API Wiki

Welcome to NextGen EmcoreView API Wiki page. In this page, a detailed tutorial with some popular language examples will be provided. Following is a quick start guide with user authentication.

User's Guide

This part of the documentation, which is mostly prose, begins with some background information about NextGen EmcoreView API, then focuses on step-by-step instructions for system engineer development with integration of using EmcoreView API calls.

Foreword

Read this before you get started with NextGen EmcoreView API. This hopefully answers some questions about the purpose and goals of the project, and when you should or should not be using it.

- ▼ API

  An application programming interface (API), is a computing interface that defines interactions between multiple software intermediaries. It defines the kinds of calls or requests that can be made, how to make them, the data formats that should be used, the conventions to follow, etc.

- ▼ REST

  NextGen EmcoreView API complies to REST architectural style which is stands for "REprenstational State Transfer". Like any other architectural style, REST also does have its own 6 guiding constraints which must be satisfied if an interface needs to be referred as RESTful. These principles are listed below.

  - ▼ Client–server

    By separating the user interface concerns from the data storage concerns, we improve the portability of the user interface across multiple platforms and improve scalability by simplifying the server components.

  - ▼ Stateless

    Each request from client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server. Session state is therefore kept entirely on the client.

  - ▼ Cacheable

    Cache constraints require that the data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests.

  - ▼ Uniform interface

    By applying the software engineering principle of generality to the component interface, the overall system architecture is simplified and the visibility of interactions is improved. In order to obtain a uniform interface, multiple architectural constraints are needed to guide the behavior of components. REST is defined by four interface constraints: identification of resources; manipulation of resources through representations; self-descriptive messages; and, hypermedia as the engine of application state.

  - ▼ Layered system

    The layered system style allows an architecture to be composed of hierarchical layers by constraining component behavior such that each component cannot “see” beyond the immediate layer with which they are interacting.

- ▼ Data Format

  Data will be transfer based on the https requests and data format will be in JSON.

▼ API Keys and Token

API keys will be needed in order to obtain a authentication token for communication. User permission is compiles to web application user database. API keys will be changed if user changes their password. Each API token is valid for 60 minuets when it is generated. API keys and Tokens can be managed by this page top section "API Keys & Token" using a Admin account.

▼ IP Address and API-keys

In this User's Guide, https://192.168.2.1/api/v1 will be used as a IP address example.

```
{
    "username": "monitor",
    "api-key": "ab709becab16d470f924530b4267647e5139d0d2"
}
```

And

```
{
    "username": "admin",
    "api-key": "fd971a34a00a3258c5e9d19f947696e043109b7d"
}
```

will be used as a user authentication api-keys.

Check Alive

The first API request is to check whether the system is online or to check whether you have the connection to the NextGen NMS card. The instruction about how to setup the IP address of a NextGen NMS card is inside the User Manual. At this point, a hostname of IP address should be available to you.

▼ C

```c
CURL *hnd = curl_easy_init();

curl_easy_setopt(hnd, CURLOPT_CUSTOMREQUEST, "GET");
curl_easy_setopt(hnd, CURLOPT_URL, "https://192.168.2.1/api/v1");

struct curl_slist *headers = NULL;
headers = curl_slist_append(headers, "user-agent: Mozilla/5.0");
curl_easy_setopt(hnd, CURLOPT_HTTPHEADER, headers);

CURLcode ret = curl_easy_perform(hnd);
```

▼ C#

```csharp
var client = new RestClient("https://192.168.2.1/api/v1");
var request = new RestRequest(Method.GET);
request.AddHeader("user-agent", "Mozilla/5.0");
IRestResponse response = client.Execute(request);
```

▼ Java

```java
OkHttpClient client = new OkHttpClient();

Request request = new Request.Builder()
  .url("https://192.168.2.1/api/v1")
  .get()
  .addHeader("user-agent", "Mozilla/5.0")
  .build();

Response response = client.newCall(request).execute();
```

▼ JavaScript

```
const settings = {
  "async": true,
  "crossDomain": true,
  "url": "https://192.168.2.1/api/v1",
  "method": "GET",
  "headers": {
    "user-agent": "Mozilla/5.0"
  }
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

▼ Node.js-HTTP

```
const http = require("https");

const options = {
  "method": "GET",
  "hostname": "192.168.2.1",
  "port": null,
  "path": "/api/v1",
  "headers": {
    "user-agent": "Mozilla/5.0"
  }
};

const req = http.request(options, function (res) {
  const chunks = [];

  res.on("data", function (chunk) {
    chunks.push(chunk);
  });

  res.on("end", function () {
    const body = Buffer.concat(chunks);
    console.log(body.toString());
  });
});

req.end();
```

▼ Python

```
import requests

url = "https://192.168.2.1/api/v1"

headers = {'user-agent': 'Mozilla/5.0'}

response = requests.request("GET", url, headers=headers)

print(response.text)
```

No matter what kind of programming language has been used, the request should be shown as below:

```
GET https://192.168.2.1/api/v1 HTTP/1.1
User-Agent: Mozilla/5.0
accept-encoding: gzip, deflate
```

And the response will be shown as followed:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 86
```

```
Server: Werkzeug/0.11.4 Python/2.7.9
Date: Sat, 01 Apr 2017 10:30:48 GMT

{
  "Message": "Welcome to Emcore NextGen NMS API Ver 1.",
  "SW Ver.": "Ver 1.1.7"
}
```

Inside the body, there will be a JSON data indicating the SW Ver of the NMS card with a greeting message. If you can receive this message, that means you have successfully setup the NMS card.

Request Token

In order to communicate with NextGen NMS securely, a API token is used for authentication. To get this token, open your web browser and navigate to API tab. API keys will be provided for you. Also, API tokens can be managed here. Please note that all the users will be compiled to the same permission as web users. For more information, please refer to the user manual. Once API key is obtained, you can request a API token based on that.

▼ C

```c
CURL *hnd = curl_easy_init();

curl_easy_setopt(hnd, CURLOPT_CUSTOMREQUEST, "POST");
curl_easy_setopt(hnd, CURLOPT_URL, "https://192.168.2.1/api/v1/token");

struct curl_slist *headers = NULL;
headers = curl_slist_append(headers, "user-agent: Mozilla/5.0");
headers = curl_slist_append(headers, "content-type: application/json");
curl_easy_setopt(hnd, CURLOPT_HTTPHEADER, headers);

curl_easy_setopt(hnd, CURLOPT_POSTFIELDS, "{\"username\": \"admin\",\"api-key\": \"fd971a34a00a3258c5e9d19f947696e043109b7d\"}");

CURLcode ret = curl_easy_perform(hnd);
```

▼ C#

```csharp
var client = new RestClient("https://192.168.2.1/api/v1/token");
var request = new RestRequest(Method.POST);
request.AddHeader("user-agent", "Mozilla/5.0");
request.AddHeader("content-type", "application/json");
request.AddParameter("application/json", "{\"username\": \"admin\",\"api-key\": \"fd971a34a00a3258c5e9d19f947696e043109b7d\"}", Paramet
IRestResponse response = client.Execute(request);
```

▼ Java

```java
OkHttpClient client = new OkHttpClient();

MediaType mediaType = MediaType.parse("application/json");
RequestBody body = RequestBody.create(mediaType, "{\"username\": \"admin\",\"api-key\": \"fd971a34a00a3258c5e9d19f947696e043109b7d\"}")
Request request = new Request.Builder()
  .url("https://192.168.2.1/api/v1/token")
  .post(body)
  .addHeader("user-agent", "Mozilla/5.0")
  .addHeader("content-type", "application/json")
  .build();

Response response = client.newCall(request).execute();
```

▼ JavaScript

```javascript
const settings = {
  "async": true,
  "crossDomain": true,
  "url": "https://192.168.2.1/api/v1/token",
  "method": "POST",
  "headers": {
    "user-agent": "Mozilla/5.0",
    "content-type": "application/json"
  },
  "processData": false,
  "data": "{\"username\": \"admin\",\"api-key\": \"fd971a34a00a3258c5e9d19f947696e043109b7d\"}"
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

▼ Node.js-HTTP

```
const http = require("https");

const options = {
  "method": "POST",
  "hostname": "192.168.2.1",
  "port": null,
  "path": "/api/v1/token",
  "headers": {
    "user-agent": "Mozilla/5.0",
    "content-type": "application/json"
  }
};

const req = http.request(options, function (res) {
  const chunks = [];

  res.on("data", function (chunk) {
    chunks.push(chunk);
  });

  res.on("end", function () {
    const body = Buffer.concat(chunks);
    console.log(body.toString());
  });
});

req.write(JSON.stringify({username: 'admin', 'api-key': 'fd971a34a00a3258c5e9d19f947696e043109b7d'}));
req.end();
```

▼ Python

```
import requests

url = "https://192.168.2.1/api/v1/token"

payload = "{\"username\": \"admin\",\"api-key\": \"fd971a34a00a3258c5e9d19f947696e043109b7d\"}"
headers = {
    'user-agent': "Mozilla/5.0",
    'content-type': "application/json"
    }

response = requests.request("POST", url, data=payload, headers=headers)

print(response.text)
```

No matter what kind of programming language has been used, the request should be shown as below:

```
POST https://192.168.2.1/api/v1/token HTTP/1.1
User-Agent: Mozilla/5.0
content-type: application/json
accept-encoding: gzip, deflate
content-length: 89

{
  "username": "admin",
  "api-key": "fd971a34a00a3258c5e9d19f947696e043109b7d"
}
```

And the response will be shown as followed:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 162
Server: Werkzeug/0.11.4 Python/2.7.9
Date: Fri, 21 Apr 2017 05:22:16 GMT

{
  "expiration": 3600,
  "token": "eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwLyXPpkeatRc2OA-_2xeMFgoM"
}
```

Please note that each token is valid for 1 hour. After it is expired, a new token need to be requested again. When a new token is generated, a expiration will be given in the response. Otherwise, only the same token will be given instead.

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 31
Server: Werkzeug/0.11.4 Python/2.7.9
Date: Fri, 21 Apr 2017 06:28:06 GMT

{
  "error": "Login expired."
}
```

You can separate the read and write request with different user token for higher security level. For example:

A Monitor permission level token can be used for read only purposes.

A User permission level token can be used for write only purposes. (And you can request this token when needed)

Once a token is obtained, please keep it available for future communication purpose. In this example, the following token has been used and it has Administrator level permission:

```
{
  "token": "eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwLyXPpkeatRc2OA-_2xeMFgoM"
}
```

There are two places this token can be put to. Inside the request header or part of the URI.

Request All Cards Info

All card info API is for understanding what kind of cards and active slot number of each card in this chassis.

▼ C

```
CURL *hnd = curl_easy_init();

curl_easy_setopt(hnd, CURLOPT_CUSTOMREQUEST, "GET");
curl_easy_setopt(hnd, CURLOPT_URL, "https://192.168.2.1/api/v1/cards");

struct curl_slist *headers = NULL;
headers = curl_slist_append(headers, "user-agent: Mozilla/5.0");
headers = curl_slist_append(headers, "authorization: eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zSt
curl_easy_setopt(hnd, CURLOPT_HTTPHEADER, headers);

CURLcode ret = curl_easy_perform(hnd);
```

▼ C#

```
var client = new RestClient("https://192.168.2.1/api/v1/cards");
var request = new RestRequest(Method.GET);
request.AddHeader("user-agent", "Mozilla/5.0");
request.AddHeader("authorization", "eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwL
IRestResponse response = client.Execute(request);
```

▼ Java

```
OkHttpClient client = new OkHttpClient();

Request request = new Request.Builder()
  .url("https://192.168.2.1/api/v1/cards")
  .get()
  .addHeader("user-agent", "Mozilla/5.0")
  .addHeader("authorization", "eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwLyXPpk
  .build();

Response response = client.newCall(request).execute();
```

▼ JavaScript

```
const settings = {
  "async": true,
  "crossDomain": true,
  "url": "https://192.168.2.1/api/v1/cards",
  "method": "GET",
  "headers": {
    "user-agent": "Mozilla/5.0",
    "authorization": "eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwLyXPpkeatRc2OA-
  }
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

▼ Node.js-HTTP

```
const http = require("https");

const options = {
  "method": "GET",
  "hostname": "192.168.2.1",
  "port": null,
  "path": "/api/v1/cards",
  "headers": {
    "user-agent": "Mozilla/5.0",
    "authorization": "eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwLyXPpkeatRc2OA-
  }
};

const req = http.request(options, function (res) {
  const chunks = [];

  res.on("data", function (chunk) {
    chunks.push(chunk);
  });

  res.on("end", function () {
    const body = Buffer.concat(chunks);
    console.log(body.toString());
  });
});

req.end();
```

▼ Python

```
import requests

url = "https://192.168.2.1/api/v1/cards"

headers = {
    'user-agent': "Mozilla/5.0",
    'authorization': "eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwLyXPpkeatRc2OA-
    }

response = requests.request("GET", url, headers=headers)

print(response.text)
```

No matter what kind of programming language has been used, the request should be shown as below:

```
GET https://192.168.2.1/api/v1/cards HTTP/1.1
User-Agent: Mozilla/5.0
Authorization: eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwLyXPpkeatRc2OA-_2xeMFgoM
accept-encoding: gzip, deflate
```

In this example, An OSU card, a NextGen L-Band Rx card, Tx card and a NextGen NMS card are sitting in slot number 1, 8, 11 and 16.

So the response will be shown as followed:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 8609
Server: Werkzeug/0.11.4 Python/2.7.9
Date: Fri, 21 Apr 2017 06:01:06 GMT

{
  "1": {
    "FW ver": " A05",
    "HW ver": "B1  ",
    "alerts": null,
    "errors": null,
    "label": null,
    "model num": "OTS-1-OSU-2-15-00               ",
    "name": "OSU",
    "parameters": {
      "Manufacturer Date       ": {
        "active": true,
        "description": null,
        "is_writable": false,
        "name": "Manufacturer Date       ",
        "value": "8042020 "
      },
      "OSU Switch CH 1         ": {
        "active": true,
        "description": null,
        "is_writable": true,
        "name": "OSU Switch CH 1         ",
        "off_val": "Normal",
        "on_val": "Cross",
        "value": false
      },
      "OSU Switch CH 2         ": {
        "active": true,
        "description": null,
        "is_writable": true,
        "name": "OSU Switch CH 2         ",
        "off_val": "Normal",
        "on_val": "Cross",
        "value": false
      }
    },
    "production date": null,
    "serial num": "VE80328 ",
    "slot_num": 1,
    "status": [
      "1. Card Status              - OK"
    ],
    "temperature": null
  },
  "8": {
    "FW ver": "Ver 1.9",
    "HW ver": "Rev X2",
    "alerts": null,
    "errors": [
      "2. RF Power - Major Low",
      "3. OPT Power - Major Low"
    ],
    "label": null,
    "model num": "OTP-XXXXX-XXXX",
    "name": "LNG-Rx",
    "parameters": {
      "Attenuation": {
        "active": true,
        "description": null,
        "high": null,
        "is_fix_range": false,
        "is_writable": true,
        "max_val": 31.5,
        "min_val": 0.0,
        "name": "Attenuation",
        "precision": 1.0,
        "step": 0.5,
        "step_array": null,
        "unit": "dB",
        "value": 10.0
      },
      "Mfg. Date": {
        "active": true,
        "description": null,
        "is_writable": false,
        "name": "Mfg. Date",
        "value": "2020-01-09"
      },
      "Optical PWR": {
        "active": true,
        "description": null,
        "is_writable": false,
        "max_val": "30.0",
        "min_val": "-99.99",
        "name": "Optical PWR",
        "step": 0.01,
        "unit": "dBm",
        "value": "-99.99"
      },
      "RF Output PWR": {
        "active": true,
        "description": null,
        "is_writable": false,
        "max_val": "35.0",
        "min_val": "-99.99",
        "name": "RF Output PWR",
        "step": 0.01,
        "unit": "dBm",
        "value": -19.86
      },
      "Vin": {
        "active": true,
        "description": null,
        "is_writable": false,
        "max_val": "15.0",
        "min_val": "0.0",
        "name": "Vin",
        "step": 0.001,
        "unit": "V",
        "value": "11.492"
      }
    },
    "production date": null,
    "serial num": "1234567890-4321",
    "slot_num": 8,
    "status": [
      "1. Power Supply - Normal",
      "4. Temperature - Normal"
    ],
    "temperature": 28.5
  },
  "11": {
    "FW ver": "Ver 1.9",
    "HW ver": "Rev X2",
    "alerts": [
      "3. OPT Power - Low"
    ],
    "errors": [
      "2. RF Power - Major Low"
    ],
      "label": "abc",
      "model num": "OTP-XXXXX-XXXX",
      "name": "LNG-Tx",
      "parameters": {
        "AGC Set Point": {
          "active": true,
          "description": null,
          "high": null,
          "is_fix_range": false,
          "is_writable": true,
          "max_val": 3.0,
          "min_val": -3.0,
          "name": "AGC Set Point",
          "precision": 1.0,
          "step": 0.5,
          "step_array": null,
          "unit": "dBm",
          "value": 0.0
        },
        "AGC/MGC": {
          "active": true,
          "description": null,
          "is_writable": true,
          "name": "AGC/MGC",
          "off_val": "AGC",
          "on_val": "MGC",
          "value": true
        },
        "Attenuation": {
          "active": true,
          "description": null,
          "high": null,
          "is_fix_range": false,
          "is_writable": true,
          "max_val": 31.5,
          "min_val": 0.0,
          "name": "Attenuation",
          "precision": 1.0,
          "step": 0.5,
          "step_array": null,
          "unit": "dB",
          "value": 0.0
        },
        "LNB Control": {
          "active": true,
          "description": null,
          "high": null,
          "is_fix_range": false,
          "is_writable": true,
          "max_val": 4.0,
          "min_val": 0.0,
          "name": "LNB Control",
          "precision": 0,
          "step": 1.0,
          "step_array": [
            "OFF",
            "13V",
            "13V w/22k",
            "18V",
            "18V w/22k"
          ],
          "unit": "Selections",
          "value": 0.0
        },
        "LNB Mon": {
          "active": true,
          "description": null,
          "is_writable": false,
          "max_val": "20.0",
          "min_val": "0.0",
          "name": "LNB Mon",
          "step": 0.001,
          "unit": "V",
          "value": 0.11
        },
        "Laser BIAS": {
          "active": true,
          "description": null,
          "is_writable": false,
          "max_val": "166.71",
          "min_val": "0.0",
        "name": "Laser BIAS",
        "step": 0.01,
        "unit": "mA",
        "value": 37.82
      },
      "Laser On/Off": {
        "active": true,
        "description": null,
        "is_writable": true,
        "name": "Laser On/Off",
        "off_val": "OFF",
        "on_val": "ON",
        "value": true
      },
      "Mfg. Date": {
        "active": true,
        "description": null,
        "is_writable": false,
        "name": "Mfg. Date",
        "value": "2020-01-09"
      },
      "Optical PWR": {
        "active": true,
        "description": null,
        "is_writable": false,
        "max_val": "30.0",
        "min_val": "-99.99",
        "name": "Optical PWR",
        "step": 0.01,
        "unit": "dBm",
        "value": 5.34
      },
      "Peak Func.": {
        "active": true,
        "buttons": [
          "PEAK"
        ],
        "is_writable": true,
        "name": "Peak Func."
      },
      "RF Input PWR": {
        "active": true,
        "description": null,
        "is_writable": false,
        "max_val": "35.0",
        "min_val": "-99.99",
        "name": "RF Input PWR",
        "step": 0.01,
        "unit": "dBm",
        "value": -17.56
      },
      "Vin": {
        "active": true,
        "description": null,
        "is_writable": false,
        "max_val": "15.0",
        "min_val": "0.0",
        "name": "Vin",
        "step": 0.001,
        "unit": "V",
        "value": 11.458
      }
    },
    "production date": null,
    "serial num": "1234567890-4321",
    "slot_num": 11,
    "status": [
      "1. Power Supply - Normal",
      "4. Temperature - Normal",
      "5. Laser Status - Normal",
      "6. AGC Func. - Normal"
    ],
    "temperature": 29.12
  },
  "16": {
    "FW ver": "Ver 11A18     ",
    "HW ver": "Rev. D",
    "alerts": null,
    "errors": null,
    "label": null,
    "model num": "NextGen Control Card",
    "name": "NextGen NMS",
    "parameters": {
      "1.SFP": {
        "active": true,
        "description": null,
        "hardware_version": null,
        "name": "SFP",
        "rx_bandwidth": null,
        "status": false,
        "tx_bandwidth": null
      },
      "Fan 1": {
        "active": true,
        "description": null,
        "is_writable": true,
        "name": "Fan 1",
        "off_val": null,
        "on_val": null,
        "value": false
      },
      "Fan 2": {
        "active": true,
        "description": null,
        "is_writable": true,
        "name": "Fan 2",
        "off_val": null,
        "on_val": null,
        "value": false
      },
      "Remote Update Enable": {
        "active": true,
        "description": null,
        "is_writable": false,
        "name": "Remote Update Enable",
        "value": "True (1-On 2-Off 3-Off)"
      },
      "SlotID": {
        "active": true,
        "description": null,
        "is_writable": false,
        "name": "SlotID",
        "value": "16"
      },
      "Software Version": {
        "active": true,
        "description": null,
        "is_writable": false,
        "name": "Software Version",
        "value": "Ver 1.1.7"
      }
    },
    "production date": "2017-03-25",
    "serial num": "123456-354    ",
    "slot_num": 16,
    "status": [
      "1. Buzzer: Off",
      "2. Global Alarm: Normal",
      "3. Relay: Disable"
    ],
    "temperature": 30.0
  }
}
```

Request Individual Card Info

Individual card API is used for query all the information of a single card. In this example, a OSU in slot number 1 is shown as below:

▼ C

```
CURL *hnd = curl_easy_init();

curl_easy_setopt(hnd, CURLOPT_CUSTOMREQUEST, "GET");
curl_easy_setopt(hnd, CURLOPT_URL, "https://192.168.2.1/api/v1/cards/1?token=eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNz
struct curl_slist *headers = NULL;
headers = curl_slist_append(headers, "user-agent: Mozilla/5.0");
curl_easy_setopt(hnd, CURLOPT_HTTPHEADER, headers);

CURLcode ret = curl_easy_perform(hnd);
```

▼ C#

```
var client = new RestClient("https://192.168.2.1/api/v1/cards/1?token=eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2f
var request = new RestRequest(Method.GET);
request.AddHeader("user-agent", "Mozilla/5.0");
IRestResponse response = client.Execute(request);
```

▼ Java

```
OkHttpClient client = new OkHttpClient();

Request request = new Request.Builder()
  .url("https://192.168.2.1/api/v1/cards/1?token=eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge7
  .get()
  .addHeader("user-agent", "Mozilla/5.0")
  .build();

Response response = client.newCall(request).execute();
```

▼ JavaScript

```
const settings = {
  "async": true,
  "crossDomain": true,
  "url": "https://192.168.2.1/api/v1/cards/1?token=eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBg
  "method": "GET",
  "headers": {
    "user-agent": "Mozilla/5.0"
  }
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

▼ Node.js-HTTP

```
const http = require("https");

const options = {
  "method": "GET",
  "hostname": "192.168.2.1",
  "port": null,
  "path": "/api/v1/cards/1?token=eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwLyXP
  "headers": {
    "user-agent": "Mozilla/5.0"
  }
};

const req = http.request(options, function (res) {
  const chunks = [];

  res.on("data", function (chunk) {
    chunks.push(chunk);
  });

  res.on("end", function () {
    const body = Buffer.concat(chunks);
    console.log(body.toString());
  });
});

req.end();
```

▼ Python

```python
import requests

url = "https://192.168.2.1/api/v1/cards/1"

querystring = {"token":"eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwLyXPpkeatRc2O

headers = {'user-agent': 'Mozilla/5.0'}

response = requests.request("GET", url, headers=headers, params=querystring)

print(response.text)
```

No matter what kind of programming language has been used, the request should be shown as below:

```
GET https://192.168.2.1/api/v1/cards/1?token=eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNl
User-Agent: Mozilla/5.0
accept-encoding: gzip, deflate
```

And the response will be shown as followed:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 1018
Server: Werkzeug/0.11.4 Python/2.7.9
Date: Fri, 21 Apr 2017 06:08:51 GMT

{
  "FW ver": " A05",
  "HW ver": "B1  ",
  "alerts": null,
  "errors": null,
  "label": null,
  "model num": "OTS-1-OSU-2-15-00               ",
  "name": "OSU",
  "parameters": {
    "Manufacturer Date       ": {
      "active": true,
      "description": null,
      "is_writable": false,
      "name": "Manufacturer Date       ",
      "value": "8042020 "
    },
    "OSU Switch CH 1         ": {
      "active": true,
      "description": null,
      "is_writable": true,
      "name": "OSU Switch CH 1         ",
      "off_val": "Normal",
      "on_val": "Cross",
      "value": false
    },
    "OSU Switch CH 2         ": {
      "active": true,
      "description": null,
      "is_writable": true,
      "name": "OSU Switch CH 2         ",
      "off_val": "Normal",
      "on_val": "Cross",
      "value": false
    }
  },
  "production date": null,
  "serial num": "VE80328 ",
  "slot_num": 1,
  "status": [
    "1. Card Status              - OK"
  ],
  "temperature": null
}
```

Request Chassis Info

A chassis info API is for obtaining the information of the chassis.

▼ C

```
CURL *hnd = curl_easy_init();

curl_easy_setopt(hnd, CURLOPT_CUSTOMREQUEST, "GET");
curl_easy_setopt(hnd, CURLOPT_URL, "https://192.168.2.1/api/v1/chassis");

struct curl_slist *headers = NULL;
headers = curl_slist_append(headers, "user-agent: Mozilla/5.0");
headers = curl_slist_append(headers, "authorization: eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zSt
curl_easy_setopt(hnd, CURLOPT_HTTPHEADER, headers);

CURLcode ret = curl_easy_perform(hnd);
```

▼ C#

```
var client = new RestClient("https://192.168.2.1/api/v1/chassis");
var request = new RestRequest(Method.GET);
request.AddHeader("user-agent", "Mozilla/5.0");
request.AddHeader("authorization", "eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwL
IRestResponse response = client.Execute(request);
```

▼ Java

```
OkHttpClient client = new OkHttpClient();

Request request = new Request.Builder()
  .url("https://192.168.2.1/api/v1/chassis")
  .get()
  .addHeader("user-agent", "Mozilla/5.0")
  .addHeader("authorization", "eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwLyXPpk
  .build();

Response response = client.newCall(request).execute();
```

▼ JavaScript

```
const settings = {
  "async": true,
  "crossDomain": true,
  "url": "https://192.168.2.1/api/v1/chassis",
  "method": "GET",
  "headers": {
    "user-agent": "Mozilla/5.0",
    "authorization": "eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwLyXPpkeatRc2OA-
  }
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

▼ Node.js-HTTP

```
const http = require("https");

const options = {
  "method": "GET",
  "hostname": "192.168.2.1",
  "port": null,
  "path": "/api/v1/chassis",
  "headers": {
    "user-agent": "Mozilla/5.0",
    "authorization": "eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwLyXPpkeatRc2OA-
  }
};

const req = http.request(options, function (res) {
  const chunks = [];

  res.on("data", function (chunk) {
    chunks.push(chunk);
  });

  res.on("end", function () {
    const body = Buffer.concat(chunks);
    console.log(body.toString());
  });
});

req.end();
```

▼ Python

```
import requests

url = "https://192.168.2.1/api/v1/chassis"

headers = {
    'user-agent': "Mozilla/5.0",
    'authorization': "eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwLyXPpkeatRc2OA-
    }

response = requests.request("GET", url, headers=headers)

print(response.text)
```

No matter what kind of programming language has been used, the request should be shown as below:

```
GET https://192.168.2.1/api/v1/chassis HTTP/1.1
User-Agent: Mozilla/5.0
Authorization: eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwLyXPpkeatRc2OA-_2xeMFgoM
accept-encoding: gzip, deflate
```

And the response will be shown as followed:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 1865
Server: Werkzeug/0.11.4 Python/2.7.9
Date: Fri, 21 Apr 2017 06:11:55 GMT

{
  "FW ver": "Ver 10A4        ",
  "HW ver": "REV. A",
  "alerts": null,
  "errors": null,
  "label": null,
  "model num": "NextGen NMS Optiva2",
  "name": "Optiva 2",
  "parameters": {
    "Backplane CardDetect    ": {
      "active": true,
      "description": null,
      "is_writable": false,
      "name": "Backplane CardDetect    ",
      "value": "0x8481"
    },
    "Backplane DipSwitch     ": {
      "active": true,
      "description": null,
      "is_writable": false,
      "name": "Backplane DipSwitch     ",
      "value": "0b1111"
    },
    "Backplane PS 1 Current  ": {
      "active": true,
      "description": null,
      "is_writable": false,
      "max_val": "10000.0",
      "min_val": "0.0",
      "name": "Backplane PS 1 Current  ",
      "step": 1.0,
      "unit": "mA",
      "value": 723.0
    },
    "Backplane PS 1 Voltage  ": {
      "active": true,
      "description": null,
      "is_writable": false,
      "max_val": "20.0",
      "min_val": "0.0",
      "name": "Backplane PS 1 Voltage  ",
      "step": 0.01,
      "unit": "Volts",
      "value": "12.04"
    },
    "Backplane PS 2 Current  ": {
      "active": true,
      "description": null,
      "is_writable": false,
      "max_val": "10000.0",
      "min_val": "0.0",
      "name": "Backplane PS 2 Current  ",
      "step": 1.0,
      "unit": "mA",
      "value": 325.0
    },
    "Backplane PS 2 Voltage  ": {
      "active": true,
      "description": null,
      "is_writable": false,
      "max_val": "20.0",
      "min_val": "0.0",
      "name": "Backplane PS 2 Voltage  ",
      "step": 0.01,
      "unit": "Volts",
      "value": "12.01"
    }
  },
  "production date": null,
  "serial num": "DEMO 2",
  "slot_num": 17,
  "status": [
    "1. Backplane Status         - Normal"
  ],
  "temperature": 25.0
}
```

Update Parameter

In order to update or write to a parameter of a single card, it is suggested to read it first. For example, to update the attenuation of NextGen L-band card sitting in slot number 11, here is the information when reading the individual card:

```
... ...
"Attenuation": {
      "active": true,
      "description": null,
      "high": null,
      "is_fix_range": false,
      "is_writable": true,
      "max_val": 31.5,
      "min_val": 0.0,
      "name": "Attenuation",
      "precision": 1.0,
      "step": 0.5,
      "step_array": null,
      "unit": "dB",
      "value": 10.0
    },
... ...
```

It will show that the "attenuation" parameter is writeable, with range from 0 to 31.5dB. It can be changed with minimum step of 0.5dB. In this example, we would like to change it to 0dB.

▼ C

```
CURL *hnd = curl_easy_init();

curl_easy_setopt(hnd, CURLOPT_CUSTOMREQUEST, "POST");
curl_easy_setopt(hnd, CURLOPT_URL, "https://192.168.2.1/api/v1/cards/11");

struct curl_slist *headers = NULL;
headers = curl_slist_append(headers, "user-agent: Mozilla/5.0");
headers = curl_slist_append(headers, "authorization: eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zSt
headers = curl_slist_append(headers, "content-type: application/json");
curl_easy_setopt(hnd, CURLOPT_HTTPHEADER, headers);

curl_easy_setopt(hnd, CURLOPT_POSTFIELDS, "{\"name\": \"Attenuation\",\"value\": 0}");

CURLcode ret = curl_easy_perform(hnd);
```

▼ C#

```
var client = new RestClient("https://192.168.2.1/api/v1/cards/11");
var request = new RestRequest(Method.POST);
request.AddHeader("user-agent", "Mozilla/5.0");
request.AddHeader("authorization", "eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwL
request.AddHeader("content-type", "application/json");
request.AddParameter("application/json", "{\"name\": \"Attenuation\",\"value\": 0}", ParameterType.RequestBody);
IRestResponse response = client.Execute(request);
```

▼ Java

```
OkHttpClient client = new OkHttpClient();

MediaType mediaType = MediaType.parse("application/json");
RequestBody body = RequestBody.create(mediaType, "{\"name\": \"Attenuation\",\"value\": 0}");
Request request = new Request.Builder()
  .url("https://192.168.2.1/api/v1/cards/11")
  .post(body)
  .addHeader("user-agent", "Mozilla/5.0")
  .addHeader("authorization", "eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwLyXPpk
  .addHeader("content-type", "application/json")
  .build();

Response response = client.newCall(request).execute();
```

▼ JavaScript

```
const settings = {
  "async": true,
  "crossDomain": true,
  "url": "https://192.168.2.1/api/v1/cards/11",
  "method": "POST",
  "headers": {
    "user-agent": "Mozilla/5.0",
    "authorization": "eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwLyXPpkeatRc2OA-
    "content-type": "application/json"
  },
  "processData": false,
  "data": "{\"name\": \"Attenuation\",\"value\": 0}"
};
```

```
$.ajax(settings).done(function (response) {
  console.log(response);
});
```

▼ Node.js-HTTP

```
const http = require("https");

const options = {
  "method": "POST",
  "hostname": "192.168.2.1",
  "port": null,
  "path": "/api/v1/cards/11",
  "headers": {
    "user-agent": "Mozilla/5.0",
    "authorization": "eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwLyXPpkeatRc2OA-
    "content-type": "application/json"
  }
};

const req = http.request(options, function (res) {
  const chunks = [];

  res.on("data", function (chunk) {
    chunks.push(chunk);
  });

  res.on("end", function () {
    const body = Buffer.concat(chunks);
    console.log(body.toString());
  });
});

req.write(JSON.stringify({name: 'Attenuation', value: 0}));
req.end();
```

▼ Python

```
import requests

url = "https://192.168.2.1/api/v1/cards/11"

payload = "{\"name\": \"Attenuation\",\"value\": 0}"
headers = {
    'user-agent': "Mozilla/5.0",
    'authorization': "eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwLyXPpkeatRc2OA-
    'content-type': "application/json"
    }

response = requests.request("POST", url, data=payload, headers=headers)

print(response.text)
```

No matter what kind of programming language has been used, the request should be shown as below:

```
POST https://192.168.2.1/api/v1/cards/11 HTTP/1.1
User-Agent: Mozilla/5.0
Authorization: eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1NTczNiwiaWF0IjoxNDkyNzUyMTM2fQ.eyJpZCI6MX0.zStBge75tGz8HNluTOwLyXPpkeatRc2OA-_2xeMFgoM
content-type: application/json
accept-encoding: gzip, deflate
content-length: 48

{
  "name": "Attenuation",
  "value": 0
}
```

And the response will be shown as followed:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 67
Server: Werkzeug/0.11.4 Python/2.7.9
Date: Fri, 21 Apr 2017 06:29:59 GMT

{
  "name": "Attenuation",
  "status": "Succeed.",
  "value": 0
}
```

Another example worth mentioning is the LNB control. Since it is a selection, we can obtain its information as below:

```
... ...
"LNB Control": {
      "active": true,
      "description": null,
      "high": null,
      "is_fix_range": false,
      "is_writable": true,
      "max_val": 4.0,
      "min_val": 0.0,
      "name": "LNB Control",
      "precision": 0,
      "step": 1.0,
      "step_array": [
        "OFF",
        "13V",
        "13V w/22k",
        "18V",
        "18V w/22k"
      ],
      "unit": "Selections",
      "value": 0.0
    },
... ...
```

A step array is provided for its reference, a correlation can be established as followed:

0 - OFF

1 - 13V

2 - 13V w/22k

3 - 18V

4 - 18V w/22k

So for example, to change its value to 18V, the following command can be used:

▼ C

```
CURL *hnd = curl_easy_init();

curl_easy_setopt(hnd, CURLOPT_CUSTOMREQUEST, "POST");
curl_easy_setopt(hnd, CURLOPT_URL, "https://192.168.2.1/api/v1/cards/11?token=eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1OTc5NiwiaWF0IjoxNDkyN

struct curl_slist *headers = NULL;
headers = curl_slist_append(headers, "user-agent: Mozilla/5.0");
headers = curl_slist_append(headers, "content-type: application/json");
curl_easy_setopt(hnd, CURLOPT_HTTPHEADER, headers);

curl_easy_setopt(hnd, CURLOPT_POSTFIELDS, "{\"name\": \"LNB Control\",\"value\": 3}");

CURLcode ret = curl_easy_perform(hnd);
```

▼ C#

```
var client = new RestClient("https://192.168.2.1/api/v1/cards/11?token=eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1OTc5NiwiaWF0IjoxNDkyNzU2MTk2
var request = new RestRequest(Method.POST);
request.AddHeader("user-agent", "Mozilla/5.0");
request.AddHeader("content-type", "application/json");
request.AddParameter("application/json", "{\"name\": \"LNB Control\",\"value\": 3}", ParameterType.RequestBody);
IRestResponse response = client.Execute(request);
```

▼ Java

```
OkHttpClient client = new OkHttpClient();

MediaType mediaType = MediaType.parse("application/json");
RequestBody body = RequestBody.create(mediaType, "{\"name\": \"LNB Control\",\"value\": 3}");
Request request = new Request.Builder()
  .url("https://192.168.2.1/api/v1/cards/11?token=eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1OTc5NiwiaWF0IjoxNDkyNzU2MTk2fQ.eyJpZCI6MX0.3ltJ1B
  .post(body)
  .addHeader("user-agent", "Mozilla/5.0")
  .addHeader("content-type", "application/json")
  .build();

Response response = client.newCall(request).execute();
```

▼ JavaScript

```
const settings = {
  "async": true,
  "crossDomain": true,
  "url": "https://192.168.2.1/api/v1/cards/11?token=eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1OTc5NiwiaWF0IjoxNDkyNzU2MTk2fQ.eyJpZCI6MX0.3ltJ
  "method": "POST",
  "headers": {
    "user-agent": "Mozilla/5.0",
    "content-type": "application/json"
  },
  "processData": false,
  "data": "{\"name\": \"LNB Control\",\"value\": 3}"
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

▼ Node.js-HTTP

```
const http = require("https");

const options = {
  "method": "POST",
  "hostname": "192.168.2.1",
  "port": null,
  "path": "/api/v1/cards/11?token=eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1OTc5NiwiaWF0IjoxNDkyNzU2MTk2fQ.eyJpZCI6MX0.3ltJ1BXaprx1sJypOUwpJZ
  "headers": {
    "user-agent": "Mozilla/5.0",
    "content-type": "application/json"
  }
};

const req = http.request(options, function (res) {
  const chunks = [];

  res.on("data", function (chunk) {
    chunks.push(chunk);
  });

  res.on("end", function () {
    const body = Buffer.concat(chunks);
    console.log(body.toString());
  });
});

req.write(JSON.stringify({name: 'LNB Control', value: 3}));
req.end();
```

▼ Python

```
import requests

url = "https://192.168.2.1/api/v1/cards/11"

querystring = {"token":"eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1OTc5NiwiaWF0IjoxNDkyNzU2MTk2fQ.eyJpZCI6MX0.3ltJ1BXaprx1sJypOUwpJZg_oW2Vw6cQ

payload = "{\"name\": \"LNB Control\",\"value\": 3}"
headers = {
    'user-agent': "Mozilla/5.0",
    'content-type': "application/json"
    }

response = requests.request("POST", url, data=payload, headers=headers, params=querystring)

print(response.text)
```

No matter what kind of programming language has been used, the request should be shown as below:

```
POST https://192.168.2.1/api/v1/cards/11?token=eyJhbGciOiJIUzI1NiIsImV4cCI6MTQ5Mjc1OTc5NiwiaWF0IjoxNDkyNzU2MTk2fQ.eyJpZCI6MX0.3ltJ1BXaprx1s
User-Agent: Mozilla/5.0
content-type: application/json
accept-encoding: gzip, deflate
content-length: 48

{
  "name": "LNB Control",
  "value": 3
}
```

And the response will be shown as followed:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 67
Server: Werkzeug/0.11.4 Python/2.7.9
Date: Fri, 21 Apr 2017 06:42:19 GMT

{
  "name": "LNB Control",
  "status": "Succeed.",
  "value": 3
}
```

EOF