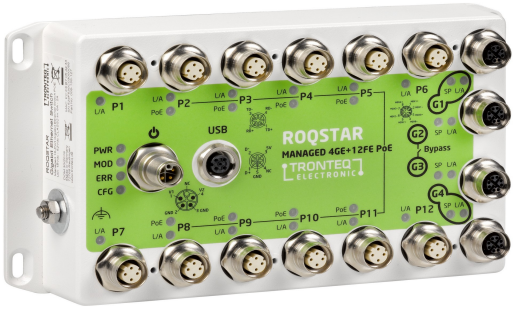
TRONTEQ
FOR CONNECTED PUBLIC TRANSPORT

Application Note



Using the OpenAPI

of ROQSTAR Managed Gigabit Ethernet Switches

Summary

ROQSTAR devices support an API (Application Programming Interface) for applications where automated deployment or monitoring is needed.

Using this API the device's configuration can be managed and status data can be queried.

Compliance with OpenAPI specification 3 gives access to numerous tools for code generation and documentation.

Supported Devices

This document describes the OpenAPI that is supported by TRONTEQ's ROQSTAR Managed Gigabit Ethernet Switches. Their part numbers are:

- 006-130-117
- 006-130-118
- 006-130-124
- 006-130-125
- 006-130-126
- 006-130-127

The commands described in this document match API version 1.0 and are part of the device's software beginning with version 2.4.0.

Content

1 Protocol and Format

1.1 General

The API uses `http` or `https` to send commands and receive responses. It can be used with many programming languages and libraries.

The exact URL paths for each command as well as additional parameters, values, examples etc. are specified in a file named ROQSTAR-GE_OpenAPI_v1.0.yaml. There are tools available to display the information in this file.

Some requests and responses of the API contain parameters that are in `JSON` format, which is human-readable. There are many tools and libraries available that can be used to construct or parse `JSON` objects.

Whether a request was successful is shown via the `http` status codes.

1.2 Authentication

In general, access to the API is granted only after successful authentication. Like in the web interface the user has to login using valid credentials (user name and password). A successful login results in a http(s) session id, which is then used in subsequent commands.

Only one session per user name can be active at the same time (including web interface sessions); a second login will invalidate the first session.

There is the option to ‘logout’, which will invalidate the given active session.

Both login and logout will generate entries in the device’s internal event log.

Aaprt from the ‘login’ request there is an additional request available without a valid session: The command to query the API version.

1.3 Configuration file encoding

Configuration files can be obtained from a ROQSTAR device – downloaded either via API or web interface. Typically the file name corresponds to the given configuration name as seen in the web interface, the file extension is `.cfg`.

When opened in a text editor, a configuration file looks like this:


```
{
    "roqs-container": {
        "content-list": [
            {
                "content-checksum": 1880215503,
                "content-compatibility": [],
                "content-data":
"H4sIAAAAAAACA+0YTa+jNvCvPHHOPtmAweSw0qq9rLTdHir18lRFxh6yVgGztk
ne01P+e20TkkCSTZU2p1ZWAsy3Z+yZsd8j8Y130fI9Mn3ZgjXR8uU9EmC4lp2Vq
o2WUbSIpIiWaBHVwAyYXz795Kj+GD9yqtyGzDDcEq2NinNf6oUCvnJ2udZveHux
RBlwkxBTppuBO5xBy6RZKeB1aveH47XvCSu1DaBn3dwfj22J257KftL8e2EMOr/
MuwWoWHD4+T8MVYItZs3racpxPCDbL465rt6UrJNDeEMLtV8xpzH068JTjOvDBR
82UJ8WIiYa2fpqxIzZKh3uVjKc0iwu8kywLCUI8hwJhKpM5ETQApUJIEKJSDjPS
YxSHiOCcopSJojjIGmVEcAVJHGJRMw5TXiRC1JiUlSoxEXGeB4nnAGLWZ66DZKh
CldpRQUrCIlTH1fmS2kEEGdOhkhZlUOeFoxhRgESjhGDooiTKsO0oCjPY0jzxG0
6XDAi0iJFgEQZ54xjSirKEGDhnk4HLTmleSWyvMRpkicxT9wMOKlYwQSFrCJ5RR
CKKaNVdd6ObsLueY9KLcV6vARSesv0MWzr7f6mrjuLsDtjvL6tmD7uFNNcuOja7
C+Stuz0wmhdq9I3rfNu+HRlJLvdX8/PEXBIFAAA",
                "content-encoding": 4,
                "content-id": "006-130-118",
                "content-name": "test.cfg",
                "content-scope": 1,
                "content-type": 1,
                "content-version": 1
            }
        ],
        "data-checksum": 1387631223,
        "meta-version": 1
    }
}
```

Before a configuration file is transmitted to a ROQSTAR device, it must be encoded in base64. The result is a single line of text that looks like this:

```
ewogICAgInJvcXMtY29udGFpbmVyIjogewogICAgICAgICJjb250ZW50LWxpc3Q
iOiBbCiAgICAgICAgICAgIHsKICAgICAgICAgICAgICAgICJjb250ZW50LWNoZW
Nrc3VtIjogMzM4MjI3NTYwLAogICAgICAgICAgICAgICAgImNvbnRlbnQtY29tc
GF0aWJpbGl0eSI6IFtdLAogICAgICAgICAgICAgICAgImNvbnRlbnQtZGF0YSI6
ICJINHNJQUFBQUFBQUNBKzBZVGErak52Q3ZQSEhPUHRtQXdlU3cwcXE5ckxUZEh
pcjE4bFJGeGg2eVZnR3p0a25lMDFQK2UyMFRra0NTVFpVMnAxWldBc3kzWit5Wn
NkOGo4WTEzMGZJOU1uM1pnalhSOHVVOUVtQzRscDJWcW8yV1ViU0lwSWlXYUJIV
ndBeVlYejc5NUtqK0dEOS9wZkh3MmJJR0hQa1h4Vm45OU9YVDE2ZmZna2pIN3Y0
Ylp2NTB5SmlRNS9HSEJzeFdhWS9CUmZ5TU0vcU1BMXgxb05tZ3YySzFnVVhVTWM
wYTR5MFZyVUdPQVQySDRZZ2RBRThCcW1GeWIvcWFXZGl5dHdrK0dHNmxONWRtS1
VJL3NyQlRxbDVCS3lZQ0F0QllwdTBKZUxlSURPZ05hRzhkanZNQXh0RUluWnBvM
3pydnJObzdLOW81N3prS0UrWWJPemsrS0x5VzBGby9ZZGxhMEJYak1FU25rclg3
UGpobUg1c0xIdE5xSTgwUk10R0NuWmFEWEsra05OMVI1RlFoNjYxcVlSMHRyZTR
kVmtpMmJwV3hrcHNEaCtpN0dsNUhpaFBCK0xBdXZDTTZBR2N0UnQ3bGZUZVE3eG
...
```

This is used as the value of the “content” argument in the related openAPI commands.

2 Basic examples

The examples in this document assume that the ROQSTAR device is using the following default settings:

- IP address: 192.168.1.1
- user name: admin
- password: password

To send the commands to the ROQSTAR device, the Linux command line tool `curl` is used. In a Windows environment it is also available using the `cygwin` runtime environment.

The session id stored in a file called cookies.jar.

Using `curl`, the `http` status codes can be output with the additional parameter `-w '\n%{http_code}\n'`

2.1 Login

Request:

```
curl -X 'POST' \
'http://192.168.1.1/oapi/v1/login' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{"user":"admin", "password":"password"}' \
-c cookies.jar
```

These commands write the session data to the file `cookies.jar`, then login using the name `admin` and the password `password`. The session data will be used for the following accesses.

2.2 Get system ID information

Request:

```
curl -X 'GET' \
'http://192.168.1.1/oapi/v1/system/id' \
-H 'accept: application/json' \
-b cookies.jar
```

Response:

```
{
  "sn": "524F520304000000",
  "mgmt mac": "FC:F8:B7:FF:FF:A0",
  "sw version": "2.3.1",
  "part number": "006-130-117",
  "description": "ROQSTAR Managed 2GE+8FE Gigabit Ethernet
Switch M12 IP54",
  "configname": "bus_2.cfg"
}
```

2.3 Get link status

Request:

```
curl -X 'GET' \
'http://192.168.1.1/oapi/v1/status/link/status' \
-H 'accept: application/json' \
-b cookies.jar
```

Response:

```
{
  "G1": true,
  "G2": true,
  "P1": true,
  "P2": false,
  "P3": true,
  "P4": true,
  "P5": false,
  ...
}
```

3 Configuration management

Configuration files can be applied in different ways.

If the ‘running configuration’ is changed that means that the currently active settings are affected – the configuration is applied or reset immediately. However, the change in settings is not saved persistently, so it will be lost when the device’s power is disconnected or the device is rebooted.

A configuration can also be saved persistently: Changes to the ‘startup configuration’ will be applied immediately and will also be loaded at the next startup.

There is also the option to reset the device to its factory default settings. This is done by using the HTTP ‘delete’ command either for the ‘running configuration’ (i.e. not saved) or the ‘startup configuration’ (i.e. saved persistently).

3.1 Download the current configuration

In this example the configuration file is named `config.cfg` and is located in the current directory. It will be applied immediately and saved persistently.

The configuration file is encoded in base64 (see section 1.3). In Linux bash, it can be decoded using the command `base64 -d`

Request:

```
curl -X 'GET' \
'http://192.168.1.1/oapi/v1/configuration/running' \
-H 'accept: application/json' \
-b cookies.jar
```

Response (shortened):

```
{
  "name": "myconfig.cfg",
  "content":
"ewogICAgInJvcXMtY29udGFpbmVyIjogewogICAgICAgICJjb250ZW50LWxpc3
QiOiBbCiAgICAgICAgICAgIHsKICAgICAgICAgICAgICAgICJjb250ZW50LWNoZ
WNrc3VtIjogODY5NTk1ODQxLAogICAgICAgICAgICAgICAgImNvbnRlbnQtY29t
```

```
cGF0aWJpbGl0eSI6IFtdLAogICAgICAgICAgICAgICAgImNvbnRlbnQtZGF0YSI
6ICJINHNJQUFBQUFBQUNBKzFaM1cvanVCSC9Wd0k5NXdKK2s4cERnVVZiRkFmc1
hoK3V1RDRzRGdaRmpuekM2V3NsMmRrZ3lQL2VJV1hMTmhOZm5QVDgxSzZ3SzR1Y
ytjMXdocHdaemo1bC9qZlhaL2RQMmJncFdwakc3UDdyVStaaGRFUFZUMVhYWnZk
WmRwdFZQcnVudDFrTmRvVHh5NmUvSXRXdis4OS9HaGFaWExsdWJRTUhCb0l2aE0
0b3VXTjM1STRTSE8rN1lZb3pBelRkQklFc2tFK1BQVEtTNTl0WFlHZ0NReGNZK2
g0WWxzQ3dCWWE5QjRZbk1IeUI0ZStCRVFtTVdHREVlMkJrQWlNWEdQa2VHSlhBc
UFWR3ZRZEdKekI2Z2RIdmdURUpqRmxne"

}
```

3.2 Upload and apply a configuration file directly

In this example the configuration file is named `config.cfg` and is located in the current directory. It will be applied immediately and saved persistently.

Prior to transmitting the configuration file it must be encoded in base64 (see section 1.3). This can be achieved using the command `base64 -w 0`

Encoding the data followed by sending the request:

```
encoded=`base64 -w 0 config.cfg`
curl -X 'PUT' \
'http://192.168.1.1/oapi/v1/configuration/startup' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{"name": "config.cfg", "content": "'$encoded'"}' \
-b cookies.jar
```

The device will respond immediately to the request, but the actual application of the configuration file will take some time. In order to know when it is finished, the status can be polled with the following command:

```
curl -X 'GET' \
'http://192.168.1.1/oapi/v1/openapi/status/configuration' \
-H 'accept: application/json' \
-b cookies.jar
```

To ensure the status is referring to the original request and not any other, the ‘id’ values received with both commands can be compared.

4 Software Update

In this example the software image is named `update.bin` and is located in the current directory.

Upload and install the file:

```
curl -X 'PUT' \
'http://192.168.1.1/oapi/v1/system/software' \
-H 'accept: */*' \
-H 'Content-Type: application/octet-stream' \
-b cookies.jar \
--data-binary '@update.bin'
```

This will take some time, typically up to 30 seconds. The response is sent afterwards, so care should be taken to have an appropriate (long enough) timeout to receive it.

The newly installed software will boot during the next startup. The command to reboot is:

```
curl -X 'POST' \
'http://192.168.1.1/oapi/v1/system/reboot' \
-H 'accept: */*' \
-d '' \
-b cookies.jar
```