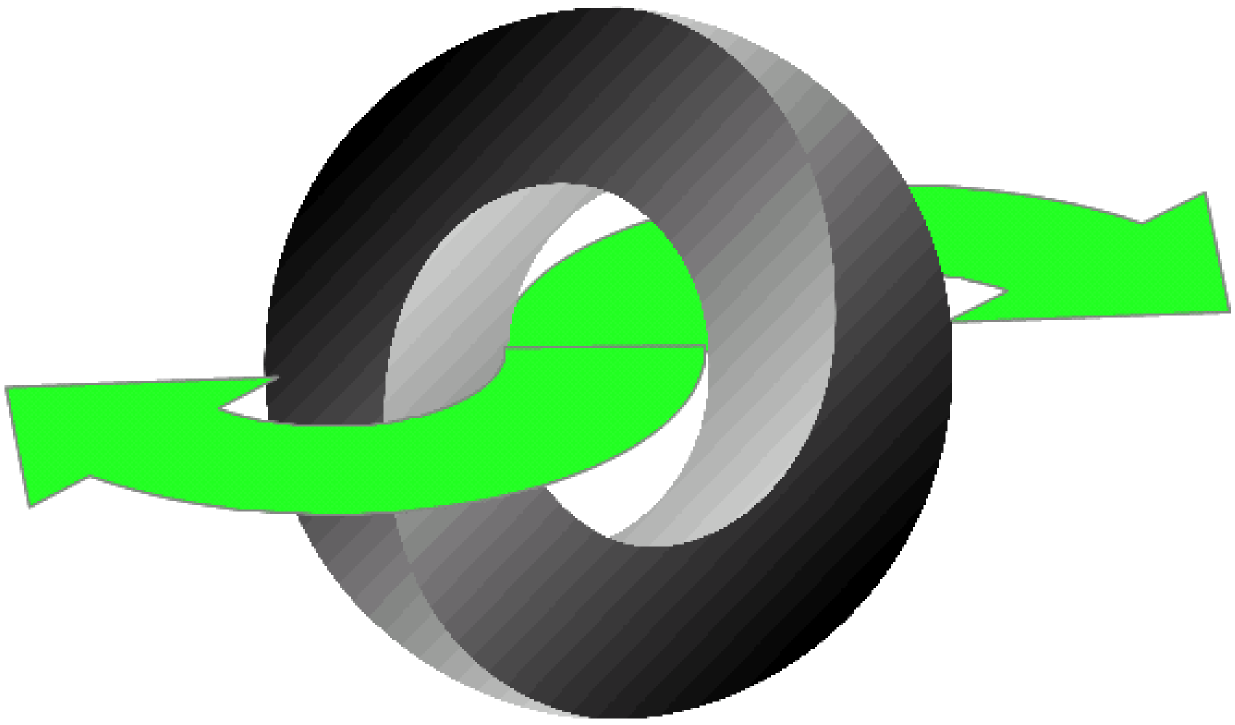
ProMoS DMS JSON Data Exchange



Version 1.4
23.05.2016

ProMoS DMS JSON Data Exchange

Printed: Mai 2016 in Belp, Switzerland

Publisher

MST Systemtechnik AG

Managing Editor

...

Technical Editors

Martin Frei

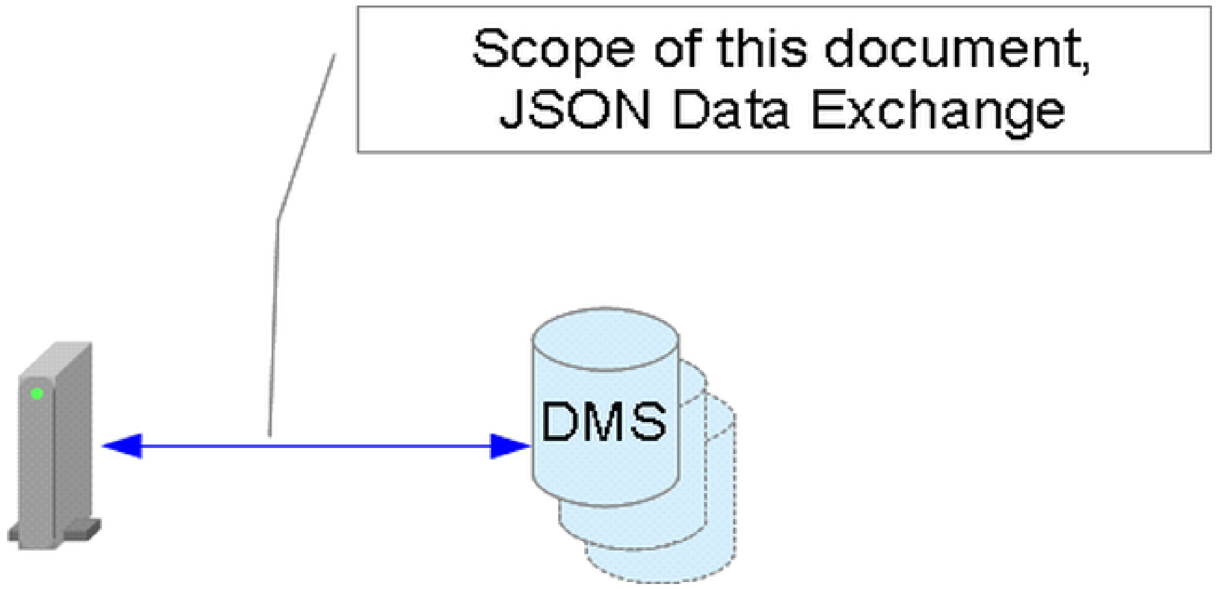
Team Coordinator

Christoph Müller

Table of contents

1 Introduction

This document describes the data exchange between an external device and the ProMoS Data Management System (DMS).



1.1 History

Version	Who	Date	Remark
1.0	mst_frem	19.03.2015	First Draft
1.1	mst_frem	23.12.2015	Additional fields and descriptions on WebSocket monitoring
1.2	mst_frem	22.01.2016	Additional description of security mechanism (SSL/TLS, authentication)
1.3	mst_frem	19.02.2016	Changed command "monitor" to "subscribe". New command "unsubscribe". Detailed description on query and tag for subscriptions.
1.4	mst_frem	13.04.2016	New commands and fields for Changelog hasProtData changed to hasChangelog New extInfos "state"

2 General

2.1 Configuration

The communication has to be enabled in the DMS configuration dialogue (communication properties). Default is enabled for http/ws and https/wss.
The port can be configured there also (default: 9020 for http/ws and 9021 for https/wss).

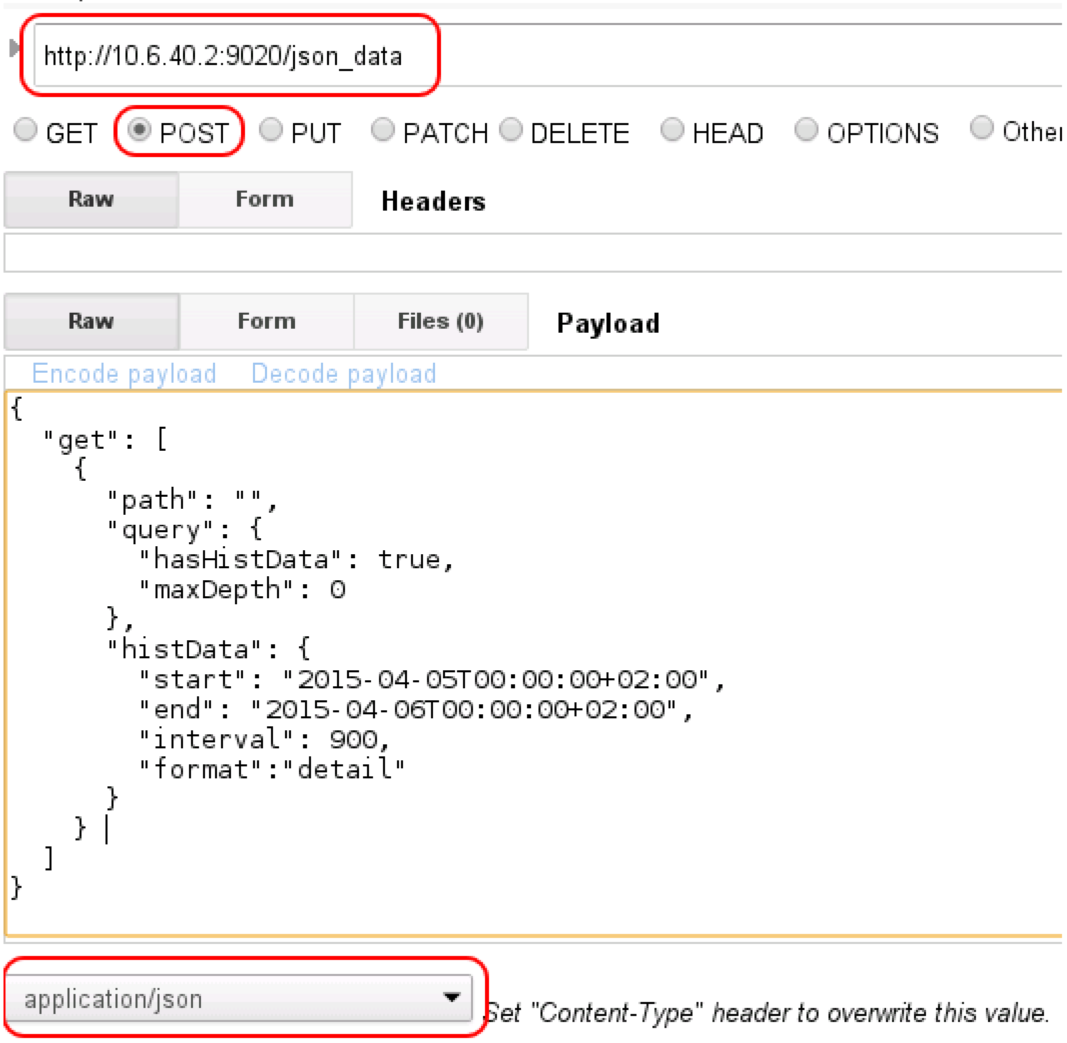
2.2 Tools

There are several tools to test the communication:

- Apache JMeter
  A powerful Java application. Primary for performance measure, but can also be used for any other test cases.
  There is a WebSocket extension, binaries can be downloaded here and dependencies here.

- Java application to test HTTP/RESTful webservices.
  With the binaries here.

- Advanced REST client for Google Chrome
  Easy to use browser plugin.
  Example:



```
{
  "get": [
    {
      "path": "",
      "query": {
        "hasHistData": true,
        "maxDepth": 0
      },
      "histData": {
        "start": "2015-04-05T00:00:00+02:00",
        "end": "2015-04-06T00:00:00+02:00",
        "interval": 900,
        "format":"detail"
      }
    }
  ]
}
```

- Simple WebSocket Client for Google Chrome

2.3 Used Technologies

All data transfers are based on JSON structures, see http://en.wikipedia.org/wiki/JSON and http://json.org/

All JSON data have to be UTF-8 encoded.
It is recommended to use real json data types, because for boolean fields - a string "FALSE" will be true!

For the JSON data transport, there are 2 options (both without or with SSL/TLS):

- HTTP(s) POST based, see http://en.wikipedia.org/wiki/POST_(HTTP)
  The JSON data for requests is transferred in the POST body (instead of the standard URL encoded request) to have the same encoding for request and response.
  Content type "application/json" is used.

- WebSocket(s) based, see http://en.wikipedia.org/wiki/WebSocket and RFC6455:
  Corresponding to RFC6455, Version 13.

  Fulfilled implementations:
  - PING/PONG and CLOSE control frames.
  - Multiple frames (fragmentation).
  - Masking.

  Specialties:
  - Client must not send masked frames (optional).
  - Larger responses are sent fragmented in frames with max. 8'192 chars of payload data.

  Limitations:
  - Receiving maximum is 4MB (4'194'304 chars) payload data per frame.
  - Receiving maximum is 4MB (4'194'304 chars) per message (summ of fragmented frames).

2.4 Security

2.4.1 Insecure connection

By default, only local clients (from 127.0.0.1) can connect to non secure connections (http/ws).
Due to maximum performance on internal networks, there is no authentication for non secure connections.

Other options see in the following chapters.

2.4.2 Secure connection (SSL/TLS)

By default, all non local clients (not from 127.0.0.1) have to connect over a secure connection (SSL/TLS, https/wss).
The client has to authenticate himself by a certificate or over "Basic access authentication".

When there is no valid client certificate, the server will request a Basic access authentication.

Other options see in the following chapters.

2.4.2.1 Certificates

The server will send a self signed certificate with his domain name, a client certificate is optional requested.

Any received certificate will be verified (issuer is our DMS, valid dates).
The CN (Common Name) of the client certificate will be used to verify the user in the client users list.

On a ProMoS V1/V2 installation this list is located at "`{INSTALL_DIR}\proj\{PROJECT}\cfg\DMS_JSON_CLIENTS.cfg`", e.g. "`C:\ProMoSNT\proj\promos\cfg\DMS_JSON_CLIENTS.cfg`"

The file contains all allowed client user names (CN) and serial number(s) of the client certificate.
`clientname:serialnumber(s)`

Multiple serial numbers are separated by ",", any valid serial number can be enabled by "*".

Example file content:

```
client1:*
ThirdPartyApp:3,4,5
```

Changes require restart of the DMS.

When the certificate is not present or invalid, the server will continue with Basic Authentication.

2.4.2.1.1 Server certificate

On a ProMoS V1/V2 installation the files are located in "`{INSTALL_DIR}\bin\`":

dms_cert.pem	The server certificate file
dms_key.pem	The server key file
dms_cert.p12	PKCS12 file for import in client app's

To regenerate the files, just remove any of the files above and restart DMS.
The serial will be increased on every generation (see here).

2.4.2.1.2 Client certificate

Example with OpenSSL to generate a client certificate.

1. Create the Client Key

```
openssl.exe genrsa -des3 -out dms_client1.key 4096
```

```
Generating RSA private key, 4096 bit long modulus
```

```
..................++.........++e is 65537 (0x10001)
Enter pass phrase for dms_client1.key:***
Verifying - Enter pass phrase for dms_client1.key:***
```

2. Create the Client CSR

```
openssl.exe req -new -key dms_client1.key -out dms_client1.csr
```

```
Enter pass phrase for dms_client1.key:***
You are about to be asked to enter information that will be incorporated into your
certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CH
State or Province Name (full name) [Some-State]:Bern
Locality Name (eg, city) []:Belp
Organization Name (eg, company) [Internet Widgits Pty Ltd]:XYZ
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:client1
Email Address []:client1@some.ch

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

The Common Name (CN) is used as user name to identify the clients certificate.

3. Sign the Client Certificate

```
openssl.exe x509 -req -days 365 -in dms_client1.csr -CA dms_cert.pem -CAkey
dms_key.pem -set_serial 01 -out dms_client1.crt
```

```
Signature ok
subject=/C=CH/ST=Belp/L=Belp/O=XYZ/CN=client1/emailAddress=client1@some.ch
Getting CA Private Key
```

Location of "`dms_cert.pem`" and "`dms_key.pem`" see previous chapter.

4. Convert Client Key to PKCS

```
openssl.exe pkcs12 -export -clcerts -in dms_client1.crt -inkey dms_client1.key -
out dms_client1.p12
```

```
Enter pass phrase for dms_client1.key:***
Enter Export Password:***
Verifying - Enter Export Password:***
```

"`dms_client1.p12`" is ready now to be installed in your browser or application.

Don't forget to insert "client1" in the clients users list.

2.4.2.2 Users/passwords for basic auth

On a ProMoS V1/V2 installation there is a user list required with password hashes:

"`{INSTALL_DIR}\proj\{PROJECT}\cfg\DMS_JSON_USERS.cfg`", e.g. "`C:\ProMoSNT\proj\promos\cfg\DMS_JSON_USERS.cfg`"

The list format must confirm to the UNIX standard for password hashing (passwd / shadow file), see also here.
Entries in the list:
`username:$id$salt$hashed`

Supported id's:

Id	Method
1	MD5 (**not** recommended)
apr1	Apache MD5 (**not** recommended)
5	SHA-256
6	SHA-512

Example file content:

```
test:$6$rounds=5000$6cD3q0iA38D/wZdT$TnCr0f.Tx7qu3.fEWcBdJwRPw2iinIIf9KSGl2OqYW0VpJ4I
username:$6$b21e5c208701eabf$NW1GcytKDLoZdMpTJcGYpd.dZ/2Pgz5KPeZQnRKAeeQbtWn/6rO2u/pR
```

- The password for the user "test" is "test1"
- The password for the user "username" is "testuserpassword"

Changes require restart of the DMS.

Tools to generate crypt(3) password hashes

Online:
https://quickhash.com/ (select Algorithm "SHA-512 /crypt(3) / $6$" or "SHA-512 /crypt(3) / $6$" or "MD5 / crypt(3) / $1$" or "" and salt or no salt for random salt generation)
http://www.cryptgenerator.de/

Command line tools:

```
php -r "print(crypt('password','salt') . \"\n\");"
mkpasswd -m sha-512
python -c 'import crypt; print crypt.crypt("password", "$6$random_salt")'
python3 -c 'import crypt; print(crypt.crypt("password", crypt.mksalt(crypt.METHOD_SHA512)))'
perl -e "print crypt('password','salt');"
ruby -e 'print "password".crypt("salt"); print("\n");'
htpasswd -nd user
openssl passwd -crypt myPassword
echo "select encrypt('password');" | mysql
```

2.4.2.3 Advanced configuration

On a ProMoS V1/V2 installation there is a configuration at the following location to allow advanced configurations:

"`{INSTALL_DIR}\proj\{PROJECT}\cfg\DMS.cfg`", e.g. "`C:\ProMoSNT\proj\promos\cfg\DMS.cfg`"

In the section "SSL" there are the following options:

`Cipher`	List of the preferenced ciphers (recommended default value see in the example below)
`UseSecureTls`	With value 1, only TLS V1.2 is allowed, 0 (**not** recommended) means older TLS versions allowed (default is 1).
`AuthRequired`	Enables (1) or disables (0 - **not** recommended) authentication on secure connections (default is 1).
`DMS_X509Serial`	Serial for next self signed certificates

By default (after first start of DMS) the file contains the following entries in the section "SSL":

```
[SSL]
Cipher=ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-SHA25
UseSecureTls=1
AuthRequired=1
```

Changes require restart of the DMS.

2.4.3 Advanced configuration

2.4.3.1 IP Lists

On a ProMoS V1/V2 installation there is a IP list configuration at the following location to block or allow connections:

"`{INSTALL_DIR}\proj\{PROJECT}\cfg\DMS_JSON_IPS.cfg`", e.g. "`C:\ProMoSNT\proj\promos\cfg\DMS_JSON_IPS.cfg`"

There are 4 sections to block / allow client IP's:
```
[nonSSL_Blocked]
[nonSSL_Allowed]
[SSL_Blocked]
[SSL_Allowed]
```

First, "xxx_Blocked" is processed and will block any found IP address.
When not blocked, "xxx_Allowed" will be processed and only IP addresses in this list will be allowed.

Below the corresponding section, IP ranges based on CIDR notation (see here) can be defined.
Comments begin with "#" on the first column.

By default (after first start of DMS) the file contains the following entries:

```
# IP list for DMS JSON communication

[nonSSL_Blocked]
# None

[nonSSL_Allowed]
# Any connection from local host
127.0.0.1
# MST portals
192.168.96.0/20
172.18.8.0/23

[SSL_Blocked]
# None

[SSL_Allowed]
# Any connection
0.0.0.0/0
```

Changes require restart of the DMS.

2.5 Base Path

The base path for all data exchanges is: "`/json_data`".

2.6 Error Messages

- HTTP-POST:
  You will get usual HTTP response codes and a plain text error message in body in case of fatal errors.

- WebSocket
  You will get a usual WebSocket close messages in case of fatal errors.

2.6.1 Examples

```
GET /json_data HTTP/1.1
Connection: keep-alive
Content-Type: application/json
Host: 10.6.40.2:9020
User-Agent: Apache-HttpClient/4.2.6 (java 1.5)

HTTP/1.1 405 OK
Date: Thu, 19 Mar 2015 14:10:12 GMT
Connection: Keep-Alive
Server: ProMoS DMS/1.0
Transfer-Encoding: chunked
Content-Type: text/html;charset=UTF-8

12
Use POST requests.
0
```

```
WebSocket
  1... .... = Fin: True
  .000 .... = Reserved: 0x00
  .... 1000 = Opcode: Connection Close (8)
  0... .... = Mask: False
  .000 1111 = Payload length: 15
  Payload
    Close: Unsupported Data (1003)
    Reason: Invalid path.
```

2.7 HTTP Example Message

This chapter shows a whole communication stream of a HTTP POST request and response.

All further examples in the next chapters only show the JSON data content (Body for HTTP POST / Payload message data for Websocket).

Request:

```
POST /json_data HTTP/1.1
Connection: keep-alive
Content-Type: application/json
Content-Length: 140
Host: 10.6.40.2:9020
User-Agent: Apache-HttpClient/4.2.6 (java 1.5)

{
  "get": [
    {"path":"EXMPL1:T11:MN:003:Vis:VMC_energy1"},
    {"path":"EXMPL1:T11:MN:003:Vis:VEnergy1V"},
    {"path":"EXMPL1:T11:MN:003:Vis:VMC_power"}
  ]
}
```

Response:

```
HTTP/1.1 200 OK
Date: Fri, 20 Mar 2015 06:53:34 GMT
Connection: Keep-Alive
Server: ProMoS DMS/1.0
Transfer-Encoding: chunked
Content-Type: application/json;charset=UTF-8

160
{
  "get": [
    {
      "path":"EXMPL1:T11:MN:003:Vis:VMC_energy1",
      "code":"ok",
      "type":"double",
      "value":3.0,
      "stamp":"2015-03-20T07:49:19,000+01:00"
    },
    {
      "path":"EXMPL1:T11:MN:003:Vis:VEnergy1V",
      "code":"ok",
      "type":"double",
      "value":0.0,
      "stamp":"2015-03-20T07:49:19,000+01:00"
    },
    {
      "path":"EXMPL1:T11:MN:003:Vis:VMC_power",
      "code":"ok",
      "type":"double",
      "value":0.597,
      "stamp":"2015-03-20T07:49:19,000+01:00"
    }
  ]
}
0
```

2.8 Timestamps

All timestamps are ISO 8601 formatted.

Timestamps from DMS:

All transmitted timestamps from DMS are formatted as follows (date/time with millisecond fractions and time offset from UTC):
`JJJJ'-'MM'-'TT'T'hh':'mm':'ss','fff'±'hh':'mm`

Examples (for time zone Europe/Zurich, CET/CEST):
- UTC "2015-03-20T07:49:19,000Z" will be transmitted as "2015-03-20T08:49:19,000+01:00"
- UTC "2015-04-28T07:10:11,000Z" will be transmitted as "2015-04-28T09:10:11,000 +02:00" (including daylight saving time offset)

Notes:
- Due to non persistent storage of the time stamp in DMS, the transmitted time stamp can be 'null' after a restart of the DMS.
- Timestamps on nodes with no data (type "none") return 'null'.

Timestamps to DMS:

You can use any valid ISO 8601 format, but it is recommended to use the following format (UTC date/time with millisecond fractions):
`JJJJ'-'MM'-'TT'T'hh':'mm':'ss','fff'Z'`

It is important to add the UTC time zone indicator ('Z') or a time offset from UTC ('±'...), as otherwise the DMS can not evaluate the time zone of the sender!

Example:
- 2015-04-28T07:10:11,023Z
- 2015-04-28T07:10:11Z

2.9 Message Tag

Beside the tag data in every command, it is possible to send a tag object (any valid JSON object - except null) in the request that will be echoed in the response.

Due to better readability these tags are not documented in the following chapters.

Example:
Request:

```
{
  "tag": {
    "reqnr": 1456,
    "flag": true
  },
  "get": [
    {
      "path":"EXMPL1:T11:MN:003:Vis:VMC_power"
    }
  ]
}
```

Response:

```
{
  "tag": {
    "reqnr": 1456,
    "flag": true
  },
  "get": [
    {
      "path":"EXMPL1:T11:MN:003:Vis:VMC_power",
      "code":"ok",
      "type":"double",
      "value":4.4444,
      "stamp":"2015-03-20T07:49:19,000+01:00"
    }
  ]
}
```

3 Data Points

3.1 Command Overview

Command	Description
get	Read data point value(s).
set	Write data point value(s).
delete	Delete data point(s).
rename	Rename data point(s).
subscribe	Monitoring data point(s). Only for WebSocket connection.
unsubscribe	Unsubscribe monitored data point(s). Only for WebSocket connection.

You can handle as much data points as you want with 1 single request.
Due to performance and available resources, it's recommend to not handle more then 10`000 data points at once.

3.1.1 Identification

Any connected client has to identify him self for commands with write access (Set/Rename/Delete).

This requires the request field "whois".
When a client has authenticated the connection (Certificate or Basic Auth) the "whois" field is optional and the authenticated user name will be used for the corresponding command.

3.1.2 Response Order

The response arrays are ordered in the same manner as in the request.

See also from the JSON specification at http://www.json.org/: "An array is an *ordered* sequence of zero or more values.".

3.1.3 Multiple Commands

It is recommended to use only one command per request!
You can use multiple command in one request, but it is not guarantied that the commands are processed in the desired order.

See also from the JSON specification at http://www.json.org/: "An object is an *unordered* set of name/value pairs".

Example:

Request:

```
{
  "whois":"DriverXY",
  "user":"",
  "set": [
    {
      "path":"EXMPL1:T11:MN:003:Vis:VMC_power",
      "value":4.4444
    }
  ],
  "get": [
    {
      "path":"EXMPL1:T11:MN:003:Vis:VMC_power"
    }
  ]
}
```

could result in:

Response:

```
{
  "set": [
    {
      "path":"EXMPL1:T11:MN:003:Vis:VMC_power",
      "code":"ok",
      "type":"double",
      "value":4.4444,
      "stamp":"2015-03-20T07:49:19,000+01:00"
    }
  ],
  "get": [
    {
      "path":"EXMPL1:T11:MN:003:Vis:VMC_power",
      "code":"ok",
      "type":"double",
      "value":4.4444,
      "stamp":"2015-03-20T07:49:19,000+01:00"
    }
  ]
}
```

or Response:

```
{
  "get": [
    {
      "path":"EXMPL1:T11:MN:003:Vis:VMC_power",
      "code":"ok",
      "type":"double",
      "value":1.1111,
      "stamp":"2015-03-20T04:23:44,000+01:00"
    }
  ],
  "set": [
    {
      "path":"EXMPL1:T11:MN:003:Vis:VMC_power",
      "code":"ok",
      "type":"double",
      "value":4.4444,
      "stamp":"2015-03-20T07:49:19,000+01:00"
    }
  ]
}
```

(where the value in the get response is an **old** value).

3.1.4 String Arrays

In ProMoS NT, Version 1.x/2.x there is a special definition for reading/writing array indexed strings:

You can use a string field to store array data and read/write to them with a desired index. First index is starting with 1.

Precondition: There must exist a string with valid starting ('(' or '{') and ending (')' or '}') array indicators. The array elements are separated with ','.

Example:
The string in the data point "`TEST:ARRAY`" is "`(11,12,13,14)`".
Now you can read an indexed value by reading data point "`TEST:ARRAY`**`[2]`**" -> this will return the value '12'.
On the other side, you can indexed write by writing on data point "`TEST:ARRAY`**`[4]`**" (e.g. value '44'), this will set the string on "`TEST:ARRAY`" to "`(11,12,13,44)`".

Any read outside a existing index will return an error.
Any write outside a valid index (range 1..1024) will return an error.
Any write with a valid index outside a existing index will expand the array and fill newly created index values with 'NULL'.

Default data types are "int". If you desire an other data type, it is necessary to add a suffix after the closing bracket (']') :
- BIT for boolean indexed values.
- FLT for double indexed values.

Example:
"`TEST:ARRAY`" is "`(T,T,F,T)`".
Reading from "`TEST:ARRAY[2]`**`BIT`**" will return true as boolean.

3.1.5 Path Limitation

In ProMoS NT, Version 1.x the maximum length of the data point path is limited to 80 characters!
In ProMoS NT, Version 2.x the maximum length of the data point path is limited to 160 characters!
Any attempt to write longer paths will be rejected.

3.2 Get

3.2.1 Example

Request:

```
{
  "get": [
    {"path":"EXMPL1:T11:MN:003:Vis:VMC_energy1"},
    {"path":"EXMPL1:T11:MN:003:Vis:VEnergy1V"},
    {"path":"EXMPL1:T11:MN:003:Vis:VMC_power"}
  ]
}
```

Response:

```
{
  "get": [
    {
      "path":"EXMPL1:T11:MN:003:Vis:VMC_energy1",
      "code":"ok",
      "type":"double",
      "value":3.0,
      "stamp":"2015-03-20T07:49:19,000+01:00"
    },
    {
      "path":"EXMPL1:T11:MN:003:Vis:VEnergy1V",
      "code":"ok",
      "type":"double",
      "value":0.0,
      "stamp":"2015-03-20T07:49:19,000+01:00"
    },
    {
      "path":"EXMPL1:T11:MN:003:Vis:VMC_power",
      "code":"ok",
      "type":"double",
      "value":0.597,
      "stamp":"2015-03-20T07:49:19,000+01:00"
    }
  ]
}
```

Response in case of fatal error:

```
{
  "get": [
    {
      "code":"error",
      "message":"Expected JSON encoded data, but got something else."
    }
  ]
}
```

3.2.2 Querying Path/Value

It is possible to search data point paths and values with query parameters (see here).

For RegEx parameters, Perl 5 syntax and semantics have to be used - corresponding the used library (PCRE in version 8.35). See also PCRE documentation.
For Perl 5 regular expression syntax, read the Perl regular expressions man page.
Extended-Patterns can also be used.

The "path" parameter in the request means the starting data point for the query.

Any query with a resulting data point list size > 100'000 will be aborted and returns a error message.

3.2.2.1 Example

Request:
(comments just to clarify)

```
{
  "get": [
    { /* get first child path's from the DMS */
      "path":"",
      "query": {
      }
    },
    { /* get all path's from the whole DMS, but without the BMO: tree */
      "path":"",
      "query": {
        "regExPath": "^(?!BMO).*$",
        "maxDepth": 0
      }
    },
    { /* get all path's from the whole DMS, but without BMO: and System: tree */
      "path":"",
      "query": {
        "regExPath": "^(?!(BMO|System)).*$",
        "maxDepth": 0
      }
    },
    { /* get all values on path "Istwert", incl. sub path's */
      "path":"EXMPL1:T11",
      "query": {
        "regExPath": ".*:Istwert",
        "maxDepth": 0
      }
    },
    { /* get all values with content 0 on path "Istwert", incl. sub path's */
      "path":"EXMPL1:T11",
      "query": {
        "regExPath": ".*:Istwert",
        "regExValue": "[0]",
        "maxDepth": 0
      }
    },
    { /* get all values with bool content true on the whole DMS */
      "path":"",
      "query": {
        "regExValue": "true",
        "isType": "bool",
        "maxDepth": 0
      }
    },
    { /* get all values who have historical data from the whole DMS */
      "path":"EXMPL1:T11",
      "query": {
        "hasHistData": true,
        "maxDepth": 0
      }
    }
  ]
}
```

Response with the found data points see this example.

3.2.3 Read Historical Data

It is possible to read out historical data from data points - see also query with "hasHistData".

The resulting response array ("histData") is limited to max. 610'000 entries! This will meet

max. ~17 years (with interval 15 minutes) or ~7 days (with interval 1 second).
Refine your period (start/end) or interval to receive the desired historical data.

A combination of query and histData is possible, e.g. to read out historical data (1 day) from all historical value data points:

Request:

```
{
  "get": [
    {
      "path": "",
      "query": {
        "hasHistData": true,
        "maxDepth": 0
      },
      "histData": {
        "start": "2015-04-04T00:00:00Z",
        "end": "2015-04-05T00:00:00Z",
        "interval": 900,
        "format":"detail"
      }
    }
  ]
}
```

3.2.3.1 Example, Compact

Request:

```
{
  "get": [
    {
      "path": "System:NT:Perf:SYSTEM",
      "histData": {
        "start": "2015-04-01T00:00:00Z",
        "interval":100
      }
    }
  ]
}
```

Response:

```
{
  "get": [
    {
      "path":"System:NT:Perf:SYSTEM",
      "code":"ok",
      "type":"double",
      "value":0.0,
      "stamp":null,
      "histData":
      [
        {
          "2015-04-03T04:33:20,000+02:00":0.32780084013938906
        },
        {
          "2015-04-03T04:35:00,000+02:00":0.7427386045455933
        },
        {
          "2015-04-03T04:36:40,000+02:00":0.9577777981758118
        },
        {
          "2015-04-03T04:38:20,000+02:00":0.846666693687439
        },
        {
          "2015-04-03T04:40:00,000+02:00":0.7355555295944214
        },
        ...
        {
          "2015-04-17T08:37:40,000+02:00":0.6244444251060486
        }
      ]
    }
  ]
}
```

3.2.3.2 Example, Detail

Request:

```
{
  "get": [
    {
      "path": "System:NT:Perf:DMS",
      "histData": {
        "start": "2015-01-01T00:00:00Z",
        "end": "2015-05-31T00:00:00Z",
        "format":"detail"
      }
    }
  ]
}
```

Response:

```
{
  "get": [
    {
      "path":"System:NT:Perf:DMS",
      "code":"ok",
      "type":"double",
      "value":0.0,
      "stamp":null,
      "histData":
      [
        {
          "stamp":"2015-04-03T04:45:00,000+02:00",
          "value":0.0,
          "state":"ok",
          "rec":"cycle"
        },
        {
          "stamp":"2015-04-03T05:00:00,000+02:00",
          "value":0.0,
          "state":"ok",
          "rec":"cycle"
        },
        {
          "stamp":"2015-04-03T05:15:00,000+02:00",
          "value":0.0,
          "state":"ok",
          "rec":"cycle"
        },
        ...
        {
          "stamp":"2015-06-01T02:00:00,000+02:00",
          "value":0.0,
          "state":"ok",
          "rec":"cycle"
        }
      ]
    }
  ]
}
```

3.2.4 Read Change Log

It is possible to read out the change log from data points - see also query with "hasChangelog".

The resulting response array ("changelog") is limited to max. 100'000 entries.
Refine your period (start/end) to receive the desired change log.

Every change of a logged data point is collected by a change log group. To get change logs for the corresponding group use change log groups.

3.2.4.1 Example

Request:

```
{
  "get": [
    {
      "path": "BHS60:AV:506:ABS_Ein",
      "showExtInfos": ["changelogGroup"],
      "changelog": {
        "start": "2016-02-29T00:00:00Z",
        "end": "2016-03-02T00:00:00Z"
      }
    }
  ]
}
```

Response:

```
{
  "get": [
    {
      "code": "ok",
      "path": "BN028:H04:VS:001:ABS_Ein",
      "type": "bool",
      "value": false,
      "stamp": null,
      "extInfos": {
        "changelogGroup": "ABS1"
      },
      "changelog":
      [
        {
          "stamp": "2016-03-01T11:54:48,536+01:00",
          "text": "Heizungsventil Aus"
        },
        {
          "stamp": "2016-03-01T11:48:48,283+01:00",
          "text": "Heizungsventil Ein"
        },
        {
          "stamp": "2016-02-29T11:48:55,715+01:00",
          "text": "Heizungsventil Aus"
        },
        {
          "stamp": "2016-02-29T11:42:58,033+01:00",
          "text": "Heizungsventil Ein"
        }
      ],
      "hasChild": true
    }
  ]
}
```

3.2.5 Request Fields

The "get" array of objects:

Field	Description	Type	M/O
path	The path of the data point you want to read.	string	Mandatory
showExtInfos	Returns optional information's like template, name, unit.... Possible values see here.	array of string	Optional

Field	Description	Type	M/O
query	The query object (definitions see below), default is no query.	object	Optional
histData	Definitions for requesting historical data (definitions see below), default is no histData.	object	Optional
changelog	Definitions for requesting change logs (definitions see below), default is no changelog	object	Optional
tag	Any data that will be echoed in the response.	*any*	Optional

The "query" object:

Field	Description	Type	M/O
regExPath	The RegEx pattern for the path. Default is none (empty).	string	Optional
regExValue	The RegEx pattern for the value. Default is none (empty). To be sure to find the correct value (e.g. bool, true) use this field in combination with 'isType'.	string	Optional
regExStamp	The RegEx pattern for the timestamp. Default is none (empty). The searched time stamp is composed as described here.	string	Optional
isType	A string containing searched types ("none", "bool", "int", "double", "string"). Default is all types. To request multiple types just separate the strings, e.g. "int,double".	string	Optional
hasHistData	A Filter to read only data points with historical data recording. Default is false.	boolean	Optional
hasChangelog	A Filter to read only data points with change log. Default is false.	boolean	Optional
hasAlarmData	A Filter to read only data points with alarm data / data recording. Default is false.	boolean	Optional
maxDepth	Maximal depth for searching path's recursive. Default is 1 (current path). 0 means no restrictions, all sub paths are searched.	number	Optional

The "histData" object:

Field	Description	Type	M/O
start	The requested start time stamp for the data (see also here).	string	Mandatory
end	The requested end time stamp for the data (see also here). Default is the current time stamp.	string	Optional
interval	Interval of requested data in seconds. Default is 15 minutes (900). Interval 0 means all recorded data, not interpolated.	number	Optional
format	Can be "compact" or "detail". Default is "compact".	string	Optional

The "changelog" object:

Field	Description	Type	M/O
start	The requested start time stamp for the change log (see also here).	string	Mandatory
end	The requested end time stamp for the change log (see also here). Default is the current time stamp.	string	Optional

3.2.6 Response Fields

The "get" array of objects:

Field	Description	Type	Occurrence
code	The code can be: - "ok": On success. - "no perm": You are not allowed to read this data point. - "not found": This data point doesn't exist on the system. - "error": In case of a fatal error.	string	Always
path	The path of the requested data point.	string	When no fatal error
value	The value of the data point.	number, boolean, string, null	On code "ok"
type	The type of the value, can be "int", "double" (a floating point number), "string", "bool" or "none" for nodes (without value).	string	On code "ok"
hasChild	Indicates whenever a data point has one or more child data point(s).	boolean	On code "ok", only, when node has child (s)
stamp	Time stamp of the last change of the value. This field can be 'null', otherwise ISO 8601 formatted (see here).	null, string	On code "ok"

extInfos	Extended information's for this data point, see below.	object	On code "ok", when showExtInfos was requested
message	This field contains an human readable error message in English.	string	Other than code "ok"
histData	An array with historical data records (see below).	array	On code "ok", when histData was requested
changelog	An array with change log records (see below).	array	On code "ok", when changelog was requested
tag	Tag data from request.	*from request*	When existing in request

The "extInfos" object:

Field	Description	Type	Occurrence
state	The current state of this value, can be "ok", "error" (e.g. communication error to the end device)	string	Always
accType	The accurate type information, can be "int8", "uint8", "int16", "uint16", "int32", "uint32", "int64", "uint64", "double32", "double64", "string", "bool" or "none" for nodes (without value).	string	Always
name	For ProMoS 1/2: the content of the topmost NAME data point on this tree.	string	When existing
template	The used template, for ProMoS 1/2: the content of the topmost OBJECT data point on this tree.	string	When existing
unit	Any present unit information.	string	When existing

comment	Any present comment.	string	When existing
changelogGroup	Group name for change log.	string	When existing

The "histData" array of objects for request format "compact":

Field	Description	Type	Occurrence
time stamp	A number object that is named with the corresponding time stamp, e.g.: { "2015-04-03T04:33:20,000+02:00": 0.95 }	number	Always

The "histData" array of objects for request format "detail":

Field	Description	Type	Occurrence
stamp	Time stamp of the historic entry, ISO 8601 formatted (see here).	string	Always
value	The historic value.	number	Always
state	State of this value, can be "ok", "comErr" (recorded during communication error) or "inv" (any other error situation).	string	Always
rec	Record reason, can be "cycle", "change", "diff" or "unknown".	string	Always

The "changelog" array of objects:

Field	Description	Type	Occurrence
path	The path of the corresponding data point.	string	Only when not requested for a single data point
stamp	Time stamp of the log entry, ISO 8601 formatted (see here).	string	Always
text	The log text.	string	Always

Additional fields in case of an alarm data point:

state	The state of the alarm, can be "occurred", "left" or "acknowledged"	string	Always
priority	The alarm priority	number	Always
priority BACnet	The alarm BACnet priority	number	Always
alarmGroup	The alarm group	number	Always

state	The state of the alarm, can be "occurred", "left" or "acknowledged"	string	Always
alarmCollectGroup	The alarm collective group	number	Always
siteGroup	The alarm site group	number	Always
screen	The scada screen name	string	When existing

3.2.7 JSON Schema

Request:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Read Request",
  "description": "Reading one or more data points",
  "type": "object",
  "properties": {
    "get": {
      "description": "The command",
      "type": "array",
      "items": {
        "title": "Data point definition",
        "type": "object",
        "properties": {
          "path": {
            "description": "The DMS path to the data point",
            "type": "string"
          },
          "showExtInfos": {
            "description": "Optional request of extented informations",
            "type": "boolean"
          },
          "query": {
            "description": "Optional query parameters",
            "type": "object",
            "properties": {
              "regExPath": {
                "description": "RegEx pattern for the path",
                "type": "string"
              },
              "regExValue": {
                "description": "RegEx pattern for the value",
                "type": "string"
              },
              "regExStamp": {
                "description": "RegEx pattern for the time stamp",
                "type": "string"
              },
              "isType": {
                "description": "Type filters",
                "type": "string",
                "enum": [ "int", "double", "string", "bool", "none" ]
              },
              "hasHistData": {
                "description": "Filter for data points with historical data",
                "type": "boolean"
              },
              "hasChangelog": {
                "description": "Filter for data points with change log",
                "type": "boolean"
              },
              "hasAlarmData": {
                "description": "Filter for data points with alarm",
                "type": "boolean"
              },
              "maxDepth": {
                "description": "Maximal depth for searching in sub path's",
                "type": "number"
              }
            }
          },
          "histData": {
            "description": "Optional parameters for historical data",
            "type": "object",
            "properties": {
              "start": {
                "description": "Time stamp for start",
                "type": "string"
              },
              "end": {
                "description": "Time stamp for end",
                "type": "string"
              },
              "interval": {
                "description": "Interval in seconds",
                "type": "number"
              },
              "format": {
                "description": "Format for the response",
                "type": "string",
                "enum": [ "compact", "detail" ]
              }
            },
            "required": ["start"]
          },
          "changelog": {
            "description": "Optional parameters for change log data",
            "type": "object",
            "properties": {
              "start": {
                "description": "Time stamp for start",
                "type": "string"
              },
              "end": {
                "description": "Time stamp for end",
                "type": "string"
              }
            },
            "required": ["start"]
          },
          "tag": {
            "description": "Any data, will be echoed on the response",
            "type": [ "object", "array", "number", "string", "boolean"]
          }
        },
        "additionalProperties": false,
        "required": ["path"]
      },
      "minItems": 1
      }
    },
    "required": ["get"]
}
```

Response:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Read Response",
  "description": "Information about one or more data points",
  "type": "object",
  "properties": {
    "get": {
      "description": "The command",
      "type": "array",
      "items": {
        "title": "Data point value",
        "type": "object",
        "properties": {
          "code": {
            "description": "The result code",
            "type": "string",
            "enum": [ "ok", "no perm", "not found", "error" ]
          },
          "path": {
            "description": "The DMS path to the data point",
            "type": "string"
          },
          "value": {
            "description": "The value of the data point",
            "type": ["number", "string", "boolean", "null"]
          },
          "type": {
            "description": "The value type",
            "type": "string",
            "enum": [ "int", "double", "string", "bool", "none" ]
          },
          "stamp": {
            "description": "The timestamp of the last change of the value, ISO
8601",
            "type": ["string", "null"]
          },
          "extInfos": {
            "title": "Extended informations for this data point",
            "type": "object",
            "properties": {
              "template": {
                "description": "The used template",
                "type": "string",
              },
              "name": {
                "description": "The name",
                "type": "string",
              },
              "unit": {
                "description": "Any present unit information",
                "type": "string",
              }
            }
          },
          "message": {
            "description": "Human readable error message",
            "type": "string"
          },
          "histData": {
            "description": "Array of the requested historical data",
            "type": "array",
            "items": {
              "title": "Historical data",
              "type": "object",
              "properties": {
                "stamp": {
                  "description": "The timestamp of the recorded value, ISO 8601",
                  "type": ["string"]
                },
                "value": {
                  "description": "The value",
                  "type": ["number"]
                },
                "state": {
                  "description": "The recording state",
                  "type": "string",
                  "enum": [ "ok", "comErr", "inv" ]
                },
                "rec": {
                  "description": "The recording reason",
                  "type": "string",
                  "enum": [ "cycle", "change", "diff", "unknown" ]
                }
              }
            }
          },
          "changelog": {
            "description": "Array of the requested change log",
            "type": "array",
            "items": {
              "title": "Change log",
              "type": "object",
              "properties": {
                "stamp": {
                  "description": "The timestamp of the recorded value, ISO 8601",
                  "type": ["string"]
                },
                "text": {
                  "description": "The text",
                  "type": ["string"]
                }
              }
            }
          },
          "tag": {
            "description": "Echo from the request",
            "type": [ "object", "array", "number", "string", "boolean"]
          }
        },
        "required": ["code"]
      }
    },
    "minItems": 1
  },
  "required": ["get"]
}
```

3.2.8 Short Request

For maximum performance and minimum data size, a short variant of the read request can be used.

The response looks like the normal response.

Note: With this request, the order of the response fields is not guarantied (See also from the JSON specification at http://www.json.org/: "An object is an unordered set of name/value pairs".).

3.2.8.1 Example

Request:

```
{
  "get": [
    "EXMPL1:T11:MN:003:Vis:VMC_energy1",
    "EXMPL1:T11:MN:003:Vis:VEnergy1V",
    "EXMPL1:T11:MN:003:Vis:VMC_power"
  ]
}
```

Response see this example.

3.2.8.2 JSON Schema

Request:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Read Request",
  "description": "Reading one or more data points",
  "type": "object",
  "properties": {
    "get": {
      "description": "The command",
      "type": "array",
      "items": {
        "title": "Data point path",
        "type": "string"
      },
      "minItems": 1
    }
  },
  "required": ["get"]
}
```

3.3 Set

3.3.1 Example

Request:

```
{
  "whois":"DriverXY",
  "user":"",
  "set": [
    {
      "path":"EXMPL1:T11:MN:003:Vis:VMC_energy1",
      "value":4.4565467567867,
      "type":"double"
    },
    {
      "path":"EXMPL1:T11:MN:003:Vis:VEnergy1V",
      "value":-1,
      "type":"double",
      "stamp":"2015-03-11T05:27:39,027+01:00"
    },
    {
      "path":"EXMPL1:TEST:BOOLEAN",
      "value":true,
      "create": true,
      "type":"bool"
    },
    {
      "path":"EXMPL1:TEST:INT",
      "value":44,
      "type":"int"
    },
    {
      "path":"EXMPL1:TEST:STRING",
      "value":"some long example message",
      "type":"string"
    }
  ]
}
```

Response::

```
{
  "set": [
    {
      "code":"ok",
      "path":"EXMPL1:T11:MN:003:Vis:VMC_energy1",
      "value":3,
      "type":"double",
      "stamp":"2015-03-20T07:49:19,000+01:00"
    },
    {
      "code":"ok",
      "path":"EXMPL1:T11:MN:003:Vis:VEnergy1V",
      "value":0,
      "type":"double",
      "stamp":"2015-03-11T05:27:39,027+01:00"
    },
    {
      "code":"no perm",
      "path":"EXMPL1:TEST:BOOLEAN",
    },
    {
      "code":"error",
      "path":"EXMPL1:TEST:INT",
      "message":"Data point doesn't exist"
    },
    {
      "code":"error",
      "path":"EXMPL1:TEST:STRING",
      "message":"Data type doesn't match"
    }
  ]
}
```

3.3.2 Request Fields

Root objects:

Field	Description	Type	M/O
whois	The identification of the sender, e.g. "sDriver".	string	Mandatory (note)
user	Any logged user name, empty when there is no current user.	string	Mandatory

The "set" array of objects:

Field	Description	Type	M/O
path	The path of the data point you want to write.	string	Mandatory
create	Flag for create datapoint when not existing (default is false).	boolean	Optional
value	The value to write (see note below). To create a node only (without data) you can send '"value": null' (and '"create":true') on a non existing data point.	number, boolean, string, null	Mandatory

Field	Description	Type	M/O
type	The type of the value. The API will check if the value type on control system matches.The type can be "int", "double" (a floating point number), "string", "bool". By default, the type is evaluated from JSON data type (see note below).	string	Optional
stamp	ISO 8601 formatted, see also here (default is current time stamp).	string	Optional
tag	Any data that will be echoed in the response.	*any*	Optional

Important notes about types:

The type of the written value is evaluated from the transmitted JSON data type, examples:

Value	JSON type	internal type
"ssttrr"	string	string
true / false	boolean	bool
123	number	int
123.0 / 123.4	number	double

Some JSON data writers will convert a double value without decimals to a JSON value without decimals (e.g. 123.0 will be transmitted as 123)

Due to this fact, it is allowed to write a "int" value (without "type" field) to a "double" data point (e.g. 123).

But: When you create a new value, the server has no chance to evaluate the right value without "type" field.
-> Writing 123 to a non existing data point with "create":true and no "type" field will create a "int" data point.
=> If you want to be sure that a "double" data point is created: use the "type" field with value "double"!

3.3.3 Response Fields

The "set" array of objects:

Field	Description	Type	Occurrence
code	The code can be: - "ok": On success. - "no perm": You are not allowed to write this data point. - "not found": This data point doesn't exist on the system. - "error": Something went wrong while writing to the control system.	string	Always

path	The path of the written data point.	string	When no fatal error
value	The value that you have set.	number, boolean, string, null	On code "ok"
type	The type of the value, can be "int", "double" (a floating point number), "string", "bool" or "none" for nodes without values.	string	On code "ok"
stamp	ISO 8601 formatted (see here).	string, null	On code "ok"
message	This field contains an human readable error message in English.	string	Other than code "ok"
tag	Tag data from request.	*from request*	When existing in request

3.3.4 JSON Schema

Request:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Write Request",
  "description": "Writing one or more data points",
  "type": "object",
  "properties": {
    "whois": {
      "description": "Identification of the sender",
      "type": "string"
    },
    "user": {
      "description": "Username",
      "type": "string"
    },
    "set": {
      "description": "The command",
      "type": "array",
      "items": {
        "title": "Data point write definition",
        "type": "object",
        "properties": {
          "path": {
            "description": "The DMS path to the data point",
            "type": "string"
          },
          "create": {
            "description": "Flag to create non existing datapoint",
            "type": "boolean"
          },
          "value": {
            "description": "The new value of the data point",
            "type": ["number", "string", "boolean", "null"]
          },
          "type": {
            "description": "The value type",
            "type": "string",
            "enum": [ "int", "double", "string", "bool" ]
          },
          "tag": {
            "description": "Any data, will be echoed on the response",
            "type": [ "object", "array", "number", "string", "boolean"]
          }
        },
        "additionalProperties": false,
        "required": ["path", "value", "type"]
      }
    },
    "minItems": 1
  },
  "required": ["whois", "user", "set"]
}
```

Response::

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Write Response",
  "description": "Information about writing one or more data points",
  "type": "object",
  "properties": {
    "set": {
      "description": "The command",
      "type": "array",
      "items": {
        "title": "Data point value",
        "type": "object",
        "properties": {
          "code": {
            "description": "The result code",
            "type": "string",
            "enum": [ "ok", "no perm", "not found", "error" ]
          },
          "path": {
            "description": "The DMS path to the data point",
            "type": "string"
          },
          "value": {
            "description": "The value of the data point",
            "type": ["number", "string", "boolean", "null"]
          },
          "type": {
            "description": "The value type",
            "type": "string",
            "enum": [ "int", "double", "string", "bool", "none" ]
          },
          "stamp": {
            "description": "The timestamp of the last change of the value, ISO
8601",
            "type": ["string", "null"]
          },
          "message": {
            "description": "Human readable error message",
            "type": "string"
          },
          "tag": {
            "description": "Echo from the request",
            "type": [ "object", "array", "number", "string", "boolean"]
          }
        },
        "required": ["code"]
      }
    },
    "minItems": 1
  },
  "required": ["set"]
}
```

3.4 Rename

3.4.1 Example

Request:

```
{
  "whois":"DriverXY",
  "rename": [
    {
      "path":"EXMPL1:T11:MN:003"
      "newPath":"EXMPL1:T11:MN:002"
    },
    {
      "path":"EXMPL1:T11:MN:003",
      "newPath":"EXMPL1:T11:MN:002"
    }
  ]
}
```

Response::

```
{
  "rename": [
    {
      "code":"ok",
      "path":"EXMPL1:T11:MN:003"
      "newPath":"EXMPL1:T11:MN:002"
    },
    {
      "code":"not found",
      "path":"EXMPL1:T11:MN:003",
      "message":"Data point doesn't exist"
    }
  ]
}
```

3.4.2 Request Fields

Root objects:

Field	Description	Type	M/O
whois	The identification of the sender, e.g. "sDriver".	string	Mandatory (note)

The "rename" array of objects:

Field	Description	Type	M/O
path	The source path of the data point you want to rename.	string	Mandatory
newPath	The destination path.	string	Mandatory
tag	Any data that will be echoed in the response.	*any*	Optional

3.4.3 Response Fields

The "rename" array of objects:

Field	Description	Type	Occurrence

code	The code can be: - "ok": On success. - "no perm": You are not allowed to rename this data point. - "not found": This data point doesn't exist on the system. - "error": Something went wrong while renaming.	string	Always
path	The path of the original data point.	string	When no fatal error
newPath	The renamed path.	string	When no fatal error
message	This field contains an human readable error message in English.	string	Other than code "ok"
tag	Tag data from request.	*from request*	When existing in request

3.4.4 JSON Schema

Request:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Rename Request",
  "description": "Renaming one or more data points",
  "type": "object",
  "properties": {
    "whois": {
      "description": "Identification of the sender",
      "type": "string"
    },
    "rename": {
      "description": "The command",
      "type": "array",
      "items": {
        "title": "Data point rename definition",
        "type": "object",
        "properties": {
          "path": {
            "description": "The DMS path to the old data point",
            "type": "string"
          },
          "newPath": {
            "description": "The DMS path to the new data point",
            "type": "string"
          },
          "tag": {
            "description": "Any data, will be echoed on the response",
            "type": [ "object", "array", "number", "string", "boolean"]
          }
        },
        "additionalProperties": false,
        "required": ["path", "newPath"]
      }
    },
    "minItems": 1
  },
  "required": ["whois", "rename"]
}
```

Response::

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Rename Response",
  "description": "Information about renaming one or more data points",
  "type": "object",
  "properties": {
    "rename": {
      "description": "The command",
      "type": "array",
      "items": {
        "title": "Data point value",
        "type": "object",
        "properties": {
          "code": {
            "description": "The result code",
            "type": "string",
            "enum": [ "ok", "no perm", "not found", "error" ]
          },
          "path": {
            "description": "The DMS path to the old data point",
            "type": "string"
          },
          "newPath": {
            "description": "The DMS path to the new data point",
            "type": "string"
          },
          "message": {
            "description": "Human readable error message",
            "type": "string"
          },
          "tag": {
            "description": "Echo from the request",
            "type": [ "object", "array", "number", "string", "boolean"]
          }
        },
        "required": ["code"]
      }
    },
    "minItems": 1
  },
  "required": ["rename"]
}
```

3.5 Delete

3.5.1 Example

Request:

```
{
  "whois":"DriverXY",
  "delete": [
    {
      "path":"EXMPL1:T11:MN:003"
    },
    {
      "path":"EXMPL1:T11:MN:003",
      "recursive": true
    },
    {
      "path":"EXMPL1:T11:MN:003:Vis:VEnergy1V"
    },
    {
      "path":"EXMPL1:TEST:BOOLEAN"
    },
  ]
}
```

Response::

```
{
  "delete": [
    {
      "code":"error",
      "path":"EXMPL1:T11:MN:003",
      "message":"Path is not empty"
    },
    {
      "code":"ok",
      "path":"EXMPL1:T11:MN:003"
    },
    {
      "code":"not found",
      "path":"EXMPL1:T11:MN:003:Vis:VEnergy1V",
      "message":"Data point doesn't exist"
    },
    {
      "code":"ok",
      "path":"EXMPL1:TEST:BOOLEAN",
    }
  ]
}
```

3.5.2 Request Fields

Root objects:

Field	Description	Type	M/O
whois	The identification of the sender, e.g. "sDriver".	string	Mandatory (note)

The "delete" array of objects:

Field	Description	Type	M/O
path	The path of the data point you want to delete.	string	Mandatory
recursive	Flag to delete sub path's (default is false). When you try to delete a data point with existing sub path's and "recursive" is false an error with "message" "Path is not empty" will be returned.	boolean	Optional
tag	Any data that will be echoed in the response.	*any*	Optional

3.5.3 Response Fields

The "delete" array of objects:

Field	Description	Type	Occurrence
code	The code can be: - "ok": On success. - "no perm": You are not allowed to delete this data point. - "not found": This data point doesn't exist on the system. - "error": Something went wrong while deleting.	string	Always
path	The path of the deleted data point.	string	When no fatal error
message	This field contains an human readable error message in English.	string	Other than code "ok"
tag	Tag data from request.	*from request*	When existing in request

3.5.4 JSON Schema

Request:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Delete Request",
  "description": "Deleting one or more data points",
  "type": "object",
  "properties": {
    "whois": {
      "description": "Identification of the sender",
      "type": "string"
    },
    "delete": {
      "description": "The command",
      "type": "array",
      "items": {
        "title": "Data point delete definition",
        "type": "object",
        "properties": {
          "path": {
            "description": "The DMS path to the data point",
            "type": "string"
          },
          "recursive": {
            "description": "Flag to delete sub path's",
            "type": "boolean"
          },
          "tag": {
            "description": "Any data, will be echoed on the response",
            "type": [ "object", "array", "number", "string", "boolean"]
          }
        },
        "additionalProperties": false,
        "required": ["path"]
      }
    },
    "minItems": 1
  },
  "required": ["whois", "delete"]
}
```

Response::

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Delete Response",
  "description": "Information about deleting one or more data points",
  "type": "object",
  "properties": {
    "delete": {
      "description": "The command",
      "type": "array",
      "items": {
        "title": "Data point value",
        "type": "object",
        "properties": {
          "code": {
            "description": "The result code",
            "type": "string",
            "enum": [ "ok", "no perm", "not found", "error" ]
          },
          "path": {
            "description": "The DMS path to the data point",
            "type": "string"
          },
          "message": {
            "description": "Human readable error message",
            "type": "string"
          },
          "tag": {
            "description": "Echo from the request",
            "type": [ "object", "array", "number", "string", "boolean"]
          }
        },
        "required": ["code"]
      }
    },
    "minItems": 1
  },
  "required": ["delete"]
}
```

3.6 Monitor

Monitoring data points is only possible on WebSocket connections!

After loss of a WebSocket connection, all monitoring configurations for the prior connection are cleared, the client is responsible to reconfigure all data point monitoring configurations.

3.6.1 Example

Request:

```
{
  "subscribe": [
    {
      "path":"System",
      "event":"onChange",
      "query": {
        "maxDepth": 0
      }
    }
  ]
}
```

Response:

```
{
  "subscribe": [
    {
      "code":"ok",
      "path":"System",
      "type":"none",
      "value":null,
      "stamp":null,
      "query": {
        "maxDepth": 0
      }
    }
  ]
}
```

Event Message:

```
{
  "event": [
    {
      "code":"onChange",
      "path":"System:Blinker:Blink0.25",
      "trigger":"<SYS>",
      "type":"bool",
      "value":true,
      "stamp":"2015-05-27T08:13:58,521+02:00"
    },
    {
      "code":"onChange",
      "path":"System:Blinker:Blink0.5",
      "trigger":"<SYS>",
      "type":"bool",
      "value":false,
      "stamp":"2015-05-27T08:13:58,521+02:00"
    }
  ]
}
```

3.6.2 Request Fields

The "subscribe" array of objects:

Field	Description	Type	M/O
path	The path of the data point you want to monitor.	string	Mandatory
query	The query object (see here), default is no query. The query content will be analyzed at the time of the event.	object	Optional
event	Requested event to monitor, can be one or more of "onChange", "onSet", "onCreate", "onRename", "onDelete" or "*". Default is "onChange". Multiple requested events are separated by ',', e.g. "onChange, onRename,onDelete". "*" means all events.	string	Optional
tag	Any data that will be echoed in the response and the event.	*any*	Optional

Note:
Multiple subscriptions can be set on the same path with different tag's.

Any subscription on the same path with the same tag (on the same connection) as in a previous subscription will overwrite the previous configuration.

To unsubscribe a monitored data point send "unsubscribe" command with the path and tag used on subscription.

The "unsubscribe" array of objects:

Field	Description	Type	M/O
path	The path of the data point you don't want to monitor anymore.	string	Mandatory
tag	The tag data used on subscription for this data point.	*any*	Optional

3.6.3 Response Fields

The "subscribe" and "unsubscribe" array of objects:

Field	Description	Type	Occurrence
code	The code can be: - "ok": On success. - "no perm": You are not allowed to monitor this data point. - "not found": This data point doesn't exist on the system. - "error": Something went wrong.	string	Always
path	The path of the monitored data point.	string	When no fatal error
query	Echo of any query object from the request.	object	When existing in request
value	The current value.	number, boolean, string, null	On code "ok"
type	The type of the value, can be "int", "double" (a floating point number), "string", "bool" or "none" for nodes without value.	string	On code "ok"
stamp	ISO 8601 formatted (see here).	string, null	On code "ok"
message	This field contains an human readable error message in English.	string	Other than code "ok"
tag	Tag data from request.	*from request*	When existing in request

Note:
Take care about the fact, that any (previous configured) "event" message can occur asynchron between the subscribe-request and the subscribe-response!

3.6.4 Event Message - Fields

Any triggered event will be transmitted with a "event" object with one ore more array entries of objects with the following content:

Field	Description	Type	Occurrence
code	The code shows the initiating event trigger ("onChange", "onSet", "onCreate", "onRename", "onDelete")	string	Always
path	The path of the monitored data point.	string	Always
newPath	The renamed path.	string	On code "onRename"
trigger	The trigger of the event. For JSON connections this is the "whois" field from the request (set/rename/delete). For ProMoS 1/2 connections (Pipe/TCP), this is the application name and the connection name (when present) from the corresponding client application (e.g. "GE@PC-WS096"). Other triggers can be: "<SYS>" (internal system operations, e.g. on "System:Time") or "<DMS>" for manipulated data points on the DMS GUI or for data points modified by PLS function.	string	Always
value	The value: - after "onChange", "onSet", "onCreate", "onRename" - before "onDelete"	number, boolean, string, null	Always
type	The type of the value, can be "int", "double" (a floating point number), "string", "bool" or "none" for nodes without value.	string	Always
stamp	Time stamp of the event, ISO 8601 formatted (see here).	string, null	Always
tag	Tag data from the "subscribe" request.	*from request*	When existing in request

Note:
When a data point is renamed, the events "onRename", "onDelete" and "onCreate" will be generated.

3.6.5 JSON Schema

Request:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Monitor Request",
  "description": "Monitoring one or more data points",
  "type": "object",
  "properties": {
    "subscribe": {
      "description": "The command",
      "type": "array",
      "items": {
        "title": "Data point monitoring definition",
        "type": "object",
        "properties": {
          "path": {
            "description": "The DMS path to the data point",
            "type": "string"
          },
          "query": {
            "description": "Optional query parameters",
            "type": "object",
            "properties": {
              "regExPath": {
                "description": "RegEx pattern for the path",
                "type": "string"
              },
              "regExValue": {
                "description": "RegEx pattern for the value",
                "type": "string"
              },
              "regExStamp": {
                "description": "RegEx pattern for the time stamp",
                "type": "string"
              },
              "isType": {
                "description": "Type filters",
                "type": "string",
                "enum": [ "int", "double", "string", "bool", "none" ]
              },
              "hasHistData": {
                "description": "Filter for data points with historical data",
                "type": "boolean"
              },
              "maxDepth": {
                "description": "Maximal depth for searching in sub path's",
                "type": "number"
              }
            }
          },
          "event": {
            "description": "The requested event (or combinations)",
            "type": "string",
            "enum": [ "onChange", "onSet", "onCreate", "onRename", "onDelete", "*" ]
          },
          "tag": {
            "description": "Any data, will be echoed on the subscribe response and
on the event",
            "type": [ "object", "array", "number", "string", "boolean"]
          }
        },
        "additionalProperties": false,
        "required": ["path"]
      }
    },
    "minItems": 1
  },
  "required": ["subscribe"]
}
```

Response::

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Monitor Response",
  "description": "Information about subscribe one or more data points",
  "type": "object",
  "properties": {
    "subscribe": {
      "description": "The command",
      "type": "array",
      "items": {
        "title": "Data point value",
        "type": "object",
        "properties": {
          "code": {
            "description": "The result code",
            "type": "string",
            "enum": [ "ok", "no perm", "not found", "error" ]
          },
          "path": {
            "description": "The DMS path to the data point",
            "type": "string"
          },
          "value": {
            "description": "The value of the data point",
            "type": ["number", "string", "boolean", "null"]
          },
          "type": {
            "description": "The value type",
            "type": "string",
            "enum": [ "int", "double", "string", "bool", "none" ]
          },
          "stamp": {
            "description": "The timestamp of the last change of the value, ISO 
8601",
            "type": ["string", "null"]
          },
          "message": {
            "description": "Human readable error message",
            "type": "string"
          },
          "tag": {
            "description": "Echo from the request",
            "type": [ "object", "array", "number", "string", "boolean"]
          }
        },
        "required": ["code"]
      }
    },
    "minItems": 1
  },
  "required": ["subscribe"]
}
```

Event Message:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Event Message",
  "description": "Information about monitor one or more data points",
  "type": "object",
  "properties": {
    "event": {
      "description": "The command",
      "type": "array",
      "items": {
        "title": "Data point value",
        "type": "object",
        "properties": {
          "code": {
            "description": "The event",
            "type": "string",
            "enum": [ "onChange", "onSet", "onCreate", "onRename", "onDelete" ]
          },
          "path": {
            "description": "The DMS path to the data point",
            "type": "string"
          },
          "trigger": {
            "description": "The trigger of this event",
            "type": "string"
          },
          "value": {
            "description": "The value of the data point",
            "type": ["number", "string", "boolean", "null"]
          },
          "type": {
            "description": "The value type",
            "type": "string",
            "enum": [ "int", "double", "string", "bool", "none" ]
          },
          "stamp": {
            "description": "The timestamp of the last change of the value, ISO
8601",
            "type": ["string", "null"]
          },
          "message": {
            "description": "Human readable error message",
            "type": "string"
          },
          "tag": {
            "description": "The echoed tag from subscribe request",
            "type": [ "object", "array", "number", "string", "boolean"]
          }
        },
        "required": ["code"]
      }
    },
    "minItems": 1
  },
  "required": ["event"]
}
```

4 Change Log Groups

This chapter describes the commands to handle change logs for data points.
Any data point can have a change log, see also "hasChangelog" from the data point query

object.

Change logs are grouped by a name, to get all available groups of change logs you can ask them by "changelogGetGroups".

4.1 Command Overview

Command	Description
changelogGetGroups	Get all available change log groups.
changelogRead	Read content of specific change log group.

4.2 ChangelogGetGroups

4.2.1 Example

Request:

```
{
  "changelogGetGroups": []
}
```

Response:

```
{
  "changelogGetGroups": [
    {
      "code":"ok",
      "groups": [
        "Login",
        "Ereign1",
        "Alarm",
        "Manip1"
      ]
    },
  ]
}
```

4.2.2 Request Fields

The "changelogGetGroups" array:

Empty array.

4.2.3 Response Fields

The "changelogGetGroups" array of objects:

Field	Description	Type	Occurrence
code	The code can be: - "ok": On success.	string	Always
groups	All group names available for the changelogRead command.	array of string	When no fatal error

4.2.4 JSON Schema

Request:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Changelog Group Request",
  "description": "Get all available change logs",
  "type": "object",
  "properties": {
    "changelogGetGroups": {
      "description": "The command",
      "type": "array"
    },
    "minItems": 1
  },
  "required": ["changelogGetGroups"]
}
```

Response::

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Changelog Group Response",
  "description": "Information about all available change logs",
  "type": "object",
  "properties": {
    "changelogGetGroups": {
      "description": "The command",
      "type": "array",
      "items": {
        "title": "Response object",
        "type": "object",
        "properties": {
          "code": {
            "description": "The result code",
            "type": "string",
            "enum": [ "ok" ]
          },
          "groups": {
            "description": "All available group names",
            "type": "array"
            "items": {
              "title": "Group name",
              "type": "string"
            }
          }
        },
        "required": ["code"]
      }
    },
    "minItems": 1
  },
  "required": ["changelogGetGroups"]
}
```

4.3 ChangelogRead

4.3.1 Example

Request:

```
{
  "changelogRead": [
    {
        "group": "ABS1",
        "start": "2016-02-29T00:00:00Z",
        "end": "2016-03-02T00:00:00Z"
    }
  ]
}
```

Response:

```
{
  "changelogRead": [
    {
      "code": "ok",
      "group": "ABS1",
      "changelog": [
      {
        "path": "BN028:H04:VS:001:ABS_Ein",
        "stamp": "2016-02-29T11:48:55,715+01:00",
        "text": "Heizungsventil Aus"
      },
      {
        "path": "BN028:H04:VS:001:ABS_Ein",
        "stamp": "2016-02-29T11:42:58,197+01:00",
        "text": "Heizungsventil Ein"
      }]
    }
  ]
}
```

4.3.2 Request Fields

The "changelogRead" array of objects:

Field	Description	Type	M/O
group	The group name of the change log you want to read.	string	Mandatory
start	The requested start time stamp for the data (see also here).	string	Mandatory
end	The requested end time stamp for the data (see also here). Default is the current time stamp.	string	Optional
tag	Any data that will be echoed in the response.	*any*	Optional

4.3.3 Response Fields

The "changelogRead" array of objects:

Field	Description	Type	Occurrence
code	The code can be: - "ok": On success.	string	Always

group	Echo from the request	array of string	When no fatal error
changelog	An array with change log records (see below).	array	On code "ok"
message	This field contains an human readable error message in English.	object	On code "ok", when showExtInfos was requested
tag	Tag data from request.	*from request*	When existing in request

The "changelog" array of objects:
See here

4.3.4 JSON Schema

Request:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Change Log Read Request",
  "description": "Reading one or more change logs by group",
  "type": "object",
  "properties": {
    "changelogRead": {
      "description": "The command",
      "type": "array",
      "items": {
        "title": "Read definition",
        "type": "object",
        "properties": {
          "group": {
            "description": "The group name",
            "type": "string"
          },
          "start": {
            "description": "Time stamp for start",
            "type": "string"
          },
          "end": {
            "description": "Time stamp for end",
            "type": "string"
          },
          "tag": {
            "description": "Any data, will be echoed on the response",
            "type": [ "object", "array", "number", "string", "boolean"]
          }
        },
        "additionalProperties": false,
        "required": ["group", "start"]
      },
      "minItems": 1
    }
  },
  "required": ["changelogRead"]
}
```

Response:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Read Response",
  "description": "Information about one or more change logs",
  "type": "object",
  "properties": {
    "changelogRead": {
      "description": "The command",
      "type": "array",
      "items": {
        "title": "Change log details",
        "type": "object",
        "properties": {
          "code": {
            "description": "The result code",
            "type": "string",
            "enum": [ "ok", "error" ]
          },
          "group": {
            "description": "The group, echoed from request",
            "type": "string"
          },
          "message": {
            "description": "Human readable error message",
            "type": "string"
          },
          "changelog": {
            "description": "Array of the requested change log",
            "type": "array",
            "items": {
              "title": "Change log",
              "type": "object",
              "properties": {
                "path": {
                  "description": "The path of the corresponding data point",
                  "type": ["string"]
                },
                "stamp": {
                  "description": "The timestamp of the recorded log, ISO 8601",
                  "type": ["string"]
                },
                "text": {
                  "description": "The text",
                  "type": ["string"]
                }
              }
            }
          },
          "tag": {
            "description": "Echo from the request",
            "type": [ "object", "array", "number", "string", "boolean"]
          }
        },
        "required": ["code"]
      }
    },
    "minItems": 1
  },
  "required": ["changelogRead"]
}
```