?id=1234'
>>> request_url = api_url + 'trend/tabular' + query
>>> request = requests.get(request_url, headers=header)
```

RESPONSE

```
[{
  "items": [  // list of items results
    [         // list of samples
      [
        1603305950,  // timestamp
        2.1,         // value
        "G",         // quality
        0,           // (optional) timestamp shift
      ],
      [1603305951, 2.3, "G"],
      [1603305952, 2.4, "G"]
    ],
    [
      [1603305950, 0.5, "G"]
    ]
  ],
  "status": "LAST"
}]
```

POST REQUEST

```
>>> request_url = api_url + 'trend/tabular'
>>> query = {
...     'period' : {
...         'from' : 1603305950,
...         'till' : 1603305959
...     },
...     'step' : 60,
...     'items' : [{
...         'pointId' : {
...             'sid' : 42809,
...             'iess' : 'OLIVER.COPE@NET1'
...         },
...         'shadePriority' : 'DEFAULT'
...     }]
... }
>>> request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

{"id": 1234 // request id}

SCHEMA

```
{
  "type": "object",
  "properties": {
    "period": {
      "type": "object",
      "properties": {"from": {"type": "integer"}, "till": {"type":
"integer"}},
      "required": ["from", "till"]
    },
    "step": {"type": "integer", "minimum": 1},
    "items": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "pointId": {
            "type": "object",
            "properties": {
              "sid": {"type": "integer", "minimum": 0},
              "iess": {"type": "string"}
            }
          },
          "shadePriority": {"type": "string", "enum": ["DEFAULT",
"REGULAR_OVER_SHADE", "SHADE_OVER_REGULAR", "REGULAR_ONLY", "SHADE_ONLY"]},
          "function": {"type": "string", "enum": [
            "AVG", "AVG_QUAL", "BETWEEN_TIME", "F_INTOOVER_DT",
"F_INTOUNDER_DT", "F_MAX_DT", "F_MIN_DT",
            "F_OVER_TIME", "F_UNDER_TIME", "INTG", "INTG_OVER",
"INTG_UNDER", "L_INTOOVER_DT", "L_INTOUNDER_DT",
            "L_MAX_DT", "L_MIN_DT", "L_OVER_TIME", "L_UNDER_TIME",
"MAX_VALUE", "MIN_VALUE", "OVER_TIME", "TIME",
            "TOGGLE", "TOGGLE_OVER", "TOGGLE_UNDER", "UNDER_TIME",
"VALUE"]},
          "params": {"type": "array", "items": {"type": "number"}}
        },
        "required": ["pointId"]
      }
    }
  },
  "required": ["period", "step", "items"]
}
```

Groups

Retrieves a list of trend groups migrated to the specified configuration version. If configuration version is unspecified, it defaults to the current version.

Parameters:

- "ver" - configuration version

REQUEST

```
>>> ver = '9.2.0.26'
>>> query = '?ver={}'.format(ver)
>>> request_url = api_url + 'trend/groups' + query
>>> request = requests.get(request_url, headers=header)
```

RESPONSE

```
{
 "Unit1": {
  "ovation_trends": {
   "": {
    "test-123456": {
     "": {
      "[@config_block@]": {
       "": {
        "id": "2",
        "version": "9.2.0.36"
       }
      },
      "chart": {
       "": {
        "0": {
         "S25B-U1-L.PCC@SFWMD": {
          "bit": "0",
          "colors": "FF0000 FFFF0000 FFFF0000 FFFF00",
          "id": "1",
          "regular_to_shade": "default",
          "yrang": "auto"
         }
        },
        "Configurations": {
         "": {
          "default": {
           "default": {
            "archive": "0",
            "autoScaleMarginPercentage": "0",
            "backgroundColor": "0",
            "description": "",
            "futureColor": "FFFFFF00",
            "minimumVerticalGridLines": "4",
            "numberOfGridLines": "7",
                        "numberOfSubGridLines": "1",
                        "pointXItemId": "-1",
                        "subwindows": {
                          "": {
                            "count": "0",
                            "layout": "quadrant",
                            "titles": {"": {"A": "", "B": "", "C": "", "D": ""}}
                          }
                        },
                        "thickLines": "0",
                        "timeRange": {
                          "": {
                            "endTime": "1637268223",
                            "startTime": "1637181823",
                            "range": "86400"
                          }
                        },
                        "type": "norm",
                        "verticalGridType": "1",
                        "verticalGridUnit": "10",
                        "viewerConfig": "1 0 0 0 1 1 0 0 0 0 0 0",
                        "viewerEventsConfig": "FF FFFF",
                        "viewerHaxisConfig": {"": {"0": "0 1 0", "1": "", "2": ""}}
                      }
                    }
                  }
                },
                "backgroundColor": "0",
                "futureColor": "FFFFFF00",
                "thickLines": "0",
                "timeRange": {
                  "": {
                    "endTime": "1637268223",
                    "startTime": "1637181823",
                    "range": "86400"
                  }
                },
                "type": "norm"
              }
            }
          }
        }
      }
    }
  }
}
```

Events

Read

GET - Retrieves trend result GET – Retrieves events read results

Parameters:

- "id" - request id
- "fields" - (optional) comma-separated list of fields returned in response. Full list of fields: "category", "type", "priority", "message", "sid", "iess", "floatValue", "intValue", "st", "ts", "tss", "aux", "foreground", "background"

GET REQUEST

```
>>> id = 1234
>>> query = '?id={}'.format(id)
>>> request_url = api_url + 'events/read' + query
>>> request = requests.get(request_url, headers=header)
```

RESPONSE

```
{
  "events": [{
    "category": "ALARM",
    "type": "ALARM_ANALOG,
    "priority": 5,
    "message": "abcd",
    "sid": 1234,
    "iess": "iess1",
    "floatValue": 5.23,
    "intValue": 32,
    "st": 1,
    "ts": 1603305950,
    "tss": 0,
    "aux": 0,
    "foreground": 123,
    "background": 123
  }]
}
```

POST - Requests event read.

All returned Events will be ordered in reverse chronological order. See also /api/v1/requests.

POST REQUEST

```
>>> request_url = api_url + 'events/read'
>>> query = {
...     'filter': {
...         'period': {
...             'from': 1699365200,
...             'till': 1699365520
...         },
...         'category': ['ALARM'],
...         'rt': ['ANALOG', 'DOUBLE'],
...         'priority': [1, 2],
...        'sg' : [3, 4],
...        'tg' : [5,6],
...        'zd': ['source1'],
...        'iessRe' : '^abc',
...        'messageRe' : '^abc',
...        'pointId': [{
...           'sid' : 1,
...           'iess': 'iess1'
...        }]
...     },
...     'maxCount': 50
... }
>>> request = requests.post(request_url, headers=header, json=query)
```

RESPONSE

```
{"id": 1234  // request id}
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "filter": {
      "type": "object",
      "properties": {
        "period": {
          "type": "object",
          "properties": {"from": {"type": "integer"}, "till": {"type": "integer"}},
          "required": ["from", "till"]
        },
        "category": {"type": "array", "items": {"type": "string", "enum": ["SYSTEM", "ALARM",
"EXTERNAL", "CUSTOM", "SET_POINT"]}},
        "rt": {"type": "array", "items": {"type": "string", "enum": ["ANALOG", "DOUBLE", "BINARY",
"PACKED", "INT64", "TEXT"]}},
        "priority": {"type": "array", "items": {"type": "integer", "minimum": 1, "maximum": 8}},
        "sg": {"type": "array", "items": {"type": "integer", "minimum": 0}},
        "tg": {"type": "array", "items": {"type": "integer", "minimum": 0}},
        "zd": {"type": "array", "items": {"type": "string"}},
        "iessRe": {"type": "string"},
        "messageRe": {"type": "string"},
        "pointId": {
          "type": "array",
          "items":  {
            "type": "object",
            "properties": {
              "sid": {"type": "integer", "minimum": 0},
              "iess": {"type": "string"}
            }
          }
        }
      }
    },
    "maxCount": {"type": "integer", "minimum": 1}
  },
  "required": ["filter"]
}
```

Events

Creates events of type CUSTOM_MESSAGE.

REQUEST

```
request_url = api_url + 'events'
query = [{
    'priority' : 1,
    'message' : 'msg1',
    'pointId' : {
        'sid' : 2,
        'iess' : 'iess1'
    },
    'floatValue' : 2.5,
    'intValue' : 3,
    'st' : 4,
    'ts' : 1699365500,
    'tss' : 0,
    'aux' : 6
}]
request = requests.post(request_url, headers=header, json=query)
```

RESPONSE

```
<Response [201]>
```

SCHEMA

```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "priority": {"type": "integer", "minimum": 1, "maximum": 8},
      "message": {"type": "string"},
      "pointId": {
        "type": "object",
        "properties": {
          "sid": {"type": "integer", "minimum": 0},
          "iess": {"type": "string"}
        }
      },
      "floatValue": {"type": "number"},
      "intValue": {"type": "integer"},
      "st": {"type": "integer", "minimum": 0},
      "ts": {"type": "integer", "minimum": 0},
      "tss": {"type": "integer", "minimum": 0},
      "aux": {"type": "integer", "minimum": 0}
    },
    "required": ["priority", "ts"]
  },
  "maxItems": 128
}
```

Reports

Configs query

Query report configurations.

REQUEST

```
>>> request_url = api_url + 'report/configs/query'
>>> query = {
...     'filters' : [{
...         'objectFilter' : {
...             'id' : [1, 2],
...             'fileRe' : '^abc',
...             'nameRe' : '^abc',
...             'sourceNameRe' : '^abc',
...             'sourceId' : [1],
...             'modified' : {
...                 'from' : 1603305950,
...                 'till' : 1603305959
          },
...             'sg' : [0, 1],
...             'tg' : [0, 1],
...             'md5' : 'abcdefg'
      },
...         'outputType' : 'FILE_RDF',
...         'executionCondition' : 'CYCLIC'
   }],
...     'page' : 2,
...     'pagesize' : 50
}
>>> request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

```
{
  "results": [{
    "id": 1,                          // id
    "reportDefinitionSource": "src1",   // source
    "reportDefinitionFile": "r1.edf",   // file
    "referenceTimeShift": "",          // reference time shift
    "runDelay": 60,                    // run delay [s]
    "timeMaskExpression": "0 * * * *",  // cron mask
    "eventsExpression": "",            // events expression
    "inputValues": "",                 // input values
    "outputMaskFileTxt": "",          // (optional) output mast file txt
    "outputMaskFileRdf": "",          // (optional) output mast file rdf
    "outputMaskFileEdf": "",          // (optional) output mast file edf
    "outputMaskFileHtml": "",         // (optional) output mast file html
    "outputMaskFilePdf": "",          // (optional) output mast file pdf
    "outputMaskFileCsv": "",          // (optional) output mast file csv
    "outputMaskDatabaseRdf": "",      // (optional) output mast database rdf
    "outputMaskDatabaseEdf": "",      // (optional) output mast database edf
    "outputMaskDatabaseHtml": ""      // (optional) output mast database html
  }],
  "matchCount": 1,       // matched count
  "totalCount": 100000,   // total count
}
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "filters": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "objectFilter": {
            "id": {"type": "array", "items": {"type": "integer", "minimum":
0}},
            "fileRe": {"type": "string"},
            "nameRe": {"type": "string"},
            "sourceNameRe": {"type": "string"},
            "sourceId": {"type": "array", "items": {"type": "integer",
"minimum": 0}},
            "modified": {
              "type": "object",
              "properties": {"from": {"type": "integer"}, "till": {"type":
"integer"}},
              "required": ["from", "till"]
            },
            "sg": {"type": "array", "items": {"type": "integer", "minimum":
0}},
            "tg": {"type": "array", "items": {"type": "integer", "minimum":
0}},
            "md5": {"type": "array", "items": {"type": "string"}}
          },
          "outputType": {
            "type": "string",
            "enum": ["FILE_TXT", "FILE_RDF", "FILE_EDF", "FILE_HTML",
"FILE_PDF", "FILE_CSV",
                     "DATABASE_RDF", "DATABASE_EDF", "DATABASE_HTML"]
          },
          "executionCondition": {"type": "string", "enum": ["CYCLIC",
"ON_EVENT"]}
        }
      }
    },
    "page": {"type": "integer", "minimum": 1},
    "pagesize": {"type": "integer", "minimum": 1}
    }
}
```

Configs

POST - create report configs

PUT - update report configs

DELETE - delete report configs

POST REQUEST

```
>>> request_url = api_url + 'report/configs'
>>> query = [{
...     'sourceId' : 123,
...     'rdfFileName' : 'myfile.rdf',
...     'config' :{
...         'referenceTimeShift': '',
...         'runDelay': 60,
...         'timeMaskExpression': '0 * * * *',
...         'eventsExpression': '',
...         'inputValues': '',
...         'outputMaskFileTxt': '',
...         'outputMaskFileRdf': '',
...         'outputMaskFileEdf': '',
...         'outputMaskFileHtml': '',
...         'outputMaskFilePdf': '',
...         'outputMaskFileCsv': '',
...         'outputMaskDatabaseRdf': '',
...         'outputMaskDatabaseEdf': '',
...         'outputMaskDatabaseHtml': ''
    }
}]
>>> request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

```
[{"id":new_report_config_id}]
```

SCHEMA

```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "sourceId": {"type": "integer", "minimum": 0},
      "rdfFileName": {"type": "string"},
      "config": {
        "type": "object",
        "properties": {
          "referenceTimeShift": {"type": "string"},
          "runDelay": {"type": "integer", "minimum": 0},
          "timeMaskExpression": {"type": "string"},
          "eventsExpression": {"type": "string"},
          "inputValues": {"type": "string"},
          "outputMaskFileTxt": {"type": "string"},
          "outputMaskFileRdf": {"type": "string"},
          "outputMaskFileEdf": {"type": "string"},
          "outputMaskFileHtml": {"type": "string"},
          "outputMaskFilePdf": {"type": "string"},
          "outputMaskFileCsv": {"type": "string"},
          "outputMaskDatabaseRdf": {"type": "string"},
          "outputMaskDatabaseEdf": {"type": "string"},
          "outputMaskDatabaseHtml": {"type": "string"}
        }
      }
    },
    "required": ["sourceId", "rdfFileName", "config"]
  }
}
```

PUT REQUEST

```
>>> request_url = api_url + 'report/configs'
>>> query = [{
...     'id' : 1,
...     'config' :{
...         'referenceTimeShift': '',
...         'runDelay': 60,
...         'timeMaskExpression': '0 * * * *',
...         'eventsExpression': '',
...         'inputValues': '1',
...         'outputMaskFileTxt': '',
...         'outputMaskFileRdf':'',
...         'outputMaskFileEdf': '',
...         'outputMaskFileHtml': '',
...         'outputMaskFilePdf': '',
...         'outputMaskFileCsv': '',
...         'outputMaskDatabaseRdf': '',
...         'outputMaskDatabaseEdf': '',
...         'outputMaskDatabaseHtml': ''
    }
}]
>>> request = requests.put(request_url, headers = header, json = query)
```

SCHEMA

```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "id": {"type": "integer", "minimum": 0},
      "config": {
        "type": "object",
        "properties": {
          "referenceTimeShift": {"type": "string"},
          "runDelay": {"type": "integer", "minimum": 0},
          "timeMaskExpression": {"type": "string"},
          "eventsExpression": {"type": "string"},
          "inputValues": {"type": "string"},
          "outputMaskFileTxt": {"type": "string"},
          "outputMaskFileRdf": {"type": "string"},
          "outputMaskFileEdf": {"type": "string"},
          "outputMaskFileHtml": {"type": "string"},
          "outputMaskFilePdf": {"type": "string"},
          "outputMaskFileCsv": {"type": "string"},
          "outputMaskDatabaseRdf": {"type": "string"},
          "outputMaskDatabaseEdf": {"type": "string"},
          "outputMaskDatabaseHtml": {"type": "string"}
        }
      }
    },
    "required": ["id", "config"]
  }
}
```

DELETE REQUEST

```
>>> request_url = api_url + 'report/configs'
>>> query = [{
...     'id' : 5,
... }]
>>> request = requests.delete(request_url, headers = header, json = query)
```

SCHEMA

```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "id": {"type": "integer", "minimum": 0}
    },
    "required": ["id"]
  }
}
```

Custom

GET - Retrieves custom report result

POST - Requests custom report

GET REQUEST

```
>>> id = 1234
>>> query = '?id={}'.format(id)
>>> request_url = api_url + 'report/custom' + query
>>> request = requests.get(request_url, headers=header)
```

RESPONSE

```
[
  ["abc", "aaa"],
  ["1.123", "2.3"]
]
```

POST REQUEST

```
>>> request_url = api_url + 'report/custom'
>>> query = {
...     'rdf' : {
...         'localTime' : True,
...         'showDstTransition' : True,
...         'showQuality' : True,
...         'precision' : 2,
...         'timeMode' : 'RELATIVE',
...         'addressingType' : 'A1',
...         'shadePriority' : "DEFAULT",
...         'rows' : [
...             [
...                 'abc'
...             ],
...             [{
...                 'content' : 'aaa',
...                 'showQuality' : True,
...                 'precision' : 3
...             }]
...         ]
...     },
...     'dtRef' : 1603305950,
...     'args' : {
...         'myarg' : {
...             'boolean' : True,
...             'number' : 5,
...             'packed' : 1,
...             'string' : 'x',
...             'timestamp' : 1603305950,
...             'point' : 'pt1',
...             'quality' : 'GOOD'
…       }
…    }
… }
```

RESPONSE

{"id":requested_id}

SCHEMA

```
{
  "type": "object",
  "properties": {
    "rdf": {
      "type": "object",
      "properties": {
        "localTime": {"type": "boolean"},
        "showDstTransition": {"type": "boolean"},
        "showQuality": {"type": "boolean"},
        "precision": {"type": "integer", "minimum": 0, "maximum": 18},
        "timeMode": {"type": "string", "enum": ["RELATIVE", "ABSOLUTE"]},
        "addressingType": {"type": "string", "enum": ["A1", "R1C1"]},
        "shadePriority": {"type": "string", "enum": ["DEFAULT",
"REGULAR_OVER_SHADE", "SHADE_OVER_REGULAR", "REGULAR_ONLY", "SHADE_ONLY"]},
        "rows": {
          "type": "array",
          "items": {
            "type": "array",
            "items": {
              "anyOf": [
                {
                  "type": "string"
                },
                {
                  "type": "object",
                  "properties": {
                    "content": {"type": "string"},
                    "showQuality": {"type": "boolean"},
                    "precision": {"type": "integer", "minimum": 0}
                  },
                  "required": ["content"]
                }
              ]
            }
          }
        }
      },
      "required": ["rows"]
    },
    "dtRef": {"type": "integer"},
    "args": {
      "type": "object",
      "additionalProperties": {
        "type": "object",
        "properties": {
          "boolean": {"type": "boolean"},
          "number": {"type": "number"},
          "packed": {"type": "integer"},
          "string": {"type": "string"},
          "timestamp": {"type": "integer"},
          "point": {"type": "string"},
          "quality": {"type": "string", "enum": ["GOOD", "FAIR", "POOR",
"BAD"]}
        }
      }
    }
  },
  "required": ["rdf", "dtRef"]
}
```

Global

Creates global report.
See also requests.

POST REQUEST

```
>>> request_url = api_url + 'report/global'
>>> query = {
...     'sourceId' : 123,
...     'file' : 'myfile.edf',
...     'name' : 'myobject',
...     'rdf' : {
...         'localTime' : True,
...         'showDstTransition' : True,
...         'showQuality' : True,
...         'precision' : 2,
...         'timeMode' : 'RELATIVE',
...         'addressingType' : 'A1',
...         'shadePriority' : 'DEFAULT',
...         'rows' : [
...             [
...                 'abc'
...             ],
...             [{
...                 'content': 'aaa',
...                 'showQuality' : True,
...                 'precision' : 3
...             }]
...         ]
...     },
...     'sg' : [0, 1],
...     'tg' : []
... }
>>> request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

```
{"id":requested_id}
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "sourceId": {"type": "integer", "minimum": 0},
    "file": {"type": "string"},
    "name": {"type": "string"},
    "rdf": {
      "type": "object",
      "properties": {
        "localTime": {"type": "boolean"},
        "showDstTransition": {"type": "boolean"},
        "showQuality": {"type": "boolean"},
        "precision": {"type": "integer", "minimum": 0, "maximum": 18},
        "timeMode": {"type": "string", "enum": ["RELATIVE", "ABSOLUTE"]},
        "addressingType": {"type": "string", "enum": ["A1", "R1C1"]},
        "shadePriority": {"type": "string", "enum": ["DEFAULT",
"REGULAR_OVER_SHADE", "SHADE_OVER_REGULAR", "REGULAR_ONLY", "SHADE_ONLY"]},
        "rows": {
          "type": "array",
          "items": {
            "type": "array",
            "items": {
              "anyOf": [
                {
                  "type": "string"
                },
                {
                  "type": "object",
                  "properties": {
                    "content": {"type": "string"},
                    "showQuality": {"type": "boolean"},
                    "precision": {"type": "integer", "minimum": 0}
                  },
                  "required": ["content"]
                }
              ]
            }
          }
        }
      },
      "required": ["rows"]
    },
    "sg": {"type": "array", "items": {"type": "integer", "minimum": 0}},
    "tg": {"type": "array", "items": {"type": "integer", "minimum": 0}}
  },
  "required": ["sourceId", "file", "rdf"]
}
```

Global run

Executes global report.

See also requests.

POST REQUEST

```
>>> request_url = api_url + 'report/global/run'
>>> query = {
...     'configId' : 123,
...     'dtRef' : 1603305950,
...     'args' : {
...         'myarg' : {
...             'boolean' : True,
...             'number' : 5,
...             'packed' : 1,
...             'string' : 'x',
...             'timestamp' : 1603305950,
...             'point' : 'pt1',
...             'quality' : 'GOOD'
...         }
...     }
... }
request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

```
{"id":requested_id}
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "configId": {"type": "integer", "minimum": 0},
    "dtRef": {"type": "integer"},
    "args": {
      "type": "object",
      "additionalProperties": {
        "type": "object",
        "properties": {
          "boolean": {"type": "boolean"},
          "number": {"type": "number"},
          "packed": {"type": "integer"},
          "string": {"type": "string"},
          "timestamp": {"type": "integer"},
          "point": {"type": "string"},
          "quality": {"type": "string", "enum": ["GOOD", "FAIR", "POOR",
"BAD", "NONE"]}
        }
      }
    }
  },
  "required": ["configId", "dtRef"]
}
```

Shades

Points

Query points with existing shade values.
The "order" parameter should be a list of fields names, for example: ["sid", "-iess"]. Adding "-" before a field name reverses the order.

POST REQUEST

```
>>> request_url = api_url + 'shades/points'
>>> query = {
...     'filters' : [{
...         'sid' : [1, 2],
...         'pointType' : 'ANALOG',
...         'iessRe' : '^abc'
...     }],
...     'order' : ['iess'],
...     'page' : 2,
...     'pagesize' : 50
... }
>>> request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

```
{
    "results": [
        {
            "sid": 1,
            "iess": "iess1",
            "type": "ANALOG"
        }
    ],
    "matchCount": 1
}
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "filters": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "sid": {"type": "array", "items": {"type": "integer", "minimum":
0}},
          "pointType": {"type": "string", "enum": ["ANALOG", "DOUBLE",
"BINARY", "PACKED", "INT64"]},
          "iessRe": {"type": "string"}
        }
      }
    },
    "order": {"type": "array", "items": {"type": "string"}},
    "page": {"type": "integer", "minimum": 1},
    "pagesize": {"type": "integer", "minimum": 1}
  }
}
```

Read

Read points shades for a specific time periods. See also requests.

GET REQUEST

```
>>> id = 1234
>>>> query = '?id={}'.format(id)
>>>> request_url = api_url + 'shades/read' + query
>>>> request = requests.get(request_url, headers=header)
```

RESPONSE

```
[{
  "items": [  // list of items results
    [         // list of samples
      [
        1603305950,  // start timestamp
        1603305960,  // end timestamp
        2.1,         // value
        "G",         // quality
      ],
      [1603305971, 1603305972, 2.2, "G"],
      [1603305981, 1603305982, 2.3, "G"]
    ],
    [
      [1603305950, 1603305951, 0.5, "G"]
    ]
  ],
  "status": "LAST"
}]
```

POST REQUEST

```
>>> request_url = api_url + 'shades/read'
>>> query = [{
...    'pointId' : {
...       'sid' : 1,
...       'iess' : 'iess1'
...    },
...    'period' : {
...       'from' : 1603305950,
...       'till' : 1603305959
...    }
... }]
>>> request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

```
{"id":requested_id}
```

SCHEMA

```
{
  "type": "array",
  "items":  {
    "type": "object",
    "properties": {
      "pointId": {
        "type": "object",
        "properties": {
          "sid": {"type": "integer", "minimum": 0},
          "iess": {"type": "string"}
        }
      },
      "period": {
        "type": "object",
        "properties": {"from": {"type": "integer"}, "till": {"type":
"integer"}},
        "required": ["from", "till"]
      }
    },
    "required": ["pointId", "period"]
  }
}
```

Write

Overwrite points shades for a specific time periods. See also requests.

POST REQUEST

```
>>> request_url = api_url + 'shades/write'
>>> query = [{
...    'pointId' : {
...        'sid' : 1,
...        'iess' : 'iess1'
...    },
...    'period' : {
...        'from' : 1603305950,
...        'till' : 1603305959
...    },
...    'value' : 1.1,
...    'quality' : 'GOOD'
... }]
>>> request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

```
{"id":requested_id}
```

SCHEMA

```
{
  "type": "array",
  "items":  {
    "type": "object",
    "properties": {
      "pointId": {
        "type": "object",
        "properties": {
          "sid": {"type": "integer", "minimum": 0},
          "iess": {"type": "string"}
        }
      },
      "period": {
        "type": "object",
        "properties": {"from": {"type": "integer"}, "till": {"type":
"integer"}},
        "required": ["from", "till"]
      },
      "value": {"type": ["number", "string", "boolean"]},
      "quality": {"type": "string", "enum": ["GOOD", "FAIR", "POOR",
"BAD"]}
    },
    "required": ["pointId", "period", "value", "quality"]
  }
}
```

Clear

Clear points shades for a specific time periods. See also requests.

POST REQUEST

```
>>> request_url = api_url + 'shades/clear'
>>> query = [{
...     'pointId' : {
...         'sid' : 1,
...         'iess' : 'iess1'
...     },
...     'period' : {
...         'from' : 1603305950,
...         'till' : 1603305959
...     }
... }]
>>> request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

```
{"id":requested_id}
```

SCHEMA

```
{
  "type": "array",
  "items":  {
    "type": "object",
    "properties": {
      "pointId": {
        "type": "object",
        "properties": {
          "sid": {"type": "integer", "minimum": 0},
          "iess": {"type": "string"}
        }
      },
      "period": {
        "type": "object",
        "properties": {"from": {"type": "integer"}, "till": {"type":
"integer"}},
        "required": ["from", "till"]
      }
    },
    "required": ["pointId", "period"]
  }
}
```

Copy

Copy shades between points for a specific time periods. The points must be of the same type. See also requests.

POST REQUEST

```
>>> request_url = api_url + 'shades/copy'
>>> query = [{
...    'srcPointId' : {
...        'sid' : 1,
...        'iess' : 'iess1'
...    },
...    'dstPointId' : {
...        'sid' : 2,
...        'iess' : 'iess2'
...    },
...    'period' : {
...        'from': 1603305950,
...        'till': 1603305959
...    }
... }]
>>> request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

```
{"id":requested_id}
```

SCHEMA

```
{
  "type": "array",
  "items":  {
    "type": "object",
    "properties": {
      "srcPointId": {
        "type": "object",
        "properties": {
          "sid": {"type": "integer", "minimum": 0},
          "iess": {"type": "string"}
        }
      },
      "dstPointId": {
        "type": "object",
        "properties": {
          "sid": {"type": "integer", "minimum": 0},
          "iess": {"type": "string"}
        }
      },
      "period": {
        "type": "object",
        "properties": {"from": {"type": "integer"}, "till": {"type":
"integer"}},
        "required": ["from", "till"]
      }
    },
    "required": ["srcPointId", "dstPointId", "period"]
  }
}
```

Users

Sg

Returns a list of all security groups (sg).

GET REQUEST

```
>>> request_url = api_url + 'sg'
>>> request = requests.get(request_url, headers=header)
```

RESPONSE

```
[{
  "id": 0,
  "name": "admin",
  "desc": ""
}]
```

Tg
Returns a list of all technological groups (tg).

GET REQUEST

```
>>> request_url = api_url + 'tg'
>>> request = requests.get(request_url, headers=header)
```

RESPONSE

```
[{
  "id": 0,
  "name": "admin",
  "desc": ""
}]
```

User sg
Returns users security group (sg).

GET REQUEST

```
>>> request_url = api_url + 'user/sg'
>>> request = requests.get(request_url, headers=header)
```

RESPONSE

```
[{
  "id": 0,
  "name": "admin",
  "desc": ""
}]
```

User profile
Returns user effective profile. Effective profile is composition of all profiles with user security groups access.

GET REQUEST

```
>>> request_url = api_url + 'user/profile'
>>> request = requests.get(request_url, headers=header)
```

RESPONSE

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE boost_serialization>
<boost_serialization signature="serialization::archive" version="19">
<Root class_id="0" tracking_level="1" version="0" object_id="_0">
	<mValue></mValue>
	<mEditable>1</mEditable>
	<mCanHaveExtraSubNodes>1</mCanHaveExtraSubNodes>
	<mSubNodes class_id="1" tracking_level="0" version="0">
		<count>0</count>
		<item_version>0</item_version>
	</mSubNodes>
</Root>
```

User query

Query users matching selected criteria. Requires admin SG.
The "order" parameter should be a list of fields names, for example: ["created", "-id"]. Adding "-" before a field name reverses the order.

POST REQUEST

```
>>> request_url = api_url + 'users/query'
>>> query = {
...     'filters' : [{
...         'id' : [1, 2],
...         'nameRe' : '^abc'
...     }],
...     'order' : ['created'],
...     'page' : 2,
...     'pagesize' : 50
... }
>>> request = requests.post(request_url, headers = header, json = query)
```

RESPONSE

```
{
  "results": [{
    "id": 1,
    "name": "user1",
    "description": "abc",
    "locked ": false,
    "created": 1603305950,
    "modified": 1603305959,
    "sg": [0, 1]
  }],
  "matchCount": 1,
}
```

SCHEMA

```
{
  "type": "object",
  "properties": {
    "filters": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "id": {"type": "array", "items": {"type": "integer", "minimum":
0}},
          "nameRe": {"type": "string"}
        }
      }
    },
    "order": {"type": "array", "items": {"type": "string"}},
    "page": {"type": "integer", "minimum": 1},
    "pagesize": {"type": "integer", "minimum": 1},
    "fields": {
      "type": "array",
      "items": {
        "type": "string",
        "enum": ["id", "name", "description", "locked", "created",
"modified", "sg"]
      }
    }
  }
}
```

Status

Status

Returns information about current server status.

GET REQUEST

```
>>> request_url = api_url + 'status'
>>> request = requests.get(request_url, headers=header)
```

RESPONSE

```
{
   "time": 1701175825,
   "timezone_offset": 3600,
   "srv_connection": "LOGGED_IN | SYNCHRONIZED | STATIC_CHANGED | DYNAMIC_CHANGED |
UPDATE_CYCLE",
   "object_srv_connection": "CONNECTED",
   "archive_srv_connection": "CONNECTED",
   "report_srv_connection": "CONNECTED",
   "objects_count": 6316,
   "objects_pending_count": 0,
   "http_connections": 2,
   "https_connections": 0,
   "soap_http_connections": 0,
   "soap_https_connections": 0,
   "live_data_connections": 0,
   "session_count": 2,
   "request_count": 23,
   "request_running_count": 0
}
```

License

Returns EDS server license information.

GET REQUEST

```
>>> request_url = api_url + 'license'
>>> request = requests.get(request_url, headers=header)
```

RESPONSE

```
{
  "AllowedNetworkAddresses": "0.0.0.0/0",
  "ClientType:DBA": "0.0.0.0/0",
  "ClientType:TERM": "0.0.0.0/0",
  "Clustering": "Yes",
  "DatabaseName": "eds",
  "ExpiryDate": "unlimited",
  "ImportAlarms": "Yes",
  "LicenseType": "Commercial",
  "LocalAlarms": "Yes",
  "MSExcelPluginEnabled": "Yes",
  "MaxArchivedPoints": "200000",
  "MaxClients": "128",
  "MaxControlDiagrams": "50000",
  "MaxDiagramSources": "10",
  "MaxPoints": "200000",
  "MaxProcessDiagrams": "50000",
  "MaxReports": "5000",
  "MaxWEBApiSessions": "unlimited",
  "MaxZDs": "20",
  "MaxZIPs": "50000",
  "MobileLicenseExpiryDate": "19/07/2022",
  "MobileLicenseMultiServer": "No",
  "MultiServerConnectivity": "Yes",
  "NotificationServiceEnabled": "No",
  "OneWayScanners": "Yes",
  "OnlineCalculationsEnabled": "Yes",
  "Product": "EnterpriseServer",
  "ReportsEnabled": "Yes",
  "TabularTrendsEnabled": "Yes",
  "Vendor": "Transition Technologies S.A.",
  "WEBApiSupportsGraphics": "Yes",
  "WinNTOwner": "xxx",
  "WinNTSerialNbr": "xxxxx-xxxxx-xxxxx-xxxxx"
}
```