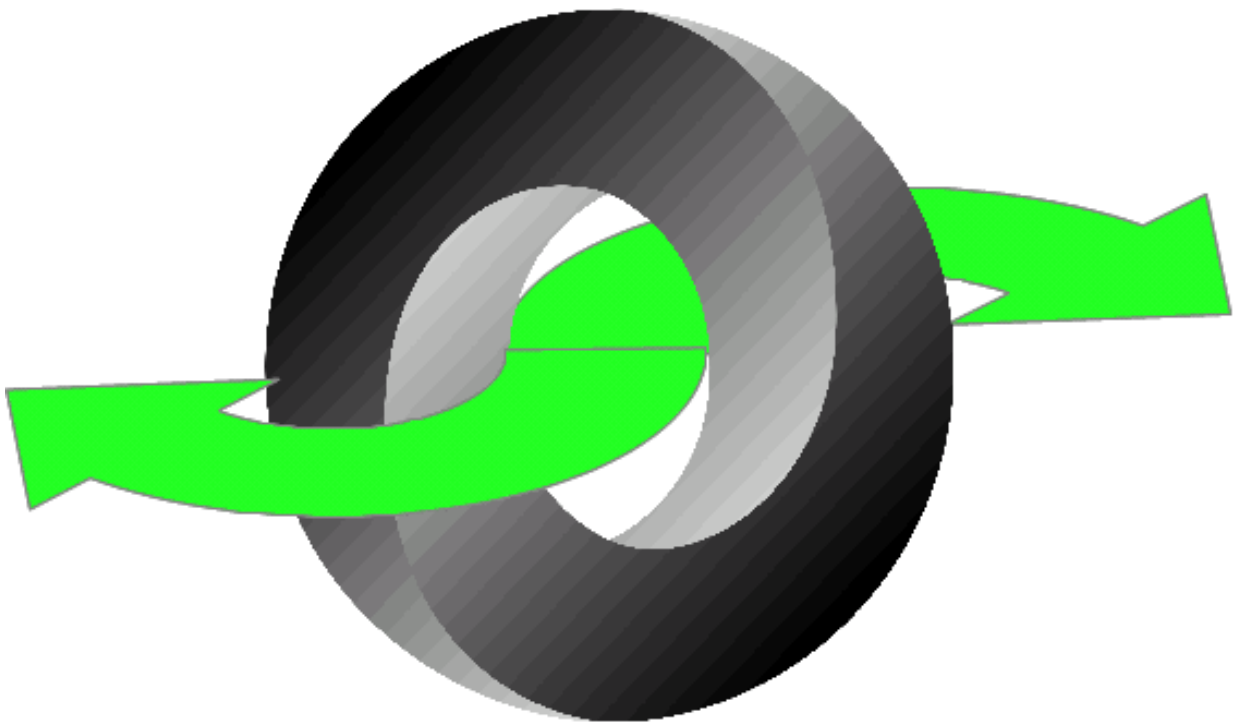


# ProMoS JSON Data Exchange over EDL-Portal

© 2021 MST Systemtechnik AG, Belp



# ProMoS JSON Data Exchange over EDL-Portal

© 2021 MST Systemtechnik AG, Belp

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Printed: November 2021 in Belp, Switzerland

## **Publisher**

*MST Systemtechnik AG*

## **Managing Editor**

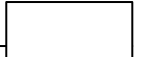
...

## **Technical Editors**

*Heiko Henning, Martin Frei*

## **Team Coordinator**

*Christoph Müller*

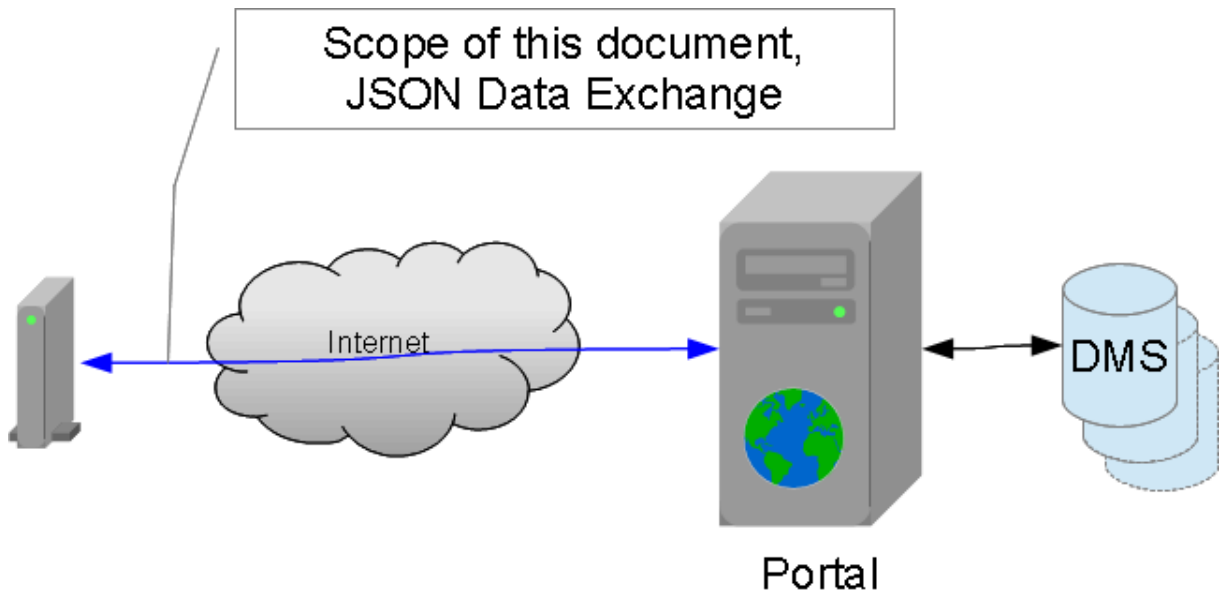


# Table of contents

<b>Kapitel 1 Introduction</b>	<b>1</b>
1.1 History.....	1
<b>Kapitel 2 Configuration</b>	<b>1</b>
<b>Kapitel 3 Data Exchange</b>	<b>1</b>
3.1 Used Technologies.....	1
3.2 Authentication.....	2
3.3 Error Messages.....	2
3.3.1 Example .....	2
3.4 Read Data.....	3
3.4.1 Example .....	3
3.4.2 Request Fields .....	5
3.4.3 Response Fields .....	5
3.4.4 JSON schema .....	6
3.5 Write Data.....	8
3.5.1 Example .....	8
3.5.2 Request Fields .....	9
3.5.3 Response Fields .....	10
3.5.4 JSON schema .....	11
3.6 Write Historical Data.....	13
3.6.1 Example .....	13
3.6.2 Request Fields .....	14
3.6.3 Response Fields .....	15

# 1 Introduction

This document describes the data exchange between an external device and the ProMoS Data Management System (DMS) over the MST EDL-Portal.



## 1.1 History

Version	Who	Date	Remark
1.0	mst_frem	26.02.2015	Draft
1.1	mst_henh	28.02.2015	API description
1.2	mst_frem	03.03.2015	Add code "error" on read data response
1.3	mst_frem	20.03.2015	Fix JSON schma (missed root object)
1.4	mst_frem	10.09.2021	Add chapter for historical data (write)

## 2 Configuration

The configuration can be done on the Portal (Web-GUI), with 2 lists (read / read/write data points).

Only allowed users can change the configuration.

## 3 Data Exchange

### 3.1 Used Technologies

The whole API is HTTP POST based, see [http://en.wikipedia.org/wiki/POST\\_\(HTTP\)](http://en.wikipedia.org/wiki/POST_(HTTP))

For the post body JSON will be used.

We use JSON body for request (instead of the standard URL encoded request) to have the same encoding for request and response.

<http://en.wikipedia.org/wiki/JSON>

## 3.2 Authentication

The authentication is Pre Shared Key based.  
Each pushing component needs to be registered in the portal configuration.  
The portal will generate a 32 character alpha numeric key.

Additionally the access can be limited by client ip(s).

## 3.3 Error Messages

You will get an HTTP status code 401, with Content-type: text/plain and a plain text error message in body in case of a invalid auth key or invalid client ip.

If the authentication was successful, you will get a http status code 200.

### 3.3.1 Example

```
HTTP/1.1 401 OK
Date: Fri, 27 Feb 2015 13:37:08 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Transfer-Encoding: chunked
Content-Type: text/plain

You are not permitted to send data.
```

## 3.4 Read Data

You can request as much data points as you want with 1 single request.  
But it's recommend to not request more then 10'000 at once, because the API needs ~2sec per 1000 data points.

The system will check if the portal owner permitted you read access for this data point(s) and deliver the current value(s) of this data point(s).  
The sequence of the data points in the response may not have the same order as in the request.

### 3.4.1 Example

#### Request:

```
POST /component/control-system-get-value/api_key/F2E1101D72EA1A3327136EB6AFEB4C81
HTTP/1.1
Host: example.edl.ch
Connection: keep-alive
Content-Type: application/json
Content-Length: 38299

{
  "get": [
    {"path": "EXAMPLE001:T11:MN:003:Vis:VMC_energy1"},
    {"path": "EXAMPLE001:T11:MN:003:Vis:VEnergy1V"},
    {"path": "EXAMPLE001:T11:MN:003:Vis:VMC_power"},
    {"path": "EXAMPLE001:not:existing"}
  ]
}
```

**Response::**

```
HTTP/1.1 200 OK
Date: Fri, 27 Feb 2015 13:37:08 GMT
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: application/json; charset=utf-8
```

```
{
  "get": [
    {
      "code": "ok",
      "path": "EXAMPLE001:T11:MN:003:Vis:VMC_energy1",
      "value": 3.165,
      "type": "double",
      "stamp": "2015-02-27T08:17:51"
    },
    {
      "code": "ok",
      "path": "EXAMPLE001:T11:MN:003:Vis:VEnergy1V",
      "value": 0.14,
      "type": "double",
      "stamp": "2015-02-27T08:17:51"
    },
    {
      "code": "no perm",
      "path": "EXAMPLE001:T11:MN:003:Vis:VEnergy1V",
    },
    {
      "code": "not found",
      "path": "EXAMPLE001:not:existing",
      "message": "Data point doesn't exist"
    }
  ]
}
```

**Response in case of fatal error:**

```
HTTP/1.1 200 OK
Date: Fri, 27 Feb 2015 13:37:08 GMT
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: application/json; charset=utf-8
```

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

```
HTTP/1.1 200 OK
Date: Fri, 27 Feb 2015 13:37:08 GMT
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: application/json; charset=utf-8

{
  "get": [
    {
      "code": "error",
      "message": "Missing \"path\" in get[33]"
    }
  ]
}
```

### 3.4.2 Request Fields

- **"path"**  
The path of the data point you want to read.  
A required string field.

### 3.4.3 Response Fields

- **"code"**  
The code can be:
  - "ok": On success.
  - "no perm": The portal owner has not permitted you to read this data point.
  - "not found": You are permitted for this data point, but it doesn't exist on the system.
  - "error": In case of a fatal error.The field always appears.
- **"path"**  
The path of the requested data point.  
It appears as long as there is no fatal error.
- **"value"**  
The value contains the current value of the data point.  
It appears only for code "ok".
- **"type"**  
This field is the data type of this data point.  
The type can be "int", "double" (a floating point number), "string" or "bool".  
It appears only for code "ok".



- **"stamp"**

This field can be NULL.

If set, it contains a string with the time stamp of the last change of this data point.

The date is ISO 8601 formatted. but without time zone indication, Because the API just forwards the time stamp just from the control system.

If the control system is a ProMoS 1.x the time is the local time zone of the object, mostly Europe/Zurich.

It appears only for code "ok".

- **"message"**

This field contains an human readable error message in English.

It appears for code other than "ok".

### 3.4.4 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"
          }
        }
      },
      "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" ]
          },
          "type": {
            "description": "The value type",
            "type": "string",
            "enum": [ "int", "double", "string", "bool" ]
          },
          "stamp": {
            "description": "The timestamp of the last change of the value, ISO
8601",
            "type": [ "string", "null" ]
          },
          "message": {
            "description": "Human readable error message",
            "type": "string"
          }
        },
        "required": [ "code" ]
      }
    },
    "minItems": 1
  },
  "required": [ "get" ]
}

```

## 3.5 Write Data

You can write as much data points as you want with 1 single request.

But it's recommend to not write more then 10'000 at once, because the API needs ~5sec per 1000 data points.

The system will check if the portal owner permitted you write access for this data point(s). The sequence of the data points in the response may not have the same order as in the request.

### 3.5.1 Example

#### Request:

```
POST /component/controll-system-set-value/api_key/F2E1101D72EA1A3327136EB6AFEB4C81
HTTP/1.1
Host: example.edl.ch
Connection: keep-alive
Content-Type: application/json
Content-Length: 38299

{
  "set": [
    {
      "path": "EXAMPLE001:T11:MN:003:Vis:VMC_energy1",
      "value": 4.4565467567867,
      "type": "double"
    },
    {
      "path": "EXAMPLE001:T11:MN:003:Vis:VEnergy1V",
      "value": -1,
      "type": "double"
    },
    {
      "path": "EXAMPLE001:TEST:BOOLEAN",
      "value": true,
      "type": "bool"
    },
    {
      "path": "EXAMPLE001:TEST:INT",
      "value": 44,
      "type": "int"
    },
    {
      "path": "EXAMPLE001:TEST:STRING",
      "value": "some long example message",
      "type": "string"
    }
  ]
}
```

**Response::**

```
HTTP/1.1 200 OK
Date: Fri, 27 Feb 2015 13:37:08 GMT
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: application/json; charset=utf-8

{
  "set": [
    {
      "code": "ok",
      "path": "EXAMPLE001:T11:MN:003:Vis:VMC_energyl",
      "value": 3,
      "type": "double"
    },
    {
      "code": "ok",
      "path": "EXAMPLE001:T11:MN:003:Vis:VEnergylV",
      "value": 0,
      "type": "double"
    },
    {
      "code": "no perm",
      "path": "EXAMPLE001:TEST:BOOLEAN",
    },
    {
      "code": "error",
      "path": "EXAMPLE001:TEST:INT",
      "message": "Data point doesn't exist"
    },
    {
      "code": "error",
      "path": "EXAMPLE001:TEST:STRING",
      "message": "Data type doesn't match"
    }
  ]
}
```

### 3.5.2 Request Fields

- **"path"**  
The path of the data point to write.  
A required string field.
- **"value"**  
The value to write.  
It is recommended to use real json data types, because for bool fields - a string "FALSE" will be TRUE!  
A required mixed value field.

- **"type"**  
The type of the value.  
The API will check if the value type on control system if it matches.  
The type can be "int", "double" (a floating point number), "string", "bool".  
A required string field.

### 3.5.3 Response Fields

- **"code"**  
The code can be:
  - "ok": On success.
  - "no perm": The portal owner has not permitted you to write this data point.
  - "not found": You are permitted for this data point, but it doesn't exist on the system.
  - "error": Something went wrong while writing to the control system.The field always appears.
- **"path"**  
The path of the written data point.  
It appears as long as there is no fatal error.
- **"value"**  
The value that you have set.  
It appears only for code "ok".
- **"type"**  
This field is the data type of this data point.  
You can use it if you want double check it.  
It appears only for code "ok"
- **"message"**  
This field contains an human readable error message in English.  
It appears for code other than "ok".

### 3.5.4 JSON schema

#### Request:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Write Request",
  "description": "Writing one or more data points",
  "type": "object",
  "properties": {
    "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"
          },
          "value": {
            "description": "The new value of the data point",
            "type": ["number", "string", "boolean"]
          },
          "type": {
            "description": "The value type",
            "type": "string",
            "enum": [ "int", "double", "string", "bool" ]
          }
        },
        "additionalProperties": false,
        "required": ["path", "value", "type"]
      }
    },
    "minItems": 1
  },
  "required": ["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" ]
          },
          "type": {
            "description": "The value type",
            "type": "string",
            "enum": [ "int", "double", "string", "bool" ]
          },
          "stamp": {
            "description": "The timestamp of the last change of the value, ISO
8601",
            "type": [ "string", "null" ]
          },
          "message": {
            "description": "Human readable error message",
            "type": "string"
          }
        },
        "required": [ "code" ]
      }
    },
    "minItems": 1
  },
  "required": [ "set" ]
}
```

## 3.6 Write Historical Data

It's recommend to not write more then 10`000 values at once.

The system will check if the portal owner permitted you write access for this data point(s).

### 3.6.1 Example

#### Request:

```
POST /component/controll-system-set-value/api_key/F2E1101D72EA1A3327136EB6AFEB4C81
HTTP/1.1
Host: example.edl.ch
Connection: keep-alive
Content-Type: application/json
Content-Length: 38299

{
  "set": [
    {
      "path": "EXAMPLE001:T11:MN:003:Vis:VMC_energy1",
      "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
        },
        {
          "stamp": "2015-04-03T04:37:00,000+02:00",
          "value": 0.5,
          "state": "inv"
        },
        {
          "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
        }
      ]
    }
  ]
}
```



**Response::**

```
HTTP/1.1 200 OK
Date: Fri, 27 Feb 2015 13:37:08 GMT
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: application/json; charset=utf-8

{
  "set": [
    {
      "code": "ok",
      "path": "EXMPL1:T11:MN:003:Vis:VMC_energy1"
    }
  ]
}
```

### 3.6.2 Request Fields

- **"path"**  
The path of the data point to write.  
A required string field.
- **"histData"** (format: detailed) - array of objects
  - **"stamp"**  
ISO 8601 formatted timestamp (UTC).  
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'h'h': 'mm': 'ss', 'fff'Z'  
e.g.: "2015-03-20T07:49:19,000Z" or "2015-03-20T08:49:19,000+01:00"
  - **"value"**  
The value to write.
  - **"state"**  
State of this value, can be "ok", "comErr" (recorded during communication error) or "inv" (any other error situation).
- **"histData"** (format: compact) - array of numeric objects - timestamp and value  
e.g. { "2015-04-03T04:33:20,000+02:00": 0.95 }

### 3.6.3 Response Fields

- **"code"**

The code can be:

- "ok": On success.
- "no perm": The portal owner has not permitted you to write this data point.
- "not found": You are permitted for this data point, but it doesn't exist on the system.
- "error": Something went wrong while writing to the control system.

The field always appears.

- **"path"**

The path of the written data point.

It appears as long as there is no fatal error.

- **"message"**

This field contains an human readable error message in English.

It appears for code other than "ok".