IO-Link

JSON

Integration for IO-Link

Draft 0.99
July 2019

Order No: 10.222

IO-Link

File name: JSON_Integration_10222_d099_Jul19

This specification has been prepared by the IO-Link community.

Any comments, proposals, requests on this document are appreciated through the IO-Link CR database www.io-link-projects.com. Please provide name and email address.

Login: IOL-JSON

Password: Report

Important notes:

NOTE 1 The IO-Link Community Rules shall be considered prior to the development and marketing of IO-Link products. The document can be downloaded from the www.io-link.com portal.

Publisher:
IO-Link Community
c/o PROFIBUS Nutzorganisation e.V.
Haid-und-Neu-Str. 7
76131 Karlsruhe
Germany
Phone: +49 721 / 96 58 590
Fax: +49 721 / 96 58 589
E-mail: info@io-link.com
Web site: www.io-link.com

CONTENTS

0 Introduction

0.1 General

The base technology of IO-Link™[1] is subject matter of the international standard IEC 61131-9 (www.iec.ch). IEC 61131-9 is part of a series of standards on programmable controllers and the associated peripherals and should be read in conjunction with other parts of the series.

0.2 Patent declaration

The IO-Link Community draws attention to the fact that it is claimed that compliance with this document may involve the use of patents concerning the point-to-point serial communication interface for small sensors and actuators as follows, where the [xx] notation indicates the holder of the patent right:

Patent number	[xx]	Title

IO-Link Community takes no position concerning the evidence, validity and scope of these patent rights.

The holders of these patents rights have assured the IO-Link Community that they are willing to negotiate licences either free of charge or under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patent rights are registered with the IO-Link Community.

Information may be obtained from:

[xx]	Name and address of patent holder

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. The IO-Link Community shall not be held responsible for identifying any or all such patent rights.

The IO-Link Community maintains on-line data bases of patents relevant to their standards. Users are encouraged to consult the databases for the most up to date information concerning patents.

[1] IO-Link™ is a trade name of the "IO-Link Community". This information is given for the convenience of users of this specification and does not constitute an endorsement by the "IO-Link Community" of the trade name holder or any of its products. Compliance to this standard does not require use of the registered logos for IO-LinkTM. Use of the registered logos for IO-Link™ requires permission of the "IO-Link Community".

IO-Link JSON Mapping

1 Motivation and scope

New use cases and requirements concerning the integration between modern IT systems and the production floor require new device interfaces. The connections today mainly focus on the integration of a device into fieldbuses and PLC systems. Cyclic data exchange and real time are the most important requirements for today field bus implementations. The techniques used are completely different than the ones used for the rest of the IT world. On the other hand, modern automation devices provide a way to communicate over TCP/IP networks beside the real time communication with the PLC over the field bus.

This document describes a device data model, objects and semantics for mapping on IT relevant connections or services.

This document describes a REST API

a) for data access to IO-Link masters, ports and devices and the gateway.

b) for IODD file management (up/download).

c) for MQTT client configuration.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IO-Link Community, *IO-Link System Description – Technology and Application, Version February 2016, Order No. 4.392 (available at http://www.io-link.com)*

IO-Link Community, *IO-Link Interface and System Specification, Version 1.1.3, Order No. 10.002 (available at http://www.io-link.com)*

IO-Link Community, *IO Device Description (IODD), Version 1.1, Order No. 10.012*

3 Terms, definitions, symbols, abbreviated terms and conventions

3.1 Common terms and definitions

For the purposes of this document, the terms and definitions given in IEC 61131-1 and IEC 61131-2, as well as the following apply.

Master
Active peer connected through ports to one up to n devices providing an interface to the gateway to the upper level communication systems (e.g. PLCs or edge gateways)

Note: Uppercase "Master" is used for SDCI equipment, while lowercase "master" is used in a generic manner.

Port
Communication interface of the Master to one Device

IODD
The XML based IO Device Description of an IO-Link device see [2].

PIN 2
The physical digital I/O interface of an IO-Link port usually used with M12 and M8 connectors

PIN 4
The physical digital I/O interface of an IO-Link port usually used with M12 and M8 connectors IO-Link communication runs physically over PIN 4 and is called C/Q.

SIO
Standard Input Output mode. This could be a digital input or a digital output.

masterNumber
This is the number of a specific master within the gateway

portNumber
This is the number of a specific port within the Master

URL
This is an Uniform Resource Locator

3.2 Symbols and abbreviated terms

C/Q	connection for communication (C) or switching (Q) signal (SIO)
DI	digital input (data coming from a Device to a Master)
DO	digital output (data going from a Master to a Device)
M/O/C	Mandatory, optional, conditional see 3.3.2

3.3 Conventions

3.3.1 Placeholders

Strings that are embraced by {} are placeholders for variables.

Note: This convention is not applicable for JSON objects.

3.3.2 Meaning of M/O/C

It is mandatory to implement the handling of objects and key value pairs that are marked with “M”.When sending a HTTP Post request a client is not obligated to provide all objects or key value pairs. The resource has to be updated by merging the newly received data with the existing data.

4 Architectural and technical scope

4.1 General objectives

As summarized in the IO-Link System Description, an IO-Link system consists of an IO-Link master, IO-Link devices and cables connecting the IO-Link devices to the IO-Link Master

A physical IO-Link Gateway consists of one or more Masters containing one or more ports. On each port an IO-Link device may be connected. The physical IO-Link Gateway may also have one or more Gateway applications (e.g. Webserver, OPC UA server or MQTT client).

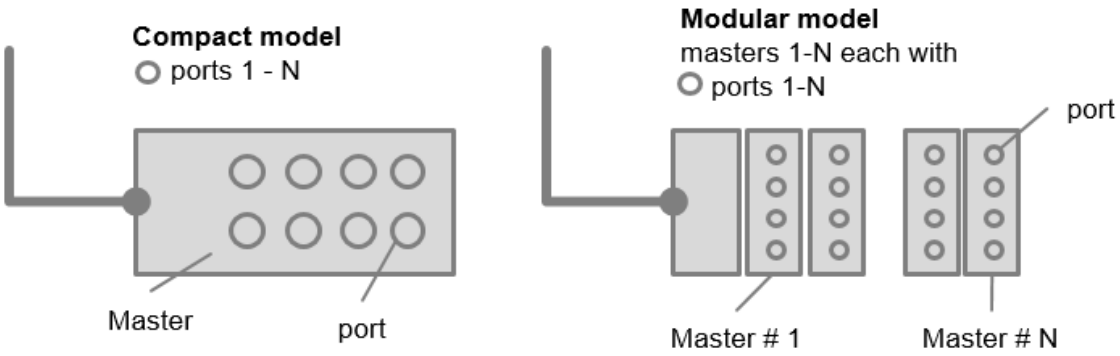

Figure 1 – Physical Master models

4.2 Security

For data security it is recommended to use TLS-PSK with AES for the transport layer security.

4.3 Device data and layer model

This device layer model (including gateway, IO-Link master and devices) is used to structure the REST API described in this specification.

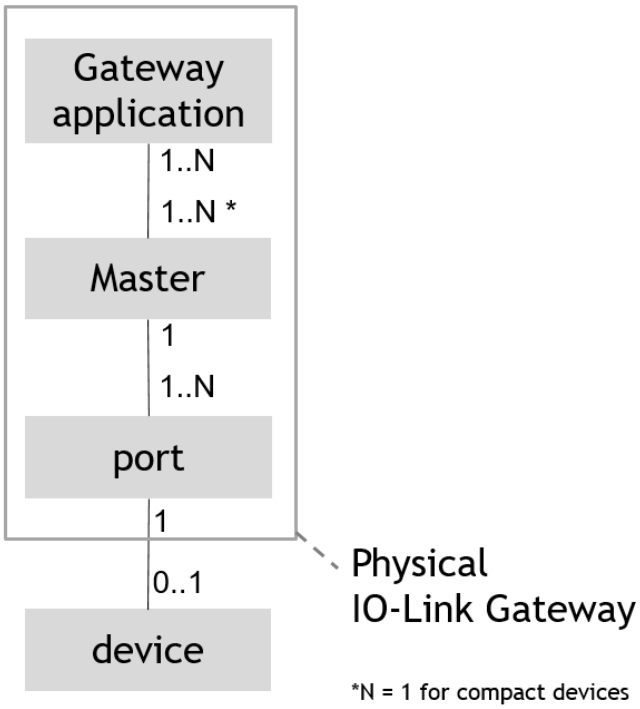

Figure 2 – Device layer model

This layer model comprises compact modules containing one Master as well as modular devices with N Masters.

4.4 General rules

4.4.1 Master numbering

Addressing of IO-Link masters within a gateway starts with number 1 for the first master.

4.4.2 Port numbering

Addressing of IO-Link ports within a master starts with number 1 for the first port.

4.4.3 Device Naming

Devices are accessed by names. Following rules for the device naming apply:

1. The default device name is created based on Master number and Port number in the format `master{masterNumber}port{portNumber}.`
2. The default device name can be changed via port configuration (see 5.6.5.).
3. The definition of duplicated device names has to be rejected.
4. A device name must only contain alphanumeric characters and underscores.
5. A device names maximum length is 32 characters minimum length is 1 character.

Examples:

```
”master2port4”  is the default name of an device connected to port 4 of master 2
”exampleSensor” this is a new assigned name to a device
```

4.4.4 Access rights

All requests for writing data may be blocked due to right restrictions. Also resetting or rebooting may be not allowed due to right restrictions. Requests for writing data without having permission to do so, will respond with an error. The data will remain unchanged.

It is recommended to use access restrictions to handle the access of multiple clients to the same resource.

4.4.5 Naming based on IODD

It is optionally also possible to access IO-Link data (comprizing IO-Link process data and IO-Link parameters) by name rather than by index and subindex.

Constraints for strings in keys of JSON key-value-pairs (see [3]) and URLs require the following conversion rules for those names:

Rule 1: Names are based on the IODD XML Element `Name` inside variables or process data resolving the text in primary language

Rule 2: Special characters get replaced by „_“

Rule 3: Leading numbers shall be prefixed with „_“

Rule 4: If there are duplicated IO-Link names, the IO-Link index or subindex has to be accessed by appending the index number or subindex number behind the name according to the following scheme: `{name}_{index}` or `{name}_{subindex}`.

Rule 5: Naming according the scheme `{name}_{index}` or `{name}_{subindex}` is always allowed even if names are not duplicated.

Examples:

Table 1 – IODD text conversion rules examples

IODD text	Name	ISDU Index/subindex	Unique name (Name_Index)
0815 variable	_0815_variable	3452	_0815_variable_3452
Switchpoint (Q1)	Switchpoint__Q1_	345	Switchpoint__Q1_345
External temperature	External_temperature [1)]	311	External_temperature_311
External temperature	External_temperature [1)]	1423	External_temperature_1423
Note 1) this is a naming conflict (duplicate)			

4.4.6 Data Type conversion

The following conversion rules in Table 2 apply for the data type mapping between IO-Link and JSON.

Table 2 – Data type conversion

IO-Link data type	JSON data type
BooleanT	Boolean
StringT	String
ArrayT	Array
OctetStringT	see Byte Array Conversion 4.4.7
IntegerT, UintegerT	Number
Float32T	Number
RecordT	Object
TimeSpanT	String
TimeT	String

4.4.7 Byte Array conversion

This is the description how to map byte arrays to JSON.

Bit sequences that are not interpretable without using information out of the IODD are represented as JSON arrays of decimal numbers. One array item is representing one byte. If the value of the bit sequence is not a multiple of 8, the value is padded with zeros from the left until to the next byte border. Byte order is big-endian, see [1].

Example:

Table 3 – Mapping example of a bit sequence

value (bin)	bit length	value (hex)	Byte Array (dec)
10 0111	6	0x27	[39]
100 0000 1111 0000	15	0x40F0	[64, 240]

4.4.8 Time format

Any time values shall be represented either as an absolute time or a relative time in the format as specified in ISO 8601 (see [4])

Note: The italic letters in the following format definitions are to be replaced by one digit. The square brackets indicate that an element is optional.

The format for absolute time is *JJJJ-MM-TT*T*hh:mm:ss.f*Z

Time format for relative time is P[*JY*][*MM*][*WW*][*TD*][T[*h*H][*m*M][*s*[.*f*]S]]

Examples:

```
Absolute time:   2018-05-18T07:31:54.123Z
Relative time:   P3Y6M4DT12H30M17.123S
```

4.4.9 Error Behaviour

While processing HTTP request errors may occur. There are several errors defined (see appendix A.2). The body of an HTTP response indicating an error contains the Error Object as defined in B.1.2.

Following general rules apply for the error handling:

a) Errors 101 and 150 (see appendix A.2) can be returned to each request.

b) If parts of POST requests are not applicable, the HTTP response has a different HTTP status code that 2xx. The body of this HTTP response shall contain the Error Object as defined in B.1.2.

c) If multiple errors occur while processing the request only the first detected error shall be responded.

d) Errors 305 and 306 (see A.2) can be returned to requests when there are query parameters added to the URL.

e) If REST API commands are not available, error 103 (see A.2) shall be responded.

5 REST API

5.1 URL

The base path for this version is listed in Table 4

Table 4 – Base path

Base path	Remark	M/O/C
/iolink/v1	Example: “**/iolink/v1**/masters/1/ports/2/status”	M

5.2 HTTP Methods

The following HTTP methods shall be used

Table 5 – HTTP Methods

HTTP Methods	description	M/O/C
GET	Request data from the server	M

POST	Transmit data to the server	M
DELETE	Delete resources on the server	M
OPTIONS	List supported HTTP methods of the server	O

5.3 HTTP Requests / Responses

The URLs are build out of a base URL which is followed by a subsequent path. Query strings are added optionally.

The HTTP request headers `Content-type` and `Accept` are set to `application/json` by default.

Table 6 – Resources overview

Resource	clause	remark	M/O/C
`/gateway`	5.4	Address the Gateway	M
`/masters`	5.5	Get all available masterNumber keys and identification information	M
`/masters/{masterNumber}`	5.5	Address one specific master	M
`/masters/{masterNumber}/ports`	5.6.1	Get all available portNumber keys	M
`/masters/{masterNumber}/ports/{portNumber}`	5.6.1	Addressing one specific port of a specific master	M
`/devices`	5.7	Addressing all devices of all masters	M
`/devices/{deviceName}`	5.7.3	Addressing specific devices by name	M
`/mqtt`	5.9	Configuration of MQTT clients	C[2)]
`/iodds`	5.8	IODD file handling	C[1)]
Note 1) this resource is mandatory if the IODD feature is supported Note 2) this resource is mandatory if the MQTT feature is supported			

Examples:

```
HTTP://192.168.178.22/iolink/v1/gateway
HTTP://192.168.178.22/iolink/v1/masters/1
HTTP://192.168.178.22/iolink/v1/masters/2/ports/7
HTTP://192.168.178.22/iolink/v1/devices/sensor34
```

5.4 Gateway

Table 7 – Resources Gateway

Resources /iolink/v1/gateway	clause	HTTP Method	description	M/O/C
`/identification`	5.4.1	GET	Read the indentification of the REST API Gateway	M
`/capabilities`	5.4.2	GET	Read the capabilities of the gateway	M
`/configuration`	5.4.3	GET	Read the network configuration of the gateway	M
`/configuration`	5.4.4	POST	Write the network configuration of the gateway	M
`/reset`	5.4.5	POST	Reset the gateway including all masters	O
`/reboot`	5.4.6	POST	Reboot the gateway including all masters	O

/events	5.4.7	GET	Read the EventLog containing all events from gateway, masters, ports and devices.	M

5.4.1 GET /identification

Read the indentification of the REST API Gateway.

Table 8 – GET /identification

	Description
Description	Read the indentification of the REST API Gateway
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	–
Errors	301 (see A.2), see 4.4.9
Success	200 OK
Request body	–
Response body	Object Gateway Identification see Table 9

Table 9 – Object GatewayIdentification

object	GatewayIdentification			
property	**type**	**value**	**description**	**M/O/C**
macAddress	string	e.g. "00:02:72:CE:A6:49"	MAC address 6 bytes	M

Example :

Request

```
GET /iolink/v1/gateway/identification
```

Response

```
{
  "macAddress": "00:02:72:CE:A6:49"
}
```

5.4.2 GET /capabilities

Read the capabilities of the IO-Link gateway.

Table 10 – GET /capabilities

	Description
Description	Read the capabilities of the IO-Link gateway
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	–
Errors	301 (see A.2), see 4.4.9

Success	200 OK
Request body	–
Response body	object defined in Table 11 – Object GatewayCapabilities

Table 11 – Object GatewayCapabilities

object	GatewayCapabilities			
property	**type**	**value**	**description**	**M/O/C**
`ioddSupported`	boolean	true, false	true if the IODD feature is supported	M
`mqttSupported`	boolean	true, false	true if the MQTT feature is supported	M

Example:

Request

```
GET /iolink/v1/gateway/capabilities
```

Response

```
{
  "ioddSupported": true,
  "mqttSupported": false
}
```

5.4.3 GET /configuration

Read the actual acting configuration of the IO-Link gateway.

Table 12 – GET /configuration

	Description
Description	Read the actual acting configuration of the IO-Link gateway. The gateway may support multiple IPv4 interfaces.
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	–
Errors	301 (see A.2), see 4.4.9
Success	200 OK
Request body	–
Response body	object defined in Table 13 – Object GatewayConfiguration

Table 13 – Object GatewayConfiguration

Object	GatewayConfiguration			
property	**type**	**Value**	**description**	**M/O/C**
`ethIpv4`	Array of object	Objects defined in Table 14 – Object MasterConfiguration		M

255 **Table 14 – Object MasterConfiguration**
256

Object	MasterConfiguration			
property	**type**	**Value**	**description**	**M/O/C**
ipAddress	string		e.g. 192.168.1.13	M
subnetMask	string		e.g. 255.255.255.0	M
standardGateway	string		e.g. 192.168.1.1	M

257

258 **Example:**

259 Request

```
GET /iolink/v1/gateway/configuration
```

260 Response

```
{
  "ethIpv4": [
    {
      "ipAddress": "192.168.1.13",
      "subnetMask": "255.255.255.0",
      "standardGateway": "192.168.1.1"
    }
  ]
}
```

261

262 ### 5.4.4 POST /configuration

263 Write configuration data to the gateway.

264 **Table 15 – POST /configuration**
265

	Description
Description	Write configuration data to the gateway.
System Behaviour	If there are no restrictions the values of the object will be modified. The response shall be send prior to the change of the IP configuration.
Path parameters	–
Query parameters	–
Errors	201, 202, 203, 204, 205, 206 (see A.2), see 4.4.9
Success	204 No Content
Request body	object defined in Table 13 – Object GatewayConfiguration
Response body	–

266

267 **Example:**

268 Request (changing an IPv4 interface)

```
POST /iolink/v1/gateway/configuration

{
  "ethIpv4": [
    {
      "ipAddress": "192.168.1.13",
```

```
      "subnetMask": "255.255.255.0",
      "standardGateway": "192.168.1.1"
    }
  ]
}
```

5.4.5 POST /reset

This invokes a reset of the IO-Link Gateway. This may reset all configuration data and interrupt all communications channels. It is recommended to log this within the EventLog (see 5.4.7).

Table 16 – POST /reset

	Description
Description	This invokes a reset of the IO-Link Gateway
System Behaviour	If there are no restrictions the response will be given and following the gateway will be reset
Path parameters	–
Query parameters	–
Errors	(see A.2), see 4.4.9
Success	204 No Content
Request body	–
Response body	–

Example:

Request

```
POST /iolink/v1/gateway/reset
```

5.4.6 POST /reboot

This invokes a reboot of the IO-Link Gateway. This may reset all configuration data and interrupt all communications channels. It is recommended to log this within the EventLog (see 5.4.7).

Table 17 – POST /reboot

	Description
Description	This invokes rebooting of the IO-Link Gateway
System Behaviour	If there are no restrictions the response will be given and following the gateway will be reset
Path parameters	–
Query parameters	–
Errors	(see A.2), see 4.4.9
Success	204 No Content
Request body	–
Response body	–

288 **Example:**

289 Request

```
POST /iolink/v1/gateway/reboot
```

290

291 5.4.7 GET /events

292 Each gateway shall have an Event Log object containing the events of devices, ports and the
293 masters. The content of the Event Log can be read by getting the object “Gateway Event Log”
294 (see Table 20).

295 **Table 18 – GET /events**

296

	Description
Description	Reading the Event Log. A filtered subset of the Event log object is achieved by adding query parameters to the URL. If there are no events to respond the “Gateway Event Log” objects is empty.
System Behaviour	Nothing will be changed or modified.
Path parameters	–
Query parameters	Event log query parameters (see Table 19)
Errors	(see A.2) , see 4.4.9
Success	200 OK
Request body	–
Response body	object defined in Table 20 – Object GatewayEventLog

297

298 **Table 19 – Event log query parameters**

299

query parameter	**values**	**Remark**
origin	ALL, GATEWAY,MASTERS, PORTS, DEVICES	Default is ALL
masterNumber	{masterNumber}	*masterNumber* is only applicable with origin=MASTERS and with origin=PORTS
portNumber	{portNumber}	*portNumber* is only applicable with origin=PORTS
deviceName	{deviceName}	*deviceName* is only applicable with origin=DEVICES
top	1…n	Delivers the oldest n events of the event buffer. *top* is mutually exclusive to *bottom*.
bottom	1…m	Delivers the latest n events of the event buffer. *bottom* is mutually exclusive to *top*.

300

301 **Table 20 – Object GatewayEventLog**

302

object	**Gateway EventLog**			
property	**Type**	**value**	**description**	**M/O/C**
`time`	string	Relative or absolute time as defined in 4.4.8	When the event was logged	M
`severity`	string	**Enumeration:** “EMERGENCY” “ALERT” “CRITICAL” “ERROR” “WARNING”	Reference is Syslog Protocol RFC5424 see [5] Indicates the severity of the message. The severity is decreasing from EMERGENCY to DEBUG. IO-Link EventTypes mapping is:	M

		“NOTICE” “INFO” “DEBUG”	"NOTIFICATION" maps to "NOTICE", "WARNING" to "WARNING" and "ERROR" to "ERROR".	
`origin`	object	object defined in Table 21 – Object Origin		M
`message`	object	object defined in Table 22 – Object Message		M

303

304 **Table 21 – Object Origin**

305

object				
property	**type**	**value**	**description**	**M/O/C**
`gateway`	string	vendor-specific String	This is for vendor specific origins e.g. Web server application or MQTT, etc.	C[1]
`masterNumber`	number	masterNumber		C[2]
`portNumber`	number	portNumber		C[3]
`deviceName`	string	deviceName		C[4]
Note 1) Origin gateway has to be used if the event comes from the gateway. Note 2) Origin masterNumber has to be used if the event comes from master, port or device. Note 3) Origin portNumber has to be used if the event comes from port or device. Note 4) Origin deviceName has to be used if the event comes from the device.				

306

307 **Table 22 – Object Message**

308

object	**Message**			
property	**type**	**value**	**description**	**M/O/C**
`code`	number	codes according in [1] Annex D shall be used if the origin is device or port		C[1]
`mode`	string	codes according in [1] 8.2.2.11 shall be used if the origin is device or port	SINGLE_SHOT APPEARS DISAPPEARS	C[1]
`text`	string	Message. (For IO-Link event code related messages (defined in [1]) shall be used)		C[1]
Note 1) At least one of the properties has to be used.				

309

310 **Examples:**

311 Reading all events of the gateway

312 Request

```
GET /iolink/v1/gateway/events
```

313 Response

```
[
```

```
  {
    "time": "2018-05-18T07:31:54.123Z",
    "severity": "WARNING",
    "origin": {
      "masterNumber": 1,
      "portNumber": 1,
      "deviceName": "Temp_sensor_1"
    },
    "message": {
      "code": 16912,
      "mode": "APPEARED",
      "text": "Device temperature over-run – Clear source of heat"
    }
  },
  {
    "time": "2018-05-18T07:35:54.123Z",
    "severity": "NOTIFICATION",
    "origin": {
      "masterNumber": 1,
      "portNumber": 2
    },
    "message": {
      "code": 65314,
      "mode": "SINGLESHOT",
      "text": "Device communication lost"
    }
  }
]
```

Request reading the oldest 3 events of port 7 of master 2

```
GET /iolink/v1/gateway/events?origin=PORTS&top=3&portNumber=7
```

Request reading all events of master 3

```
GET /iolink/v1/gateway/events?origin=MASTERS&masterNumber=3
```

5.5 Master

There can be more than 1 master addressed by one gateway. Typical multi master applications are modular devices. Masters within a gateway are addressed by “masterNumber” which starts at 1 for the first master.

Example:

```
HTTP://192.168.178.22/iolink/v1/masters/1
```

5.5.1 GET /

This is for scanning all masters within a Gateway.

Table 23 – GET /masters

	Description
Description	read all the available masterNumber keys with the corresponding identification information
System Behaviour	Nothing will be changed or modified

Path parameters	–
Query parameters	–
Errors	(see A.2), see 4.4.9
Success	200 OK
Request body	-
Response body	Array of object defined in Table 24 – Object MasterID

330

331 **Table 24 – Object MasterID**

332

object	MasterID			
property	**type**	**value**	**description**	**M/O/C**
masterNumber	number	Integer 1..n		M
serialNumber	string		Vendor specific	O
locationTag	string		Could be used for “slot” information for the modular model (see 4.1)	O

333

334 **Example:**

335 Request

```
HTTP://192.168.178.22/iolink/v1/masters
```

336 Response

```
[
  {
    "masterNumber": 1,
    "serialNumber": "lsdfkjke93922",
    "locationTag": "slot 5"
  },
  {
    "masterNumber": 2,
    "serialNumber": "dsadasda25632",
    "locationTag": "slot 6"
  }
]
```

337
338

339 **Table 25 – Resources Master**

340

Resources /iolink/v1/masters/ {masterNumber}	clause	HTTP Method	remark	M/O/C
/capabilities	5.5.2	GET	Read the capabilities of the master	M
/identification	5.5.3	GET	Read the identification of the IO-Link master	M
/identification	5.5.4	POST	Write application specific identification to a master	M

341

342 ### 5.5.2 GET /capabilities

343 This is for reading the capabilities of an IO-Link master.

344 **Table 26 – GET /capabilities**
345

	Description
Description	Read the capabilities of the IO-Link master
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	–
Errors	302 (see A.2), see 4.4.9
Success	200 OK
Request body	–
Response body	Object Master Capability (see Table 27 – Object MasterCapabilities

346
347 **Table 27 – Object MasterCapabilities**
348

Object	MasterCapabilities			
property	**type**	**values**	**description**	**M/O/C**
`numberOfPorts`	number	Integer 1..n	Total number of ports	M
`maxPowerSupply`	object	object defined in Table 110 – Object PowerSupply	Maximum total power supply for all ports.	M

349

350 **Example:**

351 Request

```
GET /iolink/v1/masters/1/capabilities
```

352 Response

```
{
  "numberOfPorts": 8,
  "maxPowerSupply": {
    "value": 0.3,
    "unit": "A"
  }
}
```

353

354 ### 5.5.3 GET /identification

355 This is for reading identification data of an IO-Link master

356 **Table 28 – GET /identification**
357

	Description
Description	Read all identification data of an IO-Link master
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	–
Errors	302 (see A.2), see 4.4.9
Success	200 OK
Request body	–
Response body	object defined in Table 29 – Object MasterIdentificationGet

Table 29 – Object MasterIdentificationGet

object	MasterIdentificationGet			
property	**type**	**value**	**description**	**M/O/C**
vendorId	integer		Vendor ID assigned by IO-Link community, see [1]	M
masterID	integer		Any vendor specific ID	O
masterType	string	**Enumeration:** “Unspecific” “Master acc. V1.0” “Master acc. V1.1” “Failsafe_Master” “Wireless_Master”	see [1]	M
serialNumber	string		Vendor specific, see [1]	O
orderCode	string		Vendor specific, see [1]	M
vendorName	string		Vendor specific, see [1]	M
hardwareRevision	string		Vendor specific, see [1]	O
softwareRevision	string		Vendor specific, see [1]	O
website	string		Link to product site	O
manual	string		Link to user manual	O
applicationSpecificTag	string		User defined tag, see [1]	O
locationTag	string		User defined tag, see [1]	O
functionTag	string		User defined tag, see [1]	O

Example:

Request

```
GET /iolink/v1/masters/1/identification
```

Response

```
{
  "vendorName": "SICK AG",
  "vendorId": 26,
  "masterId": 42,
  "masterType": "Master acc. V1.0",
  "serialNumber": "IOLM123456",
  "orderCode": "PROD-123456",
  "productName": "IO-Link Master",
  "hardwareRevision": "3.2.1.444R",
  "softwareRevision": "3.2.1.888R",
  "website": "http://www.sick.de/io-link-master",
  "manual": "http://www.sick.de/io-link-master/documentation.pdf",
  "applicationSpecificTag": "Fallback reader at the end of the belt",
  "locationTag": "Down under",
  "functionTag": "Code reading"
}
```

5.5.4 POST /identification

This is for setting the identification data of an IO-Link master.

Table 30 – POST /identification

	Description
Description	write all identification data of an IO-Link master
System Behaviour	if there are no restrictions the object will be modified
Path parameters	–
Query parameters	–
Errors	302 (see A.2), see 4.4.9
Success	204 No Content
Request body	object defined in Table 31 – Object MasterIdentificationPost
Response body	-

Table 31 – Object MasterIdentificationPost

object	**MasterIdentificationPost**			
property	**type**	**Value**	**description**	**M/O/C**
applicationSpecificTag	string		User defined tag, see [1]	O
locationTag	string		User defined tag, see [1]	O
functionTag	string		User defined tag, see [1]	O

Example

Request

```
POST /iolink/v1/masters/1/identification

{
  "applicationSpecificTag": "Fallback reader at the end of the belt",
  "locationTag": "Down under",
  "functionTag": "Code reading"
}
```

5.6 Port

5.6.1 GET /

Get all the available portId keys of one specific master.

Table 32 – GET /{masterNumber}/ports

	Description
Description	read all the available portNumber keys with the corresponding identification information
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	–
Errors	302 (see A.2), see 4.4.9

Success	200 OK
Request body	-
Response body	array of object defined in Table 33 – Object Port ID

Table 33 – Object Port ID

object	Port IDs			
property	**type**	**value**	**description**	**M/O/C**
portNumber	number	Integer 1..n		M

Example:

Request

```
HTTP://192.168.178.22/iolink/v1/masters/1/ports
```

Response

```
[
  {
    "portNumber": 1
  },
  {
    "portNumber": 2
  },
  {
    "portnumber": 3
  },
  {
    "portnumber": 4
  }
]
```

Table 34 – Resources Port

Resources masters/{masterNumber}/ ports/{portNumber}	clause	HTTP Method	remark	M/O/C
/capabilities	5.6.2	GET	Read the capability information of the specified port	M
/status	5.6.3	GET	Read the actual status of the selected port	M
/configuration	5.6.4	GET	Read the actual configuration of the specified port	M
/configuration	5.6.5	POST	Write the configuration of the specified port	M
/datastorage	5.6.6	GET	Read the datastorage content of a specific port	M
/datastorage	5.6.7	POST	Write the datastorage content to a specific port	M

5.6.2 GET /capabilities

Reads the capabilities of a specific port.

398 **Table 35 – GET /capabilities**
399

	Description
Description	Reads the capabilities of a specific port
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	–
Errors	302, 303 (see A.2), see 4.4.9
Success	200 OK
Request body	–
Response body	object defined in Table 36 – Object PortCapability

400

401

402 **Table 36 – Object PortCapability**
403

object	**Port capability**			
property	**type**	**value**	**remark**	**M/O/C**
`portType`	string	**Enumeration:** “CLASS_A” “CLASS_B” “CLASS_A_WITH_PORT_POWER_OFF_ON” “FAILSAFE_PORT_A_WITHOUT_SAFETY_DIGITAL_INPUTS” “FAILSAFE_PORT_A_WITH_ SAFETY_DIGITAL_INPUTS” “FAILSAFE_PORT_B” “WIRELESS_MASTER”	see [1], SMI	M
`maxPowerSupply`	object	object defined in Table 110 – Object PowerSupply		M

404

405 **Example:**

406 Request

```
HTTP://192.168.178.22/iolink/v1/masters/1/ports/1/capability
```

407 Response

```
{
  "maxPowerSupply": {
    "value": 0.3,
    "unit": "A"
  },
  "portType": "CLASS_A"
}
```

408

409

410 ### 5.6.3 GET /status

411 Reads the actual status of a specific port.

412 **Table 37 – GET /status**
413

	Description
Description	Reads the actual status of a specific port

System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	–
Errors	302, 303 (see A.2), see 4.4.9
Success	200 OK
Request body	–
Response body	object defined in Table 38 – Object PortStatus

414

415 **Table 38 – Object PortStatus**

416

object	**Port Status**			
property	**type**	**value**	**remark**	**M/O/C**
`statusInfo`	string	**Enumeration:** “DEACTIVATED” “INCORRECT_DEVICE” “DEVICE_STARTING” “DEVICE_ONLINE” “COMMUNICATION_LOST” “DI_C/Q DIGITAL_INPUT_PIN4” “DO_C/Q DIGITAL_OUTPUT_PIN4” “NOT_AVAILABLE”	see Table 39 – port status mapping	M
`ioLinkRevision`	string	**Enumeration:** “1.0” “1.1”		C[1]
`transmissionRate`	string	**Enumeration:** “COM1” “COM2” “COM3”		C[1]
`masterCycleTime`	object	Object defined in Table 108 – Object CycleTime		C[2]
Note 1) only applicable if port is in status “DEVICE_ONLINE” or “DEVICE_STARTING” Note 2) only applicable if port is in status “DEVICE_ONLINE”				

417

418 **Table 39 – port status mapping**

419

JSON	SMI
DEACTIVATED	DEACTIVATED, PORT_POWER_OFF
INCORRECT_DEVICE	PORT_DIAG
DEVICE_STARTING	PREOPERATE
DEVICE_ONLINE	OPERATE
COMMUNICATION_LOST	NO_DEVICE
DI_C/Q DIGITAL_INPUT_PIN4	DI_C/Q
DO_C/Q DIGITAL_OUTPUT_PIN4	DO_C/Q
NOT_AVAILABLE	NOT_AVAILABLE

420
421
422

423 **Example:**

Request

```
HTTP://192.168.178.22/iolink/v1/masters/1/ports/1/status
```

Response

```
{
  "portStatusInfo": "DEVICE_ONLINE",
  "ioLinkRevision": "1.1",
  "transmissionRate": "COM2",
  "masterCycleTime": {
    "value": 2.3,
    "unit": "ms"
  }
}
```

5.6.4 GET /configuration

Reads the actual configuration of a specific port

Table 40 – GET /configuration

	Description
Description	Read the actual configuration of a specific port
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	–
Errors	302, 303 (see A.2), see 4.4.9
Success	200 OK
Request body	–
Response body	object defined in Table 41 – Object PortConfiguration

Table 41 – Object PortConfiguration

object	**Port configuration**			
property	**type**	**values**	**description**	**M/O/C**
mode	string	**Enumeration:** “DEACTIVATED” “IOLINK_MANUAL” “IOLINK_AUTOSTART” “DIGITAL_INPUT” “DIGITAL_OUTPUT”	see [1], Table E3	M
validationAndBackup	string	**Enumeration:** “NO_DEVICE_CHECK” “TYPE_COMPATIBLE_DEVICE_V1.0” “TYPE_COMPATIBLE_DEVICE_V1.1” “TYPE_COMPATIBLE_DEVICE_V1.1_BACKUP_AND_RESTORE” “TYPE_COMPATIBLE_DEVICE_V1.1_RESTORE”	see [1], Table E3 only available if ports is in mode IOLINK MANUAL	M
pin2Configuration	string	**Enumeration:** “NOT_SUPPORTED” “DIGITAL_INPUT” “DIGITAL_OUTPUT”		M/C[2]
cycleTime	object	Object defined in Table 108 – Object	Note [4]	

		CycleTime		
vendorId	number		see [1]	C[1]
deviceId	number		see [1]	C[1]
deviceName	String	Default Name is: Maste{masterNumer}port{portNumber}		M/O[3]
Note 1) Validation is only used if port is in mode “IOLINK MANUAL” Note 2) Mandatory for GET request, C for POST request Note 3) Mandatory for GET request, O for POST request. For POST the deviceName shall have no white space (see also 4.4.3). Duplicate names shall lead to Error 207 (see A.2) Note 4) If the applied value for cycle time cannot exactly be mapped, the port shall use the next possible lower value.				

Example:

Request

```
HTTP://192.168.178.22/iolink/v1/masters/1/ports/1/configuration
```

Response

```
{
  "mode": "IOLINK_MANUAL",
  "validationAndBackup": "TYPE_COMPATIBLE_DEVICE_V1.1",
  "pin2Configuration": "DIGITAL_INPUT",
  "cycleTime": {
    "value": 2.3,
    "unit": "ms"
  },
  "vendorId": 26,
  "deviceId": 333,
  "deviceName": "Distance_sensor_1"
}
```

5.6.5 POST /configuration

This is for writing the wanted port configuration

Table 42 – POST /configuration

	Description
Description	writes the wanted configuration for a specific port
System Behaviour	if there are no restrictions the object will be modified If the applied value for cycle time cannot exactly be mapped, the port shall use the next possible lower value. By changing the configuration the communication to the device can get lost
Path parameters	–
Query parameters	–
Errors	201, 202, 203, 204, 205, 206, 207, 302, 303 (see A.2), see 4.4.9
Success	204 No Content
Request body	object defined in Table 41 – Object PortConfiguration
Response body	–

Example:

Request

```
HTTP://192.168.178.22/iolink/v1/masters/1/ports/1/configuration

{
  "mode": "IOLINK_AUTOSTART",
  "validationAndBackup": "TYPE_COMPATIBLE_DEVICE_V1.1",
  "pin2Configuration": "ANALOG_INPUT",
  "cycleTime": {
    "value": 2.3,
    "unit": "ms"
  },
  "vendorId": 26,
  "deviceId": 333,
  "deviceName": "Distance_sensor_1"
}
```

5.6.6 GET /datastorage

Getting the datastorage content (bulk data format is described [1]), added by the header (VendorId, DeviceId, Revision ID).

Table 43 – GET /datastorage

	Description
Description	Read the actual datastorage content of the port
System Behaviour	Nothing will be changed or modified If no datastorage content is available the response is empty.
Path parameters	–
Query parameters	–
Errors	302, 303 (see A.2), see 4.4.9
Success	200 OK
Request body	–
Response body	Object defined in Table 44 – Object Datastorage

Table 44 – Object Datastorage

object	**Datastorage**			
property	**type**	**value**	**description**	**M/O/C**
`header`	object	Object defined in Table 45 – Object DataStorageHeader	Object is empty if there is no datastorage content.	M
`content`	string	Base 64 coded	String is empty if there is no datastorage content. see [7] , see [1] E6	M

Table 45 – Object DataStorageHeader

object	**Data Storage Header**			
property	**type**	**Value**	**description**	**M/O/C**
`vendorId`	number		see [1]	M
`deviceId`	number		see [1]	M
`ioLinkRevision`	string	**Enumeration** “1.0”	see [1]	M

		“1.1”		

Example:

Request

```
HTTP://192.168.178.22/iolink/v1/masters/1/ports/1/datastorage
```

Response

```
{
  "header": {
    "vendorId": 15,
    "deviceId": 65253,
    "ioLinkRevision": "1.1"
  },
  "content": "dasdASDasddaSdASdASdasDasdASdasDASdasasd"
}
```

5.6.7 POST /datastorage

Writing the datastorage content (bulk data format is described [1]), added by the header (VendorId, DeviceId, Revision ID).

Table 46 – POST /datastorage

	Description
Description	Write datastorage content of the port
System Behaviour	if there are no restrictions the object will be modified After receiving the datastorage content the port is restarted if the port is in mode “Backup&Restore” or “Restore” see also [1] and see 5.6.5
Path parameters	–
Query parameters	–
Errors	201, 202, 203, 204, 205, 206, 302, 401(see A.2), see 4.4.9
Success	204 No Content
Request body	Object defined in Table 44 – Object Datastorage
Response body	–

Example:

Request

```
POST iolink/v1/masters/1/ports/1/datastorage

{
  "header": {
    "vendorId": 15,
    "deviceId": 65253,
    "ioLinkRevision": "1.1"
  },
  "content": "dasdASDasddaSdASdASdasDasdASdasDASdasasd"
}
```

5.7 Devices

Devices are addressed by device names. See port configuration in 5.6.5. for device name assignement. If no device name is assigned the default name is used.

Example:

Addressing by default device name (e.g. device is on port 6 of master1)

```
HTTP://192.168.178.22/iolink/v1/devices/master1port6
```

Addressing by assigned device name (e.g. sensor34)

```
HTTP://192.168.178.22/iolink/v1/devices/sensor34
```

5.7.1 GET /

Get all available deviceName keys and the location by master number and port number

Table 47 – GET /devices

	Description
Description	read all available deviceName keys and the location by master number and port number
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	–
Errors	(see A.2), see 4.4.9
Success	200 OK
Request body	-
Response body	Array of object defined in Table 48 – Object DeviceName

Table 48 – Object DeviceName

object	**DeviceName**			
property	**type**	**value**	**description**	**M/O/C**
deviceName	string		Device name is according 4.4.5	M
masterNumber	number	Integer 1..n		M
portNumber	number	Integer 1..n		M

Example:

Request

```
GET /iolink/v1/devices
```

Response

```
[
  {
    "deviceName": "DT35",
    "masterNumber": 1,
    "portNumber": 1
```

```
  },
  {
    "deviceName": "TAD081",
    "masterNumber": 1,
    "portNumber": 2
  },
  {
    "deviceName": "BNI_IOL",
    "masterNumber": 1,
    "portNumber": 3
  },
  {
    "deviceName": "master1port4",
    "masterNumber": 1,
    "portNumber": 4
  }
]
```

492
493
494
495
496

Table 49 – Resources Devices

Resources /devices/{deviceName}	**clause**	**HTTP Method**	**remark**	**M/O/C**
capabilities				
/capabilities	5.7.2	GET	Read the capabilities of the specific device	M
Identification				
/identification	5.7.3	GET	Read the identification of the specific device	M
/identification	5.7.4	POST	Write application specific identification to the device	M
Process data				
/processdata/value	5.7.5	GET	Read the process data value of the specified device format “byteArray” or “iodd” is given by a query string	M
/processdata/value	5.7.6	POST	Write the process data to the specified device format “byteArray” or “iodd” is given by the request body	M
Parameter information				
/parameters	5.7.7	GET	Read all available parameter indices and parameter names of the specific device. IODD support is required.	C [1)]
/parameters /{index}/subindices	5.7.8	GET	Read all available parameter sub-indices and sub-parameter names of the specific device. IODD support.is required	C[1)]
/parameters /{parameterName}/subindices	5.7.9	GET	Read all available parameter sub-indices and sub-parameter names of the specific device (referenced by parameter name). IODD support.is required.	C[1)]
Parameter values				
/parameters /{index}/value	5.7.10	GET	Read a parameter value of the specific device with the given index. The requested format “byteArray” or “iodd” is given by a query string	M
/parameters /{index}/subindices /{subindex}/value	5.7.11	GET	Read a parameter value of the specific device with the given index and subindex. The requested format “byteArray” or “iodd” is given by a query string	M
/parameters	5.7.12	GET	Read a parameter value of the specific device	C[1]

/{parameterName}/value			by parameter name. IODD support is required	
/parameters /{ parameterName } /subindices /{subParametername}/value	5.7.13	GET	Read a parameter value of the specific device by parameter name and subname. IODD support is required.	C[1]
/parameters /{index}/value	5.7.14	POST	Write a parameter value with the given index to the specified device. The format is given within the body .	M
/parameters /{parameterName}/value	5.7.15	POST	Write a parameter value by name to the specified device. The format is given within the body. IODD support is required.	C[1]
/parameters /{index}/subindices /{subindex}/value	5.7.16	POST	Write a parameter value with the given index and subindex to the specified device. The format is given within the body.	M
/parameters /{parameterName}/ /subindices /{subParameterName}/value	5.7.17	POST	Write a parameter value to the specific device by the parameter name and subname. The format is given within the body. IODD support is required.	C[1]
/blockparameterization	5.7.18	POST	Writing or reading one or more parameters by using the block parameterization method. The format can be byteArray or iodd based.	M
Events				
/events	5.7.19	GET	reading the EventLog filtered for a specific device	M
Note 1) M if IODD support is available				

5.7.2 GET /capabilities

Read the capabilities of the specific device

Table 50 – GET /capabilities

	Description
Description	Read the capabilities of the specific device
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	–
Errors	302, 303, 304, 308 (see A.2), see 4.4.9
Success	200 OK
Request body	–
Response body	Object defined in Table 51 – Object DeviceCapabilities

Table 51 – Object DeviceCapabilities

object	**Device capabilities**			
property	**type**	**Value**	**description**	**M/O/C**
`minimumCycleTime`	object	object defined in Table 108 – Object CycleTime		M

Example

Request

```
GET /iolink/v1/devices/sensor34/capabilities
```

Response

```
{
  "minimumCycleTime": {
      "value": 2.3,
      "unit": "ms"
   }
}
```

5.7.3 GET /identification

Read the identification of the device at the specified port

Table 52 – GET /identification

	Description
Description	Read the identification of the device
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	–
Errors	302, 303, 304, 308 (see A.2), see 4.4.9
Success	200 OK
Request body	–
Response body	Object defined in Table 53 – Object Device Identification Get

Table 53 – Object Device Identification Get

object	**Device Identification GET**			
property	**type**	**Value**	**description**	**M/O/C** [1)]
vendorId	number		see [1]	M
deviceId	number		see [1]	M
ioLinkRevision	string		see [1]	M
vendorName	string		see [1]	C
vendorText	string		see [1]	C
productName	string		see [1]	C
productText	string		see [1]	C
serialNumber	string		see [1]	C
hardwareRevision	string		see [1]	C
firmwareRevision	string		see [1]	C
applicationSpecificTag	string		see [1]	C
locationTag	string		see [1]	C
functionTag	string		see [1]	C

Note 1) M/O/C regarding [1]

Example:

Request

```
GET /iolink/v1/devices/sensor34/identification
```

Response

```
{
  "vendorId": 26,
  "deviceId": 42,
  "ioLinkRevision": "1.1",
  "vendorName": "SICK AG",
  "vendorText": "Sensor Intelligence.",
  "productName": "WTxx16",
  "productId": "PROD-123456",
  "productText": "Light switch",
  "serialNumber": "IOLM123456",
  "hardwareRevision": "3.2.1.444R",
  "firmwareRevision": "3.2.1.888R",
  "applicationSpecificTag": "Fallback light switch at the end of the belt",
  "locationTag": "Down under",
  "functionTag": "Check end of belt"
}
```

5.7.4 POST /identification

Writes device specific tags including applicationspecificTag, the locationTag and the functionTag (see [1]) if available.

Table 54 – POST /identification

	Description
Description	write all identification data of an IO-Link master
System Behaviour	if there are no restrictions the object will be modified
Path parameters	–
Query parameters	–
Errors	302, 303, 304, 309, 310 (see A.2), see 4.4.9
Success	204 No Content
Request body	–
Response body	Object defined in Table 55 – Object DeviceIdentificationPOST

Table 55 – Object DeviceIdentificationPOST

object	**DeviceIdentificationPOST**			
property	**type**	**Value**	**description**	**M/O/C** [1)]
`applicationSpecificTag`	string		see [1]	O
`locationTag`	string		see [1]	O
`functionTag`	string		see [1]	O
Note 1) M/O/C regarding [1]				

Example:

Request

```
POST /iolink/v1/devices/sensor34/identification

{
  "applicationSpecificTag": "Fallback light switch at the end of the belt",
  "locationTag": "Down under",
  "functionTag": "Check start of belt"
}
```

5.7.5 GET /processdata/value

Read the process data value of the specified device. The request format can be byteArray or IODD based. If IODD format is requested the IODD feature has to be supported.

Process data can always accessed by addressing a device even if no physical device is attached at the port. Process data comprises digital inputs on PIN 2 and PIN 4 (also if the device is in “SIO” mode) as well as IO-Link process data if the port is in communication.

For getting process data 3 different modes indicated by “content” can be used. Typically input process data are read, indicated by content “measure”. Additionally wanted output data (usually set by POST) can be read back by content “control”. Also a combination of both directions can be used indicated by content “measurecontrol”.

Table 56 – GET / processdata/value

	Description
Description	Read the process data (input data and optionally the output data) of the specified device. The requested format is indicated by a query string and can be byteArray (default) or IODD based. The requested content is indicated by a query string and can be measure, control or measurecontrol
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	Format request parameters (see Table 57) and process data content parameters)see Table 58)
Errors	(see A.2), see 4.4.9
Success	200 OK
Request body	–
Response body	Object defined in Table 59 – Object ProcessDataInOut

Table 57 – format request query parameters

query parameter	**values**	**Remark**
format	**Enumeration:** byteArray iodd	Default is byteArray

552 **Table 58 – process data content query parameters**
553

query parameter	values	Remark
content	**Enumeration:** measure control measurecontrol	measure (default) is reading only input data from a sensor device control is reading back output data to an actuator device measurecontrol is reading input data and reading back of output data of a device

554

555

556 **Table 59 – Object ProcessDataInOut**
557

object	ProcessDataInOut			
property	**type**	**Value**	**description**	**M/O/C**
measure	object	Object defined in Table 60 – Object ProcessData		C[1]
control	object	Object defined in Table 60 – Object ProcessData		C[2]
Note 1) if there are process input data, the process data content is “measure” or “measurecontrol” Note 2) if there are process output data, the process data content is “control” or “measurecontrol”				

558

559 **Table 60 – Object ProcessData**
560

object	ProcessData			
property	**type**	**Value**	**description**	**M/O/C**
ioLink	object	Object defined in Table 61 – Object processDataIOL		C[1]
pin2Value	boolean	true, false	true = + 24 Volt	C[2]
Pin4Value	boolean	true, false	true = + 24 Volt	C[3]
Note 1) If port is in IO-Link mode and device is connected Note 2) If Pin 2 is available and configured as digital input for GET or configured as digital output for POST Note 3) If Pin 4 is available and configured as digital input for GET or configured as digital output for POST				

561

562 **Table 61 – Object processDataIOL**
563

object	processDataIOL			
property	**type**	**Value**	**description**	**M/O/C**
valid	boolean	true,false	For output data valid is always true.	M
value	array numbers	Byte array, see 4.4.7	order in big endian	C[1]
value	boolean, string, number		simple types	C[2]
value	object	Object defined in Table 70 – Object complexParameterValue		C[3]
Note 1) If format is byteArray Note 2) If format is iodd and process data are of simple type, naming shall be according 4.4.5 Note 3) If format is “iodd” and process data are of complex type, naming shall be according 4.4.5				

Examples:

Device is a sensor in IO-Link communication, Pin 2 is available and configured as digital input

The IO-Link process data is 0xC16D8 = 792280dez

Request (format is byteArray, content is measure).

```
GET
/iolink/v1/devices/master1port6/processdata/value?format=byteArray&content=measure
```

Response

```
{
  "measure": {
    "ioLink": {
       "valid": true,
       "value": [12,22,216]
    },
    "pin2Value": true
  }
}
```

Device is a sensor in IO-Link communication and Pin 2 is available configured as a digital input. The process data are of simple type.

Request (format IODD, content is measure).

```
GET /iolink/v1/devices/master1port6/processdata/value?format=iodd&content=measure
```

Response

```
{
  "measure": {
    "ioLink": {
       "valid": true,
       "value": 15.2
    },
    "pin2Value": true
  }
}
```

Device is a sensor in IO-Link communication and Pin 2 is available configured as a digital input. The process data are of complex type

Request (format IODD, content is measure).

```
GET /iolink/v1/devices/master1port6/processdata/value?format=iodd&content=measure
```

Response

```
{
  "measure": {
    "ioLink": {
       "valid": true,
       "value": {
          "Distance": 55,
          "Quality": 12
        }
```

```
    },
    "pin2Value": true
  }
}
```

Device is a sensor in SIO Mode, Pin 2 and PIN 4 are available and configured as digital inputs

Request (format IODD, content is input).

```
GET /iolink/v1/devices/master1port6/processdata/value?format=iodd&content=measure
```

Response

```
{
  "measure": {
    "pin2Value": true,
    "pin4Value": false
  }
}
```

Device is a sensor/actuator in IO-Link communication, Pin 2 is available and configured as a digital input.

Request (format byteArray, content is measurecontrol)

```
GET
/iolink/v1/devices/master1port6/processdata/value?format=byteArray&content=measurecon
trol
```

Response

```
{
  "measure": {
      "ioLink": {
         "valid": true,
         "value": [12,22,216]
       },
      "pin2Value": false
  },
  "control": {
      "ioLink": {
         "valid": true,
         "value": [128,221,134]
       }
  }
}
```

Device is a sensor/actuator in IO-Link communication, Pin 2 is available and configured as a digital input.

Request (format IODD, content is measurecontrol)

```
GET
/iolink/v1/devices/master1port6/processdata/value?format=iodd&content=measurecontrol
```

Response

```
{
  "measure": {
      "ioLink": {
```

```
        "valid": true,
        "value": {"Pressure": 1.2}
      },
      "pin2Value": false
   },
   "control": {
      "ioLink": {
         "valid": true,
         "value": {"Temperature": 50}
      }
   }
}
```

5.7.6 POST /processdata/value

Write the process output data to the specified device. Process data can also be written by addressing a device even if no physical device is attached at the port but Pin2 or PIN 4 are available. The device is addressed by the device name (see 4.4.3)

Table 62 – POST /processdata/value

	Description
Description	Write the process output data to the specified device. The requested format is implicit defined within the body. No query parameters are needed.
System Behaviour	if there are no restrictions process data will be updated
Path parameters	–
Query parameters	–
Errors	201, 202, 203, 204, 205, 206, 302, 303, 304, 307, 308, 501, 502, 503, 601 (see A.2), see 4.4.9
Success	204 No Content
Request body	Object defined in Table 60 – Object ProcessData
Response body	–

Examples:

Device is an actuator in IO-Link communication, Pin 2 is available and configured as a digital output.

Request (implicit format byteArray)

```
POST /iolink/v1/devices/master1port6/processdata/value

{
   "pin2Value": false,
   "ioLink": {
       "valid": true,
       "value": [12,22,216]
    }
}
```

Device is an actuator in IO-Link communication, Pin 2 is available and configured as a digital output. Process data is of simple type

Request (implicit format IODD, simple type)

```
POST /iolink/v1/devices/master1port6/processdata/value
```

```
{
   "pin2Value": true,
   "ioLink": {
       "valid": true,
       "value": 32.4
    }
}
```

Device is an actuator in IO-Link communication, Pin 2 is available and configured as a digital output. Process data are of complex type

Request (implicit format IODD, complex type)

```
POST /iolink/v1/devices/master1port6/processdata/value

{
   "pin2Value": true,
   "ioLink": {
      "valid": true,
      "value": {
        "Valve_1": true,
        "Valve_2": false
      }
   }
}
```

5.7.7 GET /parameters

Read all available parameter indices and parameter names of the specific device. IODD support is required.

Table 63 – GET /parameters

	Description
Description	Reads a list of all available parameters of a device
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	–
Errors	302,303,304 (see A.2), see 4.4.9
Success	200 OK
Request body	–
Response body	Array of object defined in Table 64 – Object parameterList

Table 64 – Object parameterList

object	**parameterList**			
property	**type**	**Value**	**description**	**M/O/C** [1)]
index	integer			M
parameterName	string	Name of parameter	see 4.4.5	M
Note 1) regarding the service itself which is conditional to IODD support				

Example:

Request

```
GET /iolink/v1/devices/sensor34/parameters
```

Response

```
[
  {
    "index": 10,
    "parameterName": "Vendor_Name"
  },
  {
    "index": 12,
    "parameterName": "Product_Name"
  },
  {
    "index": 13,
    "parameterName": "ProductID"
  }
]
```

5.7.8 GET /parameters/{index}/subindices

Get all the available parameter sub-indices, sub-parameter names (referenced by Index number). IODD support is required

Table 65 – GET /parameters/{index}/subindices

	Description
Description	Get all the available parameter sub-indices, sub-parameter names (referenced by Index number).
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	–
Errors	302, 303, 304, 308, 309 (see A.2), see 4.4.9
Success	200 OK
Request body	–
Response body	Array of object defined in Table 66 – Object deviceSubParameters

Table 66 – Object deviceSubParameters

object	**deviceSubParameters**			
property	**type**	**value**	**remark**	**M/O/C**[1)]
`subIndex`	integer			M
`subParameterName`	string	Name of sub parameter	see 4.4.5	M
Note 1) regarding the service itself which is conditional to IODD support				

Example:

Request

```
GET /iolink/v1/devices/sensor34/parameters/64/subindices
```

Response

```
[
  {
    "subIndex": 1,
    "subParameterName": "switching_mode"
  },
  {
    "subIndex": 2,
    "subParameterName": "switching_value_on"
  },
  {
    "subIndex": 3,
    "subParameterName": " switching_value_off"
  }
]
```

5.7.9 GET /parameters/{parameterName}/subindices

Get all the available parameter sub-indices, sub-parameter names (referenced by parameter name). IODD support required.

Table 67 – GET /parameters/{parameterName}/subindices

	Description
Description	Get all the available parameter sub-indices, sub-parameter names (referenced by parameter name).
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	–
Errors	302, 303, 304, 308, 309, 312 (see A.2), see 4.4.9
Success	200 OK
Request body	–
Response body	Array of object defined in Table 66 – Object deviceSubParameters

Example:

A device has a parameter named “Switch_properties” with some sub indices.

Request

```
GET /iolink/v1/devices/sensor34/parameters/Switch_properties/subindices
```

Response

```
[
  {
    "subIndex": 1,
    "subParameterName": "switching_mode"
  },
  {
    "subIndex": 2,
    "subParameterName": "switching_value_on"
  },
  {
    "subIndex": 3,
    "subParameterName": " switching_value_off"
  }
```

```
]
```

5.7.10 GET /parameters/{index}/value

This function reads a specific parametervalue and its sub-parameter values with the given index from the specified device. The requested format is indicated by a query string and can be byteArray (default) or IODD based. For format “iodd” the IODD support is needed

Table 68 – GET /parameters/{index}/value

	Description
Description	reads a specific parameter and its sub-parameter values with the given index from the specified device
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	See Table 57
Errors	302, 303, 304, 308, 309, 310, 311, 601 (see A.2), see 4.4.9
Success	200 OK
Request body	–
Response body	one of objects that are listed as options in the deviceParameterValues object list (see Table 69)

Table 69 – deviceParameterValues object list

deviceParameterValues object list			
object	Object defined in Table 72 – Object rawParameterValue		C[1]
object	Object defined in Table 71 – Object simpleParameterValue		C[2]
object	Object defined in Table 70 – Object complexParameterValue		C[3]
Note 1) If format is byteArray Note 2) If format is iodd and process data are of simple type, naming shall be according 4.4.5 Note 3) only available for indices or process data with complex data types			

Table 70 – Object complexParameterValue

object	**complexParameterValue**			
property	**type**	**value**	**remark**	**M/O/C**
{parameterName}	object	Object defined in Table 71 – Object simpleParameterValue	Complex type e.g. "Distance": {"value": 15 }, "Quality": {"value": 12 }	
Note 4) IODDname is based on rules see 4.4.5				

Table 71 – Object simpleParameterValue

object	**simpleParameterValue**			
property	**type**	**value**	**remark**	**M/O/C**
value	boolean, string,		Simple types e.g. "value":15.2	

	number			

674

675 **Table 72 – Object rawParameterValue**

676

object	rawParameterValue			
property	**type**	**value**	**remark**	**M/O/C**
value	array numbers		see 4.4.7	

677

678 **Example:**

679 Request (format byteArray)

```
GET /iolink/v1/devices/sensor34/parameters/65/value/?format=byteArray
```

680 Response

```
{
 "value":[0,156,125,25]
}
```

681

682 Request (format iodd)

```
GET /iolink/v1/devices/sensor34/parameters/65/value/?format=iodd
```

683 Response (simple type)

```
{
  "value":15.2
}
```

684

685 Request (format iodd)

```
GET /iolink/v1/devices/sensor34/parameters/65/value/?format=iodd
```

686 Response (complex type)

```
{
    "Distance": {"value": 15 },
    "Quality": {"value": 12 }
}
```

687

688 ### 5.7.11 GET /parameters/{index}/subindices/{subindex}/value

689 This function reads the value of a specific sub parameter with the given index and subindex
690 from the specified device. The requested format is indicated by a query string and can be
691 byteArray (default) or IODD based. For format “iodd” the IODD support is needed

692

693 **Table 73 – GET /parameters/{index}/subindices/{subindex}/value**
694

	Description
Description	reads a specific sub parameter value with the given index and subindex
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	See Table 57
Errors	302, 303, 304, 308, 309, 310, 311, 601 (see A.2), see 4.4.9
Success	200 OK
Request body	–
Response body	one of objects that are listed as options in the deviceParameterValues object list (see Table 69)

695

696

697 **Example**

698 Request (format byteArray)

```
GET /iolink/v1/devices/sensor34/parameters/25/subindices/3/value/?format=byteArray
```

699 Response

```
{
 "value":[0,156,125,25]
}
```

700

701 Request (format iodd)

```
GET /iolink/v1/devices/sensor34/parameters/25/subindices/3/value/?format=iodd
```

702 Response

```
{
  "value":15.2
}
```

703

704 ### 5.7.12 GET /parameters/{parameterName}/value

705 This function reads a specific parameter value with the given name .The response is also
706 given by IODD naming. The parameterName shall follow the rules of IODD based naming
707 rules (see 4.4.5). IODD support is needed.

708 **Table 74 – GET /parameters/{parameterName}/value**
709

	Description
Description	reads a specific parameter value with the given name
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	Format request parameters (see Table 57)
Errors	312 (see A.2), see 4.4.9
Success	200 OK
Request body	–

Response body	one of objects that are listed as options in the deviceParameterValues object list (see Table 69)

Example

Request (format byteArray)

```
GET /iolink/v1/devices/sensor34/parameters/nameOfVariable/value/?format=byteArray
```

Response

```
{
 "value":[0,156,125,25]
}
```

Request (format iodd)

```
GET /iolink/v1/devices/sensor34/parameters/nameOfVariable/value/?format=iodd
```

Response (simple type)

```
{
  "value":15.2
}
```

Request (format iodd)

```
GET /iolink/v1/devices/sensor34/parameters/nameOfVariable/value/?format=iodd
```

Response (complex type)

```
{
    "Distance": {"value": 15 },
    "Quality": {"value": 12 }
}
```

5.7.13 GET /parameters/{parameterName}/subindices/{subParameterName}/value

This function reads the content of a specific sub parameter value with the given name and subname from the specified device.

Table 75 – GET /parameters/{parameterName}/subindices/{subParameterName}/value

	Description
Description	reads the content of a specific sub parameter value with the given name and subname
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	Format request parameters (see Table 57)
Errors	312 (see A.2)), see 4.4.9
Success	200 OK
Request body	–

Response body	one of objects that are listed as options in the deviceParameterValues object list (see Table 69)

Example

Request (format byteArray)

```
GET
/iolink/v1/devices/sensor34/parameters/nameOfVariable/subindices/subName/value?format
= byteArray
```

Response

```
{
 "value":[0,156,125,25]
}
```

Request (format iodd)

```
GET
/iolink/v1/devices/sensor34/parameters/nameOfVariable/subindices/subName/value?format
= iodd
```

Response

```
{
 "value": 152
}
```

5.7.14 POST /parameters/{index}/value

This function writes the parameter with the given index to the specified device. The format is given within the body

Table 76 – POST /parameters/{index}/value

	Description
Description	writes the parameter with the given index
System Behaviour	if there are no restrictions process data will be updated
Path parameters	–
Query parameters	–
Errors	(see A.2),), see 4.4.9
Success	204 No Content
Request body	one of objects that are listed as options in the deviceParameterValues object list (see Table 69)
Response body	–

745 **Example**

```
POST /iolink/v1/devices/sensor34/parameters/125/value

{
  "value":[0,156,125,25]
}
```

746

747 **Example**

```
POST /iolink/v1/devices/sensor34/parameters/125/value

{
 "value": 15
}
```

748

749 **Example**

```
POST /iolink/v1/devices/sensor34/parameters/125/value

{
   "Distance": {"value": 15},
   "Quality": {"value": 12}
}
```

750

751 ### 5.7.15 POST /parameters/{parameterName}/value

752 This function writes the parameter with the given name to the specified device. The format is
753 given within the body

754 **Table 77 – POST /parameters/{parameterName}/value**

755

	Description
Description	writes the parameter with the given name to the specified device
System Behaviour	if there are no restrictions parameter will be updated
Path parameters	–
Query parameters	–
Errors	312 (see A.2)), see 4.4.9
Success	204 No Content
Request body	one of objects that are listed as options in the deviceParameterValues object list (see Table 69)
Response body	–

756

757 **Example**

```
POST /iolink/v1/devices/sensor34/parameters/variableName/value

{
  "value":[0,156,125,25]
}
```

758

759

```
POST /iolink/v1/devices/sensor34/parameters/variableName/value

{
  "value": 15
}
```

```
POST /iolink/v1/devices/sensor34/parameters/variableName/value

{
  "Distance": {"value": 15},
  "Quality": {"value": 12}
}
```

5.7.16 POST /parameters/{index}/subindices{subindex}/value

Write the parameter with the given index and subindex to the specified device. The format is given within the body.

Table 78 – POST /parameters/{index}/subindices{subindex}value

	Description
Description	writes the sub parameter with the given name
System Behaviour	if there are no restrictions parameter will be updated
Path parameters	–
Query parameters	–
Errors	302, 303, 304, 308, 309, 205, 206, 501 (see A.2),), see 4.4.9
Success	204 No Content
Request body	one of objects that are listed as options in the deviceParameterValues object list (see Table 69)
Response body	–

Example

```
POST /iolink/v1/devices/sensor34/parameters/35/subindices/4/value

{
  "value":[0,156,125,25]
}
```

5.7.17 POST /parameters/{parameterName}/subindices{subParameterName}/value

This function writes the sub-parameter with the given parameter name and subParameterName to the specified device.

Table 79 – POST /parameters/{ parameterName }/subindices{ subParameterName }value

	Description
Description	writes the sub-parameter with the given parameter name and subParameterName. The requested format is implicit defined within the body

System Behaviour	if there are no restrictions parameter will be updated
Path parameters	–
Query parameters	–
Errors	205, 206, 302, 303, 304, 308, 309, 312, 501 (see A.2), see 4.4.9
Success	204 No Content
Request body	one of objects that are listed as options in the deviceParameterValues object list (see Table 69)
Response body	–

777

778 **Example**

```
POST /iolink/v1/devices/sensor34/parameters/35/subindices/4/value

{
  "value": 152
}
```

779

780 ### 5.7.18 POST /blockparametrization

781 Writing or reading one or more parameters by using the block parameterization method see
782 [1]. The request format can be byteArray or iodd based.

783 **Table 80 – POST /blockparametrization**

784

	Description
Description	Writing or reading one or more parameters as a block (consitent)
System Behaviour	if there are no restrictions for writing the objects will be modified
Path parameters	–
Query parameters	Format request parameters see Table 57
Errors	(see A.2), see 4.4.9
Success	200 OK for “read” 204 No Content for “write”
Request body	Object defined in Table 81 – Object BlockparametrizationRequest
Response body	Array of Object defined in Table 82 – Object BlockparametrizationReadResponse

785

786 **Table 81 – Object BlockparametrizationRequest**

787

object	**BlockparametrizationReadRequest**			
property	**type**	**value**	**remark**	**M/O/C**
direction	string	**Enumeration:** “read” “write”		M
parameters	objects	Object defines in Table 83 – Object ParameterAddress	Addresses all parameters or subParameters	M

788

789 **Table 82 – Object BlockparametrizationReadResponse**
790
791

object	BlockparametrizationReadResponse			
property	**type**	**value**	**remark**	**M/O/C**
	object	Object defines in Table 83 – Object ParameterAddress		M
	object	Object defined in Table 84 – Object ParameterValue		M

792

793 **Table 83 – Object ParameterAddress**
794
795

object	ParameterAddress			
property	**type**	**value**	**remark**	**M/O/C**
`address`	object	Object defined in Table 85 – Object IndexAddress		C[1]
`address`	object	Object defined in Table 86 – Object SubIndexAddress		C[1]
Note 1: Either one one of them has to be used				

796

797 **Table 84 – Object ParameterValue**
798
799

object	ParameterAddress			
property	**type**	**value**	**remark**	**M/O/C**
`value`	array numbers	See 4.4.7		C[1]
`value`	boolean, string, number		`Simple types`	C[2]
`value`	object	Object defined in Table 70 – Object complexParameterValue	`Complex types`	C[3]
Note 1: this is format byteArray Note 2: this is for format iodd Note 2: this is for format iodd but not for subindex/subparameter access				

800

801

802

803 **Table 85 – Object IndexAddress**
804
805

object	IndexAddress			
property	**type**	**value**	**remark**	**M/O/C**
`index`	number	integer	Number of index	C[1]

			e.g. {"index": 123}	
parameterName	string	{parameterName}	Name of Index {"parameterName": "Application_tag"}	C[2]
Note 1: this is format byteArray Note 2: this is format iodd if iodd is supported				

Table 86 – Object SubIndexAddress

object	SubIndexAddress			
property	**type**	**value**	**remark**	**M/O/C**
index	number	integer		C[1]
subIndex	number	integer		C[1]
parameterName	string	{parameterName}		C[2]
parameterSubName	string	{parametersubName}		C[2]
Note 1: this is format byteArray Note 2: this is format iodd if iodd is supported				

Example:

Read Request (format byteArray)

```
POST /iolink/v1/devices/sensor34/blockparametrization/?format=byteArray

{
    "direction": "read",
    "parameters": [
      {
        "address": {"index": 123}
      },
      {
        "address": {"index": 233,"subIndex": 2}
      }
    ]
}
```

Response (request format is byteArray)

```
{
  [
    {
      "address": {"index": 123},
      "value": [15,232,22]
    },
    {
      "address": {"index": 233,"subIndex": 2},
      "value": [23,149,206]
    }
  ]
}
```

Read Request (format is iodd)

```
POST /iolink/v1/devices/sensor34/blockparametrization/?format=iodd

{
    "direction": "read",
    "parameters": [
      {
        "address": {"parameterName": "applicationTag"}
      },
      {
        "address": {"{"parameterName": "hystersis","subParametername": "channelB" }
      }
    ]
}
```

Response

```
{
 [
    {
      "address": {"parameterName": "Application_tag"},
      "value": {"value": "Level 2, row 3"}
    },
    {
      "address": {"parameterName": "Hysteresis","subParameterName": "Channel_B"},
      "value": {"value": 123}
    }
  ]
}
```

Write Request (format is byteArray)

```
POST /iolink/v1/devices/sensor34/blockparametrization/?format=byteArray

{
    "direction": "write",
    "parameters": [
      {
        "address": {"index": 123},
        "value": [15,232,22]
      },
      {
        "address": {"index": 233,"subIndex": 2},
        "value": [23,149,206]
      }

    ]
}
```

Write Request (format iodd)

```
POST /iolink/v1/devices/sensor34/blockparametrization/?format=iodd

{
    "direction": "write",
    "parameters": [
      {
        "address": {"parameterName": "Application_tag"},
        "value": {"value": "Level 2, row 3"}
      },
      {
        "address": {"parameterName": "Hysteresis","subParameterName": "Channel_B"},
        "value": {"value": 123}
```

```
      }
    ]
}
```

5.7.19 GET /events

This is for reading the EventLog for a specific device and similar to the reading of the EventLog but filtered for this specific device (origin is device).

Table 87 – GET /events

	Description
Description	This is for reading the EventLog for a specific device
System Behaviour	Nothing will be changed or modified.
Path parameters	–
Query parameters	Device Event log query parameters (see Table 88)
Errors	(see A.2), see 4.4.9
Success	200 OK
Request body	–
Response body	Object defined in Table 20 – Object GatewayEventLog

Table 88 – DeviceEvent log query parameters

query parameter	**values**	**Remark**
top	1…n	Delivers the oldest n events of the event Log. *top* is mutually exclusive to *bottom*.
bottom	1…m	Delivers the latest n events of the event lOG. *bottom* is mutually exclusive to *top*.

Example:

Request

```
GET /iolink/v1/devices/Temp_sensor_1/events
```

Response

```
[
  {
    "time": "2018-05-18T07:31:54.123Z",
    "severity": "WARNING",
    "origin": {
      "master": 1,
      "port": 1,
      "device": "Temp_sensor_1"
    },
    "message": {
      "code": 16912,
      "mode": "APPEARED",
      "text": "Device temperature over-run – Clear source of heat"
    }
  }
]
```

5.8 IODD

Within the feature IODD the IODDs are stored (uploading) on the gateway.

For a specific device (referenced by VendorID and Device ID) only one version of an IODD shall be stored.

All request marked as (M) are mandatory if the feature IODD is supported

Table 89 – Resources for IODDs

Resources	clause	HTTP Method	remark	M/O/C
/iodds	5.8.1	GET	Get a list of all IODD (representations) that are available on the gateway	M
/iodds	5.8.2	POST	Update or create an IODD	M
/iodds/vendors/{vendorId}/devices/{deviceId}/revisions/{ioLinkRevision}	5.8.4	GET	Get a specific IODD	O
/iodds/vendors/{vendorId}/devices/{deviceId}/revisions/{ioLinkRevision}	5.8.3	DELETE	Delete a specific IODD representation	M

5.8.1 GET /iodds

Get a list of all IODDs (representations) that are available on the gateway.

Table 90 – GET /iodds

	Description
Description	Get a list of all IODDs (representations) that are available on the gateway.
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	–
Errors	301 (see A.2), see 4.4.9
Success	200 OK
Request body	–
Response body	Array of objects defined in Table 91 – Object IODDIdentification

Table 91 – Object IODDIdentification

Object	IODDIdentification			
property	**type**	**Value**	**description**	**M/O/C**
`vendorId`	Integer		see [1]	M
`deviceId`	Integer		see [1]	M
`version`	string		see [1]	M
`releaseDate`	string		see [1]	M
`ioLinkRevision`	string	**Enumeration:** “1.0” “1.1”	see [1]	M

Example:

Request

```
GET /iolink/v1/iodds
```

Response

```
 [
  {
    "vendorId": 1234,
    "deviceId": 4567,
    "version": "4.3",
    "releaseDate": "2018-01-02",
    "ioLinkRevision": "1.1"
  },
  {
    "vendorid": 4321,
    "deviceId": 8765,
    "version": "2.1",
    "releaseDate": "2015-01-02",
    "ioLinkRevision": "1.1"
  }
]
```

5.8.2 POST /iodds

Update or create an IODD on the gateway. Application Type is XML

Table 92 – POST /iodds

	Description
Description	Update or create an IODD on the gateway
System Behaviour	if there are no restrictions process data will be updated
Path parameters	–
Query parameters	–
Errors	602, 603, 604, 605 (see A.2), see 4.4.9
Success	204 No Content
Request body	–
Response body	–

5.8.3 DELETE /iodds

Delete a specific IODD on the gateway. The IODD is addressed by path parameters.

Table 93 – DELETE /iodds

	Description
Description	deletes an IODD on the gateway
System Behaviour	if there are no restrictions process data will be updated
Path parameters	/vendors/{vendorId}/devices/{deviceId}/revisions/{ioLinkRevision}
Query parameters	–
Errors	301, 305, 306, 601(see A.2), see 4.4.9

Success	204 No Content
Request body	–
Response body	–

Example:

Request

```
DELETE /iolink/v1/iodds/vendors/26/devices/34654/revisions/1.1
```

5.8.4 GET /iodds

Read a specific IODD (XML file) from the gateway. The IODD is addressed by path parameters.

Table 94 – GET /iodds

	Description
Description	Read a specific IODD (XML file) from the gateway.
System Behaviour	Nothing will be changed or modified
Path parameters	/vendors/{vendorId}/devices/{deviceId}/revisions/{ioLinkRevision}
Query parameters	–
Errors	301,601 (see A.2), see 4.4.9
Success	200 OK
Request body	–
Response body	–

Example:

Request

```
GET /iolink/v1/iodds/vendors/26/devices/34654/revisions/1.1
```

5.9 MQTT

This section is related to the configuration of the MQTT Publisher defining publishing topics and configurations for addressing the MQTT broker. All is referenced to MQTT Version 3.1.1

All request marked as (M) are mandatory if the MQTT feature is supported.

Table 95 – Resources MQTT configuration

URLs /mqtt	**clause**	**HTTP Method**	**remark**	**M/O/C**
/configuration	5.9.1	GET	Read the MQTT configuration of the gateway	M
/configuration	5.9.2	POST	Update the MQTT configuration of the gateway	M
/topics	5.9.3	GET	Get the list of MQTT topics	M

/topics	5.9.4	POST	Create a new MQTT topic	M
/topics/{topicId}	5.9.5	DELETE	Delete a specific MQTT topic	M
/topics/{topicId}	5.9.6	GET	Get one specific MQTT topic	M

5.9.1 GET /mqtt/configuration

Read the MQTT configuration of the gateway by getting the object “MQTT Configuration” . For details of MQTT protocol and client configuration see [6]

Table 96 – GET /mqtt/configuration

	Description
Description	Read the MQTT configuration
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	–
Errors	101, 102, 103 (see A.2)
Success	200 OK
Request body	–
Response body	Object defined in Table 97 – Object MQTTConfiguration

Table 97 – Object MQTTConfiguration

object	**MQTT Base Configuration**			
property	**type**	**value**	**description**	**M/O/C**
`brokerAddress`	string		url of the broker	M
`username`	string			M
`password`	string			M
`lastWill`	object	Object defined in Table 98 – Object LastWill		M
`keepAliveTime`	number			M

Table 98 – Object LastWill

object	**Last Will**			
property	**type**	**Value**	**description**	**M/O/C**
`topic`	string			M
`message`	string			M
`qoS`	string	**Enumeration:** “0_ONLY_ONCE” “1_AT_LEAST_ONCE” “2_EXACTLY_ONCE”		M
`retain`	boolean	true, false		M

Example:

905 Request

```
GET /iolink/v1/mqtt/configuration
```

906 Response

```
{
  "brokerAddress": "192.168.2.1/mqttbroker",
  "username": "iolink_json",
  "password": "1234",
  "lastWill": {
    "topic": "my_temperature_sensor",
    "message": " Process data transfer stopped ",
    "qoS": "0_ONLY_ONCE",
    "retain": true
  },
  "keepAliveTime": 0
}
```

907

908 ### 5.9.2 POST /mqtt/configuration

909 Write the MQTT configuration to the gateway by updating the object “MQTT Configuration”

910

911 **Table 99 – POST /mqtt/configuration**

912

	Description
Description	writes the MQTT configuration to the gateway
System Behaviour	if there are no restrictions the configuration will be updated
Path parameters	–
Query parameters	–
Errors	101,102, 103, 150, 201, 202, 203, 204, 205, 206 (see A.2)
Success	200 OK
Request body	Object defined in Table 97 – Object MQTTConfiguration
Response body	–

913

914 **Example:**

915 Request

```
POST /iolink/v1/mqtt/configuration
{
  "brokerAddress": "192.168.2.1/mqttbroker",
  "username": "iolink_json",
  "password": "1234",
  "lastWill": {
    "topic": "my_temperature_sensor",
    "message": " Process data transfer stopped ",
    "qoS": "0_ONLY_ONCE",
    "retain": true
  },
  "keepAliveTime": 0
}
```

916

917 ### 5.9.3 GET /mqtt/topics

918 Get the list of MQTT topics. Supported MQTT topics are of type processData or event.

919 **Table 100 – GET /mqtt/topics**
920

	Description
Description	Read the list of MQTT topics
System Behaviour	Nothing will be changed or modified
Path parameters	–
Query parameters	–
Errors	101, 102, 103 (see A.2)
Success	200 OK
Request body	–
Response body	Array of objects defined in Table 101 – Object MQTTtopic

921

922

923 **Table 101 – Object MQTTtopic**
924

object	**MQTTtopic**			
property	**type**	**value**	**description**	**M/O/C**
topicId	number		The topic number as a reference	M[2]
topicName	string			M
qoS	string	**Enumeration:** “0_ONLY_ONCE” “1_AT_LEAST_ONCE” “2_EXACTLY_ONCE”		M
deviceName	string	{deviceName}		M
processData	object	MQTT ProcessData topic see Table 102		C[1]
event	object	null		C[1]
Note 1: Either one of processData or event Note 2: This is not applicable for POST				

925

926 **Table 102 – Object MQTT ProcessDataTopic**
927

object	**MQTT ProcessDataTopic**			
property	**type**	**Value**	**description**	**M/O/C**
direction	string	**Enumeration:** “measure” “control” “measurecontrol”		M
format	string	**Enumeration:** “byteArray” “iodd”		M
interval	object	“CycleTime” see Table 108		C[1]
onChange	Boolean	true, false		C[1]
Note 1: Either one of interval or onChange				

928

929 **Example**:

930 Request

```
GET /iolink/v1/mqtt/topics
```

931 Response

```
[
  {
    "topicId": 1,
    "topicName": "PD inPOST",
    "qos": "At least once",
    "qos": "1_AT_LEAST_ONCE",
    "deviceName": "DT35",
    "processData": {
      "direction": "measure",
      "format": "iodd",
      "interval": {
        "value": 10,
        "unit": "ms"
      }
    }
  },
  {
    "topicId": 2,
    "topicName": "Event",
    "qos": "2_EXACTLY_ONCE",
    "deviceName": "TAD081",
    "event": null
  },
  {
    "topicId": 3,
    "topicName": "PD",
    "qos": "0_ONLY_ONCE",
    "deviceName": "BNI_IOL",
    "processData": {
      "direction": "measurecontrol",
      "format": "iodd",
      "onChange": true
    }
  }
]
```

932
933

5.9.4 POST /mqtt/topics

935 Create one new MQTT topic on the gateway

936

937 **Table 103 – POST /mqtt/topics**

938

	Description
Description	Create a new MQTT topic on the gateway
System Behaviour	if there are no restrictions the object will be modified
Path parameters	–
Query parameters	–
Errors	102 (see A.2)
Success	200 OK
Request body	Object defined in Table 101 – Object MQTTtopic

Response body	–

Example:

Request

```
POST /iolink/v1/mqtt/topics
{
  "qos": "1_AT_LEAST_ONCE",
  "deviceName": "DT35",
  "processData": {
    "direction": "measure",
    "format": "iodd",
    "interval": {
      "value": 10,
      "unit": "ms"
    }
  }
}
```

5.9.5 DELETE /mqtt/topics/{topicID}

Delete a specific MQTT topic on the gateway referenced by the topic ID. The topic ID is addressed by a path parameter.

Table 104 – DELETE /mqtt

	Description
Description	Deletes a MQTT topic
System Behaviour	if there are no restrictions the topic will be deleted
Path parameters	/topics/{topicID}
Query parameters	–
Errors	101,102, 103, 150 (see A.2)
Success	204 No Content
Request body	–
Response body	–

Example:

Request

```
DELETE /iolink/v1/mqtt/topics/2
```

5.9.6 GET /mqtt/topics/{topicId}

Read one specific MQTT topic on the gateway referenced by the topic ID.

Table 105 – GET /mqtt/topics

	Description
Description	Read one specific MQTT topic
System Behaviour	Nothing will be changed or modified
Path parameters	/{topicID}
Query parameters	–
Errors	101, 102, 103 (see A.2)

Success	200 OK
Request body	–
Response body	Object defined in Table 101 – Object MQTTtopic

Example:

Request

```
GET /iolink/v1/mqtt/topics/2
```

Response

```
{
   {
    "topicId": 2,
    "topicName": "PD inPOST",
    "qos": "1_AT_LEAST_ONCE ",
    "deviceName": "DT35",
    "processData": {
      "direction": "measure",
      "format": "iodd",
      "interval": {
        "value": 10,
        "unit": "ms"
      }
    }
}
```

6 MQTT topics format

This section describes the publishing format for the topic. Topics can be either be of type processData or event.

For process data the topic format is as a object defined in Table 59 – Object ProcessDataInOut

Example: (Process data topic)

```
{
 "measure": {
    "ioLink": {
       "valid": true,
       "value": {
          "Distance": 55,
          "Quality": 12
        }
    },
    "pin2Value": true
  }
}
```

For Event_Log entries the topic format is as a object defined Table 20 – Object GatewayEventLog. Only one EventLog entry shall be published at the time the event was entered into the EventLog.

Example: (Event Log entry topic)

```
  {
    "time": "2018-05-18T07:31:54.123Z",
```

```
    "severity": "WARNING",
    "origin": {
      "master": 1,
      "port": 1,
      "device": "Temp_sensor_1"
    },
    "message": {
      "code": 16912,
      "mode": "APPEARED",
      "text": "Device temperature over-run – Clear source of heat"
    }
  }
```

Annex A
(normative)

Status Codes and Errors on HTTP

A.1 HTTP Status Codes

Each request on HTTP can response with or without an error indicated bey the status code.

Table 106 – HTTP Status codes

Status Code		description
200	OK	The request completed successfully
201	Created	A new resource has been created successfully
204	No Content	Response code for POST requests without content in response and the DELETE.
400	Bad Request	
403	Forbidden	The user is not permitted to perform the requested operation
404	Not Found	The requested resource did not exist
500	Server error	

A.2 JSON Errors

Each negative response indicated by a HTTP status code other than 200 is added by an individual JSON Error object.

Table 107 – JSON Error codes

Error Code	HTTP Status code	JSON Message	Notes
General errors			
101	500	Internal server error	
102	500	Internal communication error	
103	404	Operation not supported	
104	400	Action locked by another client	Note 1
150	403	Permission denied	Note 2
JSON parsing errors			
201	400	Parsing failed (error while parsing the incoming JSON value)	
202	400	Invalid data (given for the value, e.g. malformed IP address)	
203	400	Invalid JSON data type (e.g. string instead of numbers)	
204	400	Unknown enumeration value	
205	400	Value out of range (exceeds the minimum or maximum value)	
206	400	Out of bounds (an array/string was accessed exceeding its maximum length)	
207	400	Duplicated deviceName	
Resource access errors			
301	404	Resource not found (e.g. wrong URL).	
302	404	masterNumber not found	
303	404	portNumber not found	
304	404	deviceName not found	

305	400	Incorrect query parameter name	
306	400	Incorrect query parameter value	
307	400	Port is not configured in IOLINK_MANUAL or IOLINK_AUTOSTART mode.	
308	404	IO-Link Device is not accessible (not connected or communication error)	
309	404	IO-Link parameter not found	
310	404	IO-Link parameter access not supported by the device	
311	400	IO-Link parameter access error	Note 3
312	404	IO-Link parameter name is not unique Please use the [name]_[index] format	
Data Storage errors			
401	400	Mismatch between configured device and data storage meta data	
Process Data handling errors			
501	400	Writing processdata to Pin 2 is not possible. Pin 2 is not configured as DIGITAL_OUTPUT	
502	400	Writing processdata to Pin 4 is not possible. Pin 4 is not configured as DIGITAL_OUTPUT	
503	400	Attached IO-Link device has no output process data	
IODD errors			
601	400	IODD is not available.	
602	500	Unable to upload IODD XML file. Insufficient memory space	
603	400	Uploaded file is no valid IODD XML. Upload rejected	
604	400	Uploaded file is no valid IODD XML. CRC check failed. Upload rejected	
605	400	Uploaded file is no valid IODD XML. Parsing failed. Upload rejected	
Data content errors			
701	400	Incomplete data set	
702	400	Data set not applicable (not supported), the whole data set is rejected	
703	400	Incompatible data set combination, the whole data set is rejected	

Note 1: (e.g. by a fieldbus controller)

Note 2: (e.g. due to user management)

Note 3: The additional iolinkErrorCode and iolinkErrorMessage fields contain the IO-Link error code and the incident text from the ErrorTypes table

991

Annex B
(normative)

JSON base objects

B.1 General JSON objects

B.1.1 Cycle time object

Table 108 – Object CycleTime

object	Cycle Time			
property	**type**	**Value**	**description**	**M/O/C**
value	number			M
unit	string	“ms”	Milliseconds	M

Example:

```
{
    "value": 2.3,
    "unit": "ms"
}
```

B.1.2 Error object

Table 109 – Object Error

object	Error			
property	**type**	**Value**	**description**	**M/O/C**
msgCode	number		Integer value	M
message	string	See A.2, See [1]		M

Example:

```
{
    "msgCode": 150,
    "message": "Permission denied"
}
```

B.1.3 Power Supply object

Table 110 – Object PowerSupply

object	PowerSupply			
property	**type**	**Value**	**description**	**M/O/C**
value	number			M

`unit`	string	“A”	SI unit Ampere	M

Example:

```
{
    "value": 0.3,
    "unit": "A"
}
```

Bibliography

[1] IO-Link Community, *IO-Link Interface and System*, V1.1.3, July 2019, Order No. 10.002

[2] IO-Link Community, *IO Device Description (IODD), Version 1.1, Order No. 10.012*

[3] https://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf

[4] DIN ISO 8601 https://www.iso.org/iso-8601-date-and-time-format.html

[5] RFC 5424 Syslog Protocol https://tools.ietf.org/html/rfc5424

[6] www.oasis-open.org

[7] IETF RFC 4648 https://www.ietf.org/rfc/rfc4648.txt

IO-Link Community
c/o PROFIBUS Nutzerorganisation e.V.
Haid-und-Neu-Str. 7
76131 Karlsruhe
Germany
Phone: +49 (0) 721 / 96 58 590
Fax: +49 (0) 721 / 96 58 589
e-mail: info@io-link.com
http://www.io-link.com/