JSON

Integration for IO-Link

Version 1.0.0
Mar 2020

Order No: 10.222

File name: JSON_Integration_10222_V100_Mar20

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.

Disclaimer:

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^{TM}[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^{TM} 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^{TM} 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 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.

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

Device
single passive peer to a Master such as a sensor or actuator.

NOTE: Uppercase "Device" is used for SDCI equipment, while lowercase "device" is used in a generic manner

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

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

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

masterNumber
This is the number of a specific Master within the gateway.

Port
Communication interface of the Master to one Device.

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

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

URL
This is a Uniform Resource Locator.

3.2 Symbols and abbreviated terms

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 4.5.1
SMI	standardized Master interface see [1]

3.3 Conventions

3.3.1 Placeholders

Strings that are embraced by {} are placeholders for variables. Placeholders within an URL are path parameters.

NOTE: This convention is not applicable for JSON objects.

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. See Figure 1 – Physical Gateway models. 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).

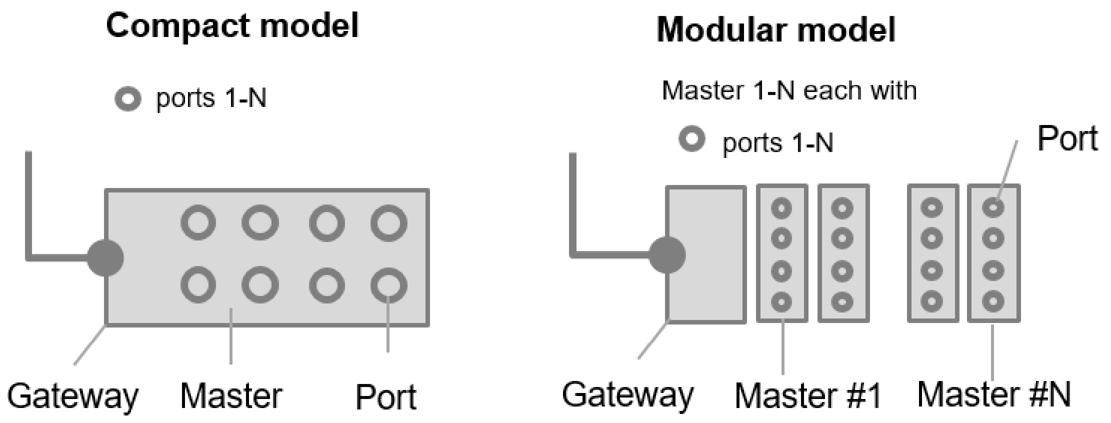

Figure 1 – Physical Gateway models

4.2 Features

This specification supports two features “IODD support” and “MQTT support”.

IODD support allows addressing and representation of data by names and data types defined in the IODD.

MQTT support allows configuration of an MQTT client (publisher) and the connection to an MQTT broker.

Both features are optional.

4.3 Security

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

4.4 Device data and layer model

This device layer model (including Gateway, IO-Link Master and Devices) see Figure 2 is used to structure the REST API described in this specification. This layer model comprises compact modules containing one Master as well as modular devices with N Masters.

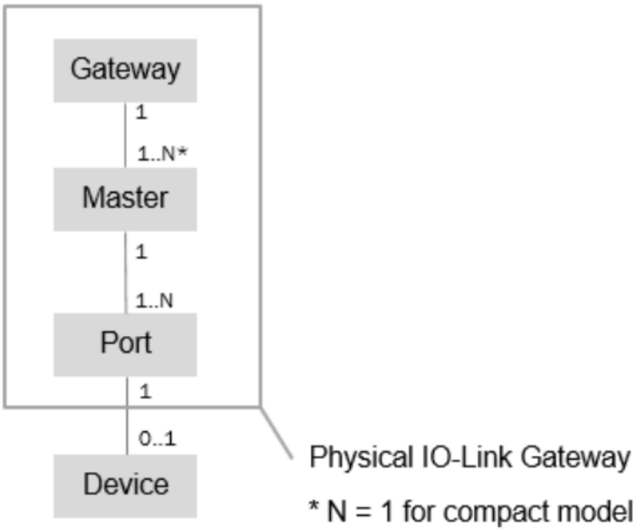

Figure 2 - Device layer model

Each layer has resources which are addressed by a url path. The model shows that a physical Gateway may have multiple gateway applications and one or more Masters. Each Master has one or more Ports and on each Port no or one device is connected.

4.5 General rules

Vendor-specific extensions for REST API commands and JSON key-value pairs are allowed.

A JSON for IO-Link Server shall return an error on all REST API commands and JSON key-value pairs that are not supported.

4.5.1 Usage of M/O/C

- It is mandatory to implement the handling of objects and key value pairs that are marked with “M”.
- When sending an 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.

4.5.2 Master numbering

Addressing of IO-Link Masters within a Gateway starts with number 1 for the first Master.

4.5.3 Port numbering

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

4.5.4 Device Naming

Devices are accessed by names (aliases). Following rules for the Device naming apply:

1. The default device alias is created based on Master number and Port number in the format `master{masterNumber}port{portNumber}`.
2. The default device alias 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. The minimum length of the device name is 1 character and the maximum is 32.

Examples:

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

4.5.5 Access rights

All requests for writing data may be blocked due to right restrictions. Also resetting or rebooting may not be 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.

4.5.6 Naming based on IODD

It is optionally also possible to access IO-Link data (comprising IO-Link Process Data and IO-Link parameters) by name rather than by index and subindex. See [2].

Constraints for strings in keys of JSON key-value-pairs (see [9]) 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: Only alphanumeric characters and underscores are allowed. All other characters are replaced by "_".

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

Rule 4: If there are duplicate 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.

Rule 6: The naming of ArrayT elements is “`element_{subindex}`”.

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)			

4.5.7 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 (see [1])	JSON data type
BooleanT	Boolean
StringT	String
ArrayT	Array
OctetStringT	see Byte array conversion 4.5.8
IntegerT, UintegerT	Number
Float32T	Number
RecordT	Object
TimeSpanT	String
TimeT	String

4.5.8 Byte array conversion

This is the description on 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 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]`

4.5.9 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: Each italic letter in the following format definitions is to be replaced by one digit. The square brackets indicate that an element is optional.

The format for absolute time is `YYYY-MM-DDThh:mm:ss.fZ`

Time format for relative time is `P[YY][MM][WW][DD][T[hH][mM][s[.f]S]]`

Examples:

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

4.5.10 Error Behavior

While processing HTTP requests 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.

The 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	**Example**	**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 (details see [8]).

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`	0	Address the Gateway.	M
`/masters`	5.5	Get all available masterNumber keys and identification information.	M
`/masters/{masterNumber}`	5.5	Address a specific master.	M
`/masters/{masterNumber}/ports`	5.6.1	Get all available portNumber keys.	M
`/masters/{masterNumber}/ports/{portNumber}`	5.6.1	Address a specific port of a specific Master.	M
`/devices`	5.7	Address all Devices of all Masters.	M
`/devices/{deviceAlias}`	5.7.3	Address a 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 identification of the 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 identification of the Gateway.

Table 8 – GET /identification

	Description
Description	Read the indentification of the Gateway.
System Behavior	Nothing will be changed or modified.
Path	/gateway/identification
Query parameters	–
Errors	(see A.2), (see 4.5.10)
Success	HTTP 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
`serialNumber`	string		Vendor specific	O
`productId`	string		Vendor specific	O
`vendorName`	string		Vendor specific	M
`productName`	string		Vendor specific	O
`hardwareRevision`	string		Vendor specific	O
`firmwareRevision`	string		Vendor specific	O
`productInstanceUri`	string		Vendor specific	O

Example:

Request

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

Response

```
{
  "macAddress": "00:02:72:CE:A6:49",
  "serialNumber": "C134A746",
  "productID": "TMP34Z",
  "vendorName": "SensorCompany",
  "productName": "FlowSensor34",
  "hardwareRevision": "V3.45",
  "firmwareRevision": "V1.30",
  "productInstanceUri": "sensor.tmp.23.com"
}
```

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 Behavior	Nothing will be changed or modified.
Path	/gateway/capabilities
Query parameters	–
Errors	(see A.2), see 4.5.10
Success	HTTP 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 active configuration of the IO-Link Gateway. The Gateway may support multiple IPv4 interfaces.

Table 12 – GET /configuration

	Description
Description	Read the actual active configuration of the IO-Link Gateway.
System Behavior	Nothing will be changed or modified.
Path	/gateway/configuration
Query parameters	–
Errors	(see A.2), (see 4.5.10)
Success	HTTP 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 objects	Objects defined in Table 14 – Object NetworkInterfaceConfiguration		M

Table 14 – Object NetworkInterfaceConfiguration

Object	**NetworkInterfaceConfiguration**			
Property	**Type**	**Value**	**Description**	**M/O/C**
`ipConfiguration`	string	**Enumeration:** “MANUAL” “DHCP” “AUTO_IP” “DCP”	MANUAL: Assignment of the IP address by other device specific means DHCP: RFC 2131 defines the “Dynamic Host Configuration Protocol”, allowing automatic assignment of IP addresses. AUTO_IP: RFC 3927 defines “Dynamic Configuration of IPv4 Link-Local Addresses", allowing fully-automatic assignment of Link-Local IP addresses. DCP: PROFINET defines the “Discovery and Configuration Protocol”, a link-layer protocol that allows the manual assignment of IP addresses.	M[2]
`ipAddress`	string		e.g. 192.168.1.13	C[1]
`subnetMask`	string		e.g. 255.255.255.0	C[1]
`standardGateway`	string		e.g. 192.168.1.1	C[1]
NOTE 1 Only allowed for POST if `ipConfiguration` is “MANUAL” and Mandatory for GET. NOTE 2 For POST at least one of the methods shall be implemented				

Example:

Request

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

Response

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

5.4.4 POST /configuration

Write configuration data to the gateway.

Table 15 – POST /configuration

	Description
Description	Write configuration data to the gateway.
System Behavior	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	/gateway/configuration
Query parameters	–
Errors	(see A.2), (see 4.5.10)
Success	HTTP 204 - No Content
Request body	Object defined in Table 13 – Object GatewayConfiguration
Response body	–

Example:

Request (changing an IPv4 interface)

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

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

5.4.5 POST /reset

Invoke 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	Invoke a reset of the IO-Link Gateway.
System Behavior	If there are no restrictions the response will be given and following the Gateway will be reset.
Path	/gateway/reset
Query parameters	–
Errors	(see A.2), (see 4.5.10)
Success	HTTP 204 - No Content
Request body	–
Response body	–

Example:

Request

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

5.4.6 POST /reboot

Invoke 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	Invoke a reboot of the IO-Link Gateway.
System Behavior	If there are no restrictions the response will be given and following the Gateway will reboot.
Path	/gateway/reboot
Query parameters	–
Errors	(see A.2), (see 4.5.10)
Success	HTTP 204 - No Content
Request body	–
Response body	–

Example:

Request

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

5.4.7 GET /events

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

Table 18 – GET /events

	Description
Description	Read 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 Behavior	Nothing will be changed or modified.
Path	/gateway/events
Query parameters	Event log query parameters (see Table 19)
Errors	(see A.2) , (see 4.5.10)
Success	HTTP 200 - OK
Request body	–
Response body	Object defined in Table 20 – Object GatewayEventLog

Table 19 – Event log query parameters

Query parameter	**Values**	**Remark**
`origin`	**Enumeration:**	Default is ALL

	“ALL” “GATEWAY” “MASTERS” “PORTS” “DEVICES”	
`masterNumber`	{masterNumber}	masterNumber is only applicable with origin=MASTERS and with origin=PORTS
`portNumber`	{portNumber}	*portNumber* is only applicable with origin=PORTS
`deviceAlias`	{deviceAlias}	*deviceAlias* 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…n	Delivers the latest n events of the event buffer. *bottom* is mutually exclusive to *top*.

Table 20 – Object GatewayEventLog

Object	**Gateway EventLog**			
Property	**Type**	**Value**	**Description**	**M/O/C**
`time`	string	Relative or absolute time as defined in 4.5.9	When the event was logged	M
`severity`	string	**Enumeration:** “EMERGENCY” “ALERT” “CRITICAL” “ERROR” “WARNING” “NOTICE” “INFO” “DEBUG”	Indicates the severity of the message. The severity is decreasing from EMERGENCY to DEBUG. Reference is Syslog Protocol RFC5424 see [5] IO-Link EventTypes shall be mapped as following: "NOTIFICATION" to "NOTICE", "WARNING" to "WARNING", "ERROR" to "ERROR". All severity level can also occur on gateway and master event level.	M
`origin`	object	Object defined in Table 21 – Object Origin		M
`message`	object	Object defined in Table 22 – Object Message		M

Table 21 – Object Origin

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]
`deviceAlias`	string	{deviceAlias}		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 `deviceAlias` has to be used if the event comes from the Device.

320

321 **Table 22 – Object Message**

322

Object	**Message**			
Property	**Type**	**Value**	**Description**	**M/O/C**
`code`	number	Codes according to [1] Annex D shall be used if the origin is device or port		C[1]
`mode`	string	Codes according to [1] 8.2.2.11 shall be used if the origin is device or port	SINGLESHOT APPEARS DISAPPEARS	C[1]
`text`	string	Message. (For IO-Link event code related messages (defined in [1]) shall be used)		C[1]

NOTE 1 For IO-Link events code and mode are mandatory, text is optional. For all other events text is mandatory

323

324 **Examples:**

325 Example 1: Reading all events of the Gateway

326 Request

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

327 Response

```
[
  {
    "time": "2018-05-18T07:31:54.123Z",
    "severity": "WARNING",
    "origin": {
      "masterNumber": 1,
      "portNumber": 1,
      "deviceAlias": "Temp_sensor_1"
    },
    "message": {
      "code": 16912,
      "mode": "APPEARS",
      "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"
    }
  }
]
```

328

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

```
GET /iolink/v1/gateway/events?origin=PORTS&top=3&masterNumber=2&portNumber=7
```

Example 3: Request reading all events of master 3

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

5.5 Master

There can be more than one 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:

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

5.5.1 GET /masters

This request 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 Behavior	Nothing will be changed or modified
Path	/masters
Query parameters	-
Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK
Request body	-
Response body	Array of object defined in Table 24 – Object MasterId

Table 24 – Object MasterId

Object	**MasterId**			
Property	**Type**	**Value**	**Description**	**M/O/C**
`masterNumber`	number	Integer 1..n	see 4.5.2	M
`serialNumber`	string		Vendor specific	O
`locationTag`	string		Could be used for “slot” information for the modular model (see 4.1)	O

Example:

Request

```
GET /iolink/v1/masters
```

Response

```
[
  {
    "masterNumber": 1,
    "serialNumber": "A12345678B",
    "locationTag": "slot 5"
  },
  {
    "masterNumber": 2,
    "serialNumber": "123A45B783",
    "locationTag": "slot 6"
  }
]
```

Table 25 – Resources Master

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 Master	M
`/identification`	5.5.4	POST	Write application specific identification to a Master	C[1]
NOTE 1 : This is not needed if no element of Table 31 – Object MasterIdentificationPost is implemented.				

5.5.2 GET /capabilities

Read the capabilities of a specific IO-Link Master.

Table 26 – GET /capabilities

	Description
Description	Read the capabilities of a specific IO-Link Master.
System Behavior	Nothing will be changed or modified.
Path	/masters/{masterNumber}/capabilities (see Table 116)
Query parameters	–
Errors	(see A.2), see 4.5.10
Success	HTTP 200 - OK
Request body	–
Response body	Object Master Capability (see Table 27 – Object MasterCapabilities)

Table 27 – Object MasterCapabilities

Object	**MasterCapabilities**			
Property	**Type**	**Value**	**Description**	**M/O/C**
`numberOfPorts`	number	Integer 1..n	Total number of ports	M
`maxPowerSupply`	object	Object defined in Table 115 – Object PowerSupply	Maximum total power supply for all ports.	M

Example:

Request

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

Response

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

5.5.3 GET /identification

Read all identification data of an IO-Link Master.

Table 28 – GET /identification

	Description
Description	Read all identification data of an IO-Link Master.
System Behavior	Nothing will be changed or modified.
Path	/masters/{masterNumber}/identification (see Table 116)
Query parameters	–
Errors	(see A.2), (see 4.5.10)
Success	HTTP 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`	number	integer	Vendor ID assigned by IO-Link community, see [1]	M
`masterId`	number	integer	Any vendor specific ID	M
`masterType`	string	**Enumeration:** “Unspecific” “Master acc. V1.0” “Master acc. V1.1” “Failsafe_Master” “Wireless_Master”	see [1]	M
`serialNumber`	string		Vendor specific, in analogy to [1]	O
`productId`	string		Vendor specific, in analogy to [1]	O
`vendorName`	string		Vendor specific, in analogy to [1]	M

`productName`	string		Vendor specific, in analogy to [1]	O
`hardwareRevision`	string		Vendor specific, in analogy to [1]	O
`firmwareRevision`	string		Vendor specific, in analogy to [1]	O
`vendorUrl`	string		Link to product site	O
`manualUrl`	string		Link to user manual	O
`applicationSpecificTag`	string		User defined tag in analogy to [1]	O
`locationTag`	string		User defined tag in analogy to [1]	O
`functionTag`	string		User defined tag in analogy to [1]	O

Example:

Request

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

Response

```
{
  "vendorName": "Vendor GmbH",
  "vendorId": 26,
  "masterId": 42,
  "masterType": "Master acc. V1.0",
  "serialNumber": "IOLM123456",
  "productId": "PROD-123456",
  "productName": "IO-Link Master",
  "hardwareRevision": "3.2.1.444R",
  "firmwareRevision": "3.2.1.888R",
  "vendorUrl": "http://www.io-link.com/io-link-master",
  "manualUrl": "http://www.io-link.com/io-link-master/documentation.pdf",
  "applicationSpecificTag": "Fallback reader",
  "locationTag": "Down under",
  "functionTag": "Code reading"
}
```

5.5.4 POST /identification

Write identification data of an IO-Link Master.

Table 30 – POST /identification

	Description
Description	Write identification data of an IO-Link Master.
System Behavior	If there are no restrictions the object will be modified.
Path	/masters/{masterNumber}/identification (see Table 116)
Query parameters	–
Errors	(see A.2), see 4.5.10
Success	HTTP 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, in analogy to [1]	O[1]
`locationTag`	string		User defined tag, in analogy to [1]	O[1]
`functionTag`	string		User defined tag, in analogy to [1]	O[1]
NOTE 1 At least one of the properties has to be used if the request is issued				

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 /ports

Read all the available portNumber keys with the corresponding identification information.

Table 32 – GET /ports

	Description
Description	Read all the available portNumber keys with the corresponding identification information
System Behavior	Nothing will be changed or modified
Path	/master/{masterNumber}/ports (see Table 116)
Query parameters	–
Errors	(see A.2), see 4.5.10
Success	HTTP 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
`statusInfo`	string	**Enumeration:**		M

		“DEACTIVATED” “INCORRECT_DEVICE” “DEVICE_STARTING” “DEVICE_ONLINE” “COMMUNICATION_LOST” “DIGITAL_INPUT_C/Q” “DIGITAL_OUTPUT_C/Q” “NOT_AVAILABLE”		
`deviceAlias`	string	{deviceAlias}		M

Example:

Request

```
GET /iolink/v1/masters/1/ports
```

Response

```
[
  {
    "portNumber": 1,
    "statusInfo": "DEVICE_ONLINE",
    "deviceAlias": "Distance_Sensor"
  },
  {
    "portNumber": 2,
    "statusInfo": "DEVICE_ONLINE",
    "deviceAlias": "Pressure_Sensor"
  },
  {
    "portnumber": 3,
    "statusInfo": "DIGITAL_INPUT_C/Q",
    "deviceAlias": "Switching_Sensor"
  },
  {
    "portnumber": 4,
    "statusInfo": "DEACTIVATED",
    "deviceAlias": "Empty_port"
  }
]
```

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

Read the capabilities of a specific port.

Table 35 – GET /capabilities

	Description
Description	Read the capabilities of a specific port.
System Behavior	Nothing will be changed or modified.
Path	/master/{masterNumber}/ports/{portNumber}/capabilities (see Table 116)
Query parameters	–
Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK
Request body	–
Response body	Object defined in Table 36 – Object PortCapability

Table 36 – Object PortCapability

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 115 – Object PowerSupply		M

Example:

Request

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

Response

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

5.6.3 GET /status

Read the actual status of a specific port.

Table 37 – GET /status

	Description
Description	Read the actual status of a specific port.
System Behavior	Nothing will be changed or modified.
Path	/master/{masterNumber}/ports/{portNumber}/status (see Table 116)
Query parameters	–
Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK
Request body	–
Response body	Object defined in Table 38 – Object PortStatus

Table 38 – Object PortStatus

Object	**Port Status**			
Property	**Type**	**Value**	**Remark**	**M/O/C**
`statusInfo`	string	**Enumeration:** "DEACTIVATED" "INCORRECT_DEVICE" "DEVICE_STARTING" "DEVICE_ONLINE" "COMMUNICATION_LOST" "DIGITAL_INPUT_C/Q" "DIGITAL_OUTPUT_C/Q" "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 112 – 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"				

Table 39 – Port status mapping

JSON	**SMI** see [1]
DEACTIVATED	DEACTIVATED, PORT_POWER_OFF
INCORRECT_DEVICE	PORT_DIAG
DEVICE_STARTING	PREOPERATE
DEVICE_ONLINE	OPERATE
COMMUNICATION_LOST	NO_DEVICE
DIGITAL_INPUT_C/Q	DI_C/Q
DIGITAL_OUTPUT_C/Q	DO_C/Q
NOT_AVAILABLE	NOT_AVAILABLE

Example:

Request

```
GET /iolink/v1/masters/1/ports/1/status
```

Response

```
{
  "statusInfo": "DEVICE_ONLINE",
  "iolinkRevision": "1.1",
  "transmissionRate": "COM2",
  "masterCycleTime": {
    "value": 2.3,
    "unit": "ms"
  }
}
```

5.6.4 GET /configuration

Read the actual configuration of a specific port

Table 40 – GET /configuration

	Description
Description	Read the actual configuration of a specific port.
System Behavior	Nothing will be changed or modified.
Path	/master/{masterNumber}/ports/{portNumber}/configuration (see Table 116)
Query parameters	–
Errors	(see A.2), see 4.5.10
Success	HTTP 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	see Table 42 –
`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	see Table 42

`iqConfiguration`	string	**Enumeration:** “NOT_SUPPORTED” “DIGITAL_INPUT” “DIGITAL_OUTPUT”		see Table 42
`cycleTime`	object	Object defined in Table 112 – Object CycleTime	NOTE [1]	see Table 42
`vendorId`	number		see [1]	see Table 42
`deviceId`	number		see [1]	see Table 42
`deviceAlias`	String	Default Name is: Master{masterNumer}port{portNumber}		see Table 42
NOTE 1 If the applied value for cycle time cannot exactly be mapped, the port shall use the next possible higher value. If the cycle time is greater than 132.8 ms the error 702 shall be returned.				

Table 42 – Conditions for the Port Configuration object

	Datastorage		
request/Property	**POST**	**GET**	
`mode`	O[5]	M	
`validationAndBackup`	C[1]	C[1]	
`iqConfiguration`	O[5]	M	
`cycleTime`	O[3]	C[4]	
`vendorId`	C[1,2]	C[1,2]	
`deviceId`	C[1,2]	C[1,2]	
`deviceAlias`	O[5]	M	
NOTE 1 with mode IO-LINK_MANUAL NOTE 2 with ValidationAndBackup: other than NO_DEVICE_CHECK NOTE 3 can only be used with mode IO-LINK_MANUAL or IO-LINK_AUTOSTART NOTE 4 With mode IO-LINK_MANUAL or IO-LINK_AUTOSTART it returns 0ms if the cycle time was not configured NOTE 5 At least one of the `mode`, `iqConfiguration`, `deviceAlias` property shall be included to the POST request			

Example:

Request

```
GET /iolink/v1/masters/1/ports/1/configuration
```

Response

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

5.6.5 POST /configuration

Write the wanted configuration for a specific port.

Table 43 – POST /configuration

	Description
Description	Write the wanted configuration for a specific port.
System Behavior	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	/master/{masterNumber}/ports/{portNumber}/configuration (see Table 116)
Query parameters	–
Errors	(see A.2), see 4.5.10
Success	HTTP 204 - No Content
Request body	Object defined in Table 41 – Object PortConfiguration
Response body	–

Example:

Request

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

{
  "mode": "IOLINK_AUTOSTART",
  "validationAndBackup": "TYPE_COMPATIBLE_DEVICE_V1.1",
  "iqConfiguration": "DIGITAL_INPUT",
  "cycleTime": {
    "value": 2.3,
    "unit": "ms"
  },
  "vendorId": 26,
  "deviceId": 333,
  "deviceAlias": "Distance_sensor_1"
}
```

5.6.6 GET /datastorage

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

Table 44 – GET /datastorage

	Description
Description	Read the actual datastorage content of the port.
System Behavior	Nothing will be changed or modified. If no datastorage content is available the response is empty.
Path	/master/{masterNumber}/ports/{portNumber}/datastorage (see Table 116)
Query parameters	–

Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK
Request body	–
Response body	Object defined in Table 45 – Object Datastorage

Table 45 – Object Datastorage

Object	**Datastorage**			
Property	**Type**	**Value**	**Description**	**M/O/C**
`header`	object	Object defined in Table 46 – 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 46 – 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" "1.1"	see [1]	M

Example:

Request

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

Response

```
{
  "header": {
    "vendorId": 15,
    "deviceId": 65253,
    "iolinkRevision": "1.1"
  },
  "content": "YmFzZTY0IGVuY3J5cHRlZCBjb250ZW50"
}
```

5.6.7 POST /datastorage

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

Table 47 – POST /datastorage

	Description
Description	Write the datastorage content of the port.
System Behavior	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	/master/{masterNumber}/ports/{portNumber}/datastorage (see Table 116)
Query parameters	–
Errors	(see A.2), (see 4.5.10)
Success	HTTP 204 - No Content
Request body	Object defined in Table 45 – Object Datastorage
Response body	–

Example:

Request

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

{
  "header": {
    "vendorId": 15,
    "deviceId": 65253,
    "iolinkRevision": "1.1"
  },
  "content": "YmFzZTY0IGVuY3J5cHRlZCBjb250ZW50"
}
```

5.7 Devices

Devices are addressed by device names. See port configuration in 5.6.5. for deviceAlias assignment. If no deviceAlias is assigned the default name is used. IODD support is required for all requests where query parameter `format` has the value `iodd`.

Example:

Addressing by default device name (e.g. Device is on port 6 of Master1)

```
GET /iolink/v1/devices/master1port6/identification
```

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

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

5.7.1 GET /devices

Get all available deviceAlias keys and the location by Master number and port number.

Table 48 – GET /devices

	Description
Description	Get all available deviceAlias keys and the location by Master number and Port number.
System Behavior	Nothing will be changed or modified.
Path	/devices
Query parameters	–

Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK
Request body	-
Response body	Array of objects defined in Table 49 – Object DeviceAlias

Table 49 – Object DeviceAlias

Object	**DeviceAlias**			
Property	**Type**	**Value**	**Description**	**M/O/C**
`deviceAlias`	string		Device name is according to 4.5.6	M
`masterNumber`	number	Integer 1..255		M
`portNumber`	number	Integer 1..255		M

Example:

Request

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

Response

```
[
  {
    "deviceAlias": "DT35",
    "masterNumber": 1,
    "portNumber": 1
  },
  {
    "deviceAlias": "TAD081",
    "masterNumber": 1,
    "portNumber": 2
  },
  {
    "deviceAlias": "BNI_IOL",
    "masterNumber": 1,
    "portNumber": 3
  },
  {
    "deviceAlias": "master1port4",
    "masterNumber": 1,
    "portNumber": 4
  }
]
```

Table 50 – Resources Devices

Resources /devices/{deviceAlias}	**Clause**	**HTTP Method**	**Remark**	**M/O/C**
Capabilities				
`/capabilities`	5.7.2	GET	Read the capabilities from the specific Device	M
Identification				

`/identification`	5.7.3	GET	Read the identification from 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 values (input and output) from the specified Device format “byteArray” or “iodd” is given by a query string	M
`/processdata/getdata/value`	5.7.6	GET	Read the process data input values from the specified Device format “byteArray” or “iodd” is given by a query string	M
`/processdata/setdata/value`	5.7.7	GET	Read the process data output values from the specified Device format “byteArray” or “iodd” is given by a query string	M
`/processdata/value`	5.7.8	POST	Write the process data output values to the specified Device format “byteArray” or “iodd” is given by the request body	M
Parameter information				
`/parameters`	5.7.9	GET	Read all available parameter indices and parameter names from the specific Device. IODD support is required	C1)
`/parameters/{index}/subindices`	5.7.10	GET	Read all available parameter sub-indices and sub-parameter names from the specific device. IODD support is required	C1)
`/parameters/{parameterName}/subindices`	5.7.11	GET	Read all available parameter sub-indices and sub-parameter names from the specific device (referenced by parameter name). IODD support is required	C1)
Parameter values				
`/parameters/{index}/value`	5.7.12	GET	Read a parameter value from 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`	5.7.13	GET	Read a parameter value from the specific device with the given index and subindex. The requested format “byteArray” or “iodd” is given by a query string	M
`/parameters/{parameterName}/value`	5.7.14	GET	Read a parameter value from the specific Device by parameter name. IODD support is required	C[1]
`/parameters/{parameterName}/subindices/{subParametername}/value`	5.7.15	GET	Read a parameter value from the specific Device by parameter name and subname. IODD support is required.	C[1]
`/parameters/{index}/value`	5.7.16	POST	Write a parameter value with the given index to the specified Device. The format is given within the body .	M
`/parameters/{parameterName}/value`	5.7.17	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`	5.7.18	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`	5.7.19	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`	5.7.20	POST	Write or read one or more parameters using the block parameterization method. The format can be byteArray or IODD based.	M
Events				
`/events`	5.7.21	GET	reading the EventLog filtered for a specific Device	M
NOTE The evaluation of IODD conditions will be done implicit by the JSON server NOTE 1 Mandatory if IODD support is available				

5.7.2 GET /capabilities

Read the capabilities of the specific Device.

Table 51 – GET /capabilities

	Description
Description	Read the capabilities of the specific Device.
System Behavior	Nothing will be changed or modified.
Path	/devices/{deviceAlias}/capabilities (see Table 116)
Query parameters	–
Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK
Request body	–
Response body	Object defined in Table 52 – Object DeviceCapabilities

Table 52 – Object DeviceCapabilities

Object	**Device capabilities**			
Property	**Type**	**Value**	**Description**	**M/O/C**
`minimumCycleTime`	object	Object defined in Table 112 – Object CycleTime		M
`supportedProfiles`	Array of numbers		see [10]	C[1]
NOTE 1 Mandatory if Device supports profiles				

Example

Request

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

Response

```
{
  "minimumCycleTime": {
    "value": 2.3,
    "unit": "ms"
   },
  "supportedProfiles": [10,32770]
}
```

5.7.3 GET /identification

Read the identification of the Device.

Table 53 – GET /identification

	Description
Description	Read the identification of the Device.
System Behavior	Nothing will be changed or modified.
Path	/devices/{deviceAlias}/identification (see Table 116)
Query parameters	–
Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK
Request body	–
Response body	Object defined in Table 54 – Object Device Identification Get

Table 54 – 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
`productId`	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 according to [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

Write Device specific tags including the applicationSpecificTag, the locationTag and the functionTag (see [1]) if available.

Table 55 – POST /identification

	Description
Description	Write all identification data of an IO-Link Device.
System Behavior	If there are no restrictions the object will be modified.
Path	/devices/{deviceAlias}/identification (see Table 116)
Query parameters	–
Errors	(see A.2), (see 4.5.10)
Success	HTTP 204 - No Content
Request body	–
Response body	Object defined in Table 56 – Object DeviceIdentificationPOST

Table 56 – Object DeviceIdentificationPOST

Object	**DeviceIdentificationPOST**			
Property	**Type**	**Value**	**Description**	**M/O/C** [1]
`applicationSpecificTag`	string		see [1]	O[2]
`locationTag`	string		see [1]	O[2]
`functionTag`	string		see [1]	O[2]
NOTE 1 M/O/C according [1] NOTE 2 at least one of the properties shall be present				

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 input values and output values of the specified Device. The request format can be byteArray or IODD based. If IODD format is requested the IODD feature shall be supported.

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

Table 57 – GET /processdata/value

	Description
Description	Read the process data (input values and/or the output values) of the specified Device. The requested format is indicated by a query string and can be byteArray (default) or IODD based.
System Behavior	Nothing will be changed or modified.
Path	/devices/{deviceAlias}/processdata/value (see Table 116)
Query parameters	Format request parameters (see Table 58)
Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK
Request body	–
Response body	Object defined in Table 59 – Object ProcessDataInOut

Table 58 – format request query parameters

Query parameter	**Values**	**Remark**
format	**Enumeration:** “byteArray” “iodd”	Default is “byteArray”

Table 59 – Object ProcessDataInOut

Object	**ProcessDataInOut**			
Property	**Type**	**Value**	**Description**	**M/O/C**
`getData`	object	Object defined in Table 60 – Object ProcessData		C[1]
`setData`	object	Object defined in Table 60 – Object ProcessData		C[2]
NOTE 1 if the request is GET /processdata/getdata/value or GET /processdata/value the process data content is “getData” NOTE 2 if the request is GET /processdata/setdata/value or GET /processdata/value the process data content is “setData”				

Table 60 – Object ProcessData

Object	**ProcessData**			
Property	**Type**	**Value**	**Description**	**M/O/C**
`iolink`	object	Object defined in Table 61 – Object processDataIOL		C[1]
`iqValue`	boolean	true, false	true = high signal	C[2]
`cqValue`	boolean	true, false	true = high signal	C[3]

NOTE 1 If port is in IO-Link mode and Device is connected

NOTE 2 If I/Q is available and configured as digital input for GET or configured as digital output for POST

NOTE 3 If C/Q is available and configured as digital input for GET or configured as digital output for POST

Table 61 – Object processDataIOL

Object	**processDataIOL**			
Property	**Type**	**Value**	**Description**	**M/O/C**
`valid`	boolean	true,false	For output data valid is always true.	M
`value`	array of numbers	Byte array (see 4.5.8)	order in big endian	C[1]
`value`	boolean, string, number		simple types	C[2]
`value`	object	Object defined in Table 72 – Object complexParameterValue		C[3]

NOTE 1 If the format is “byteArray”

NOTE 2 If the format is “iodd” and process data are of simple type, naming shall be according 4.5.6

NOTE 3 If the format is “iodd” and process data are of complex type, naming shall be according 4.5.6

Examples:

Example 1:

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

Request (format byteArray)

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

Response

```
{
  "getData": {
    "iolink": {
      "valid": true,
      "value": [12,22,216]
    },
    "iqValue": false
  },
  "setData": {
    "iolink": {
      "valid": true,
      "value": [128,221,134]
    }
  }
}
```

Example 2:

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

Request (format IODD)

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

Response

```
{
  "getData": {
    "iolink": {
      "valid": true,
      "value": {"Pressure": 1.2}
    },
    "iqValue": false
   },
   "setData": {
     "iolink": {
       "valid": true,
       "value": {"Temperature": 50}
     }
   }
}
```

5.7.6 GET /processdata/getdata/value

Read the process data input 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 be accessed by addressing a device even if no physical device is attached at the port. Process data comprises digital inputs on I/Q and C/Q (also if the Device is in “SIO” mode) as well as IO-Link process data if the port is in communication.

Table 62 – GET /processdata/getdata/value

	Description
Description	Read the process data input value of the specified Device. The requested format is indicated by a query string and can be byteArray (default) or IODD based.
System Behavior	Nothing will be changed or modified.
Path	/devices/{deviceAlias}/processdata/getdata/value (see Table 116)
Query parameters	Format request parameters (see Table 58)
Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK
Request body	–
Response body	Object defined in Table 60 – Object ProcessData

Examples:

Example 1: Device is a sensor in IO-Link communication, I/Q is available and configured as digital input

The IO-Link process data is 0xC16D8 = 792280_{dec}

Request (format is byteArray).

```
GET /iolink/v1/devices/master1port6/processdata/getdata/value?format=byteArray
```

Response

```
{
  "iolink": {
    "valid": true,
    "value": [12,22,216]
  },
  "iqValue": true
}
```

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

Request (format IODD).

```
GET /iolink/v1/devices/master1port6/processdata/getdata/value?format=iodd
```

Response

```
{
  "iolink": {
    "valid": true,
    "value": 15.2
  },
  "iqValue": true
}
```

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

Request (format IODD).

```
GET /iolink/v1/devices/master1port6/processdata/getdata/value?format=iodd
```

Response

```
{
  "iolink": {
    "valid": true,
    "value": {
      "Distance": {"value": 15 },
      "Quality": {"value": 12 }
    }
  },
  "iqValue": true
}
```

Example 3: Device is a sensor in SIO Mode, I/Q and C/Q are available and configured as digital inputs

Request (format IODD).

```
GET /iolink/v1/devices/master1port6/processdata/getdata/value?format=iodd
```

Response

```
{
  "iqValue": true,
  "cqValue": false
}
```

5.7.7 GET /processdata/setdata/value

Read the process data output 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 be accessed by addressing a device even if no physical device is attached at the port. Process data comprises digital inputs on I/Q and C/Q (also if the Device is in “SIO” mode) as well as IO-Link process data if the port is in communication.

Table 63 – GET /processdata/setdata/value

	Description
Description	Read the process data (input data) of the specified Device. The requested format is indicated by a query string and can be byteArray (default) or IODD based.
System Behavior	Nothing will be changed or modified.
Path	/devices/{deviceAlias}/processdata/setdata/value (see Table 116)
Query parameters	Format request parameters (see Table 58)
Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK
Request body	–
Response body	Object defined in Table 60 – Object ProcessData

5.7.8 POST /processdata/value

Write the process data output value 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 I/Q or C/Q are available. The Device is addressed by the device name (see 4.4.3).

Table 64 – POST /processdata/value

	Description
Description	Write the process data output value to the specified Device. The requested format is implicitly defined within the body. No query parameters are needed.
System Behavior	If there are no restrictions process data will be updated.
Path	/devices/{deviceAlias}/processdata/value (see Table 116)

Query parameters	–
Errors	(see A.2),(see 4.5.10)
Success	HTTP 204 - No Content
Request body	Object defined in Table 60 – Object ProcessData
Response body	–

Examples:

Example 1:

Device is an actuator in IO-Link communication, I/Q is available and configured as a digital output.

Request (implicit format byteArray)

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

{
  "iqValue": false,
  "iolink": {
    "valid": true,
    "value": [12,22,216]
  }
}
```

Example 2:

Device is an actuator in IO-Link communication, I/Q 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

{
  "iqValue": true,
  "iolink": {
    "valid": true,
    "value": 32.4
  }
}
```

Example 3:

Device is an actuator in IO-Link communication, I/Q 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

{
  "iqValue": true,
  "iolink": {
    "valid": true,
    "value": {
      "Valve_1": true,
      "Valve_2": false
    }
  }
}
```

5.7.9 GET /parameters

Read a list of all available parameter indices and parameter names of the specific Device. IODD support is required.

Table 65 – GET /parameters

	Description
Description	Read a list of all available parameter indices and parameter names of the specific Device.
System Behavior	Nothing will be changed or modified.
Path	/devices/{deviceAlias}/parameters (see Table 116)
Query parameters	–
Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK
Request body	–
Response body	Array of object defined in Table 66 – Object parameterList

Table 66 – Object parameterList

Object	**parameterList**			
Property	**Type**	**Value**	**Description**	**M/O/C**
`index`	number	integer	see [1]	M
`parameterName`	string	Name of parameter	see 4.5.6	M

Example:

Request

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

Response

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

5.7.10 GET /parameters/{index}/subindices

Read all available parameter subindices and subparameter names (referenced by ISDU). IODD support is required.

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

	Description
Description	Read all available parameter sub-indices, sub-parameter names (referenced by ISDU).
System Behavior	Nothing will be changed or modified.
Path	/devices/{deviceAlias}/parameters/{index}/subindices (see Table 116)
Query parameters	–
Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK
Request body	–
Response body	Array of object defined in Table 68 – Object deviceSubParameters

Table 68 – Object deviceSubParameters

Object	**deviceSubParameters**			
Property	**Type**	**Value**	**Remark**	**M/O/C**
`subIndex`	number	integer		M
`subParameterName`	string		Name of sub parameter see 4.5.6	M

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"
  }
]
```

5.7.11 GET /parameters/{parameterName}/subindices

Read all available parameter subindices, subparameter names (referenced by parameter name). IODD support required.

Table 69 – GET /parameters/{parameterName}/subindices

	Description
Description	Read all available parameter subindices, subparameter names (referenced by parameter name).
System Behavior	Nothing will be changed or modified.
Path	/devices/{deviceAlias}/parameters/{parameterName}/subindices (see Table 116)
Query parameters	–
Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK
Request body	–
Response body	Array of object defined in Table 68 – Object deviceSubParameters

Example:

A Device has a parameter named “Switch_properties” with some subindices.

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"
  }
]
```

5.7.12 GET /parameters/{index}/value

Read a specific parameter value and its sub-parameter values (if the parameter has complex type) 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 70 – GET /parameters/{index}/value

	Description
Description	Read a specific parameter and its subparameter values (if the parameter has complex type) with the given index from the specified Device.
System Behavior	Nothing will be changed or modified.
Path	/devices/{deviceAlias}/parameters/{index}/value (see Table 116)
Query parameters	See Table 58
Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK

Request body	–
Response body	one of objects that are listed as options in the deviceParameterValues object list (see Table 71)

Table 71 – deviceParameterValues object list

deviceParameterValues object list			
object	Object defined in Table 74 – Object rawParameterValue		C¹
object	Object defined in Table 73 – Object simpleParameterValue		C²
object	Object defined in Table 72 – Object complexParameterValue		C³
NOTE 1 If format is “byteArray” NOTE 2 If format is “iodd” and process data are of simple type, naming shall be according to 4.5.6 NOTE 3 only available for indices or process data with complex data types			

Table 72 – Object complexParameterValue

Object	**complexParameterValue**			
Property	**Type**	**Value**	**Remark**	**M/O/C**
`{parameterName}`	object	Object defined in Table 73 – Object simpleParameterValue		
NOTE IODDname is based on rules see 4.5.6				

Table 73 – Object simpleParameterValue

Object	**simpleParameterValue**			
Property	**Type**	**Value**	**Remark**	**M/O/C**
`value`	boolean, string, number		Simple types e.g. "value":15.2	

Table 74 – Object rawParameterValue

Object	**rawParameterValue**			
Property	**Type**	**Value**	**Remark**	**M/O/C**
`value`	array of numbers		see 4.5.8	

Example:

Example 1: Request (format byteArray)

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

Response

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

Example 2: Request (format iodd)

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

Response (simple type)

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

Example 3: Request (format iodd)

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

Response (complex type)

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

5.7.13 GET /parameters/{index}/subindices/{subindex}/value

Read the value of a specific sub parameter with the given index and subindex 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 75 – GET /parameters/{index}/subindices/{subindex}/value

	Description
Description	Read the value of a specific sub parameter with the given index and subindex.
System Behavior	Nothing will be changed or modified.
Path	/devices/{deviceAlias}/parameters/{index}/subindices/{subindex}/value (see Table 116)
Query parameters	See Table 58
Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK
Request body	–
Response body	one of objects that are listed as options in the deviceParameterValues object list (see Table 71)

Example

Example 1: Request (format byteArray)

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

Response

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

Example 2: Request (format iodd)

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

Response

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

5.7.14 GET /parameters/{parameterName}/value

Read a specific parameter value with the given name. The parameterName shall follow the rules of IODD based naming rules (see 4.5.6). IODD support is needed.

Table 76 – GET /parameters/{parameterName}/value

	Description
Description	Read a specific parameter value with the given name.
System Behavior	Nothing will be changed or modified.
Path	/devices/{deviceAlias}/parameters/{parameterName}/value (see Table 116)
Query parameters	Format request parameters (see Table 58)
Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK
Request body	–
Response body	one of objects that are listed as options in the deviceParameterValues object list (see Table 71)

Example

Example 1: Request (format byteArray)

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

Response

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

Example 2: Request (format iodd)

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

Response (simple type)

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

Example 3: Request (format iodd)

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

Response (complex type)

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

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

Read the content of a specific sub parameter value with the given name and subname from the specified Device.

Table 77 – GET /parameters/{parameterName}/subindices/{subParameterName}/value

	Description
Description	Read the content of a specific sub parameter value with the given name and subname.
System Behavior	Nothing will be changed or modified.
Path	/devices/{deviceAlias}/parameters/{parameterName}/subindices/{subParameterName}/value (see Table 116)
Query parameters	Format request parameters (see Table 58)
Errors	(see A.2)), (see 4.5.10)
Success	HTTP 200 - OK
Request body	–
Response body	one of objects that are listed as options in the deviceParameterValues object list (see Table 71)

Example

Example 1: Request (format byteArray)

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

Response

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

Example 2: Request (format iodd)

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

Response

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

5.7.16 POST /parameters/{index}/value

Write the parameter with the given index to the Device. The format is given within the body.

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

	Description
Description	Write the parameter with the given index to the Device.
System Behavior	If there are no restrictions, parameter will be updated.
Path	/devices/{deviceAlias}/parameters/{index}/value (see Table 116)
Query parameters	–
Errors	(see A.2), (see 4.5.10)
Success	HTTP 204 - No Content
Request body	one of objects that are listed as options in the deviceParameterValues object list (see Table 71)
Response body	–

Example 1 (raw bytes)

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

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

Example 2 (simple type)

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

{
  "value": 15
}
```

Example 3 (complex type)

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

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

5.7.17 POST /parameters/{parameterName}/value

Write the parameter with the given name to the specified Device. The format is given within the body.

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

	Description
Description	Write the parameter with the given name to the specified Device.
System Behavior	If there are no restrictions parameter will be updated.
Path	/devices/{deviceAlias}/parameters/{parameterName}/value (see Table 116)
Query parameters	–
Errors	(see A.2)), (see 4.5.10)
Success	HTTP 204 - No Content
Request body	One of objects that are listed as options in the deviceParameterValues object list (see Table 71)
Response body	–

Example

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

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

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

{
  "value": 15
}
```

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

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

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

Write the sub parameter with the given index and subindex to the Device. The format is given within the body.

Table 80 – POST /parameters/{index}/subindices{subindex}value

	Description
Description	Write the sub parameter with the given index and subindex to the Device.
System Behavior	If there are no restrictions parameter will be updated.
Path	/devices/{deviceAlias}/parameters/{index}/subindices/{subindex}value (see Table 116)
Query parameters	–
Errors	(see A.2),), (see 4.5.10)
Success	HTTP 204 - No Content

Request body	One of objects that are listed as options in the deviceParameterValues object list (see Table 71)
Response body	–

Example

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

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

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

Write the sub-parameter with the given parameter name and subParameterName to the specified Device.

Table 81 – POST /parameters/{ parameterName }/subindices/{subParameterName}/value

	Description
Description	Write the sub-parameter with the given parameter name and subParameterName. The requested format is implicit defined within the body.
System Behavior	If there are no restrictions parameter will be updated.
Path	/devices/{deviceAlias}/parameters/{parameterName}/subindices/{subParametername}value see Table 116
Query parameters	–
Errors	(see A.2), (see 4.5.10)
Success	HTTP 204 - No Content
Request body	One of objects that are listed as options in the deviceParameterValues object list (see Table 71)
Response body	–

Example

```
POST
/iolink/v1/devices/sensor34/parameters/name_of_parameter/subindices/name_of_subpara
meter/value

{
  "value": 152
}
```

5.7.20 POST /blockparametrization

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

Table 82 – POST /blockparametrization

	Description
Description	Writing or reading one or more parameters as a block (consistent).
System Behavior	If there are no restrictions for writing, the objects will be modified.
Path	/devices/{deviceAlias}/blockparametrization (see Table 116)

Query parameters	Format request parameters (see Table 58)
Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK
Request body	Object defined in Table 83 – Object BlockParametrizationRequest
Response body	Array of object defined in Table 84 – Object BlockParametrizationAnswer

Table 83 – Object BlockParametrizationRequest

Object	**BlockParametrizationReadRequest**			
Property	**Type**	**Value**	**Remark**	**M/O/C**
`direction`	string	**Enumeration:** “read” “write”		M
`parameters`	Array of objects	Object defines in Table 85 – Object BlockParameters	Addresses and values (on write) of all parameters or subParameters	M

Table 84 – Object BlockParametrizationAnswer

Object	**BlockparametrizationAnswer**			
Property	**type**	**Value**	**Remark**	**M/O/C**
`identifier`	object	Object defined in Table 87 – Object SubIndexAddress		M
`result`	object	Object defined in Table 88 – Object ParameterResult		M

Table 85 – Object BlockParameters

Object	**BlockParameters**			
Property	**Type**	**Value**	**Remark**	**M/O/C**
`identifier`	object	Object defined in Table 87 – Object SubIndexAddress		M
`content`	object	Object defined in Table 86 – Object ParameterValue		C[1]
NOTE 1 when `direction` is write				

Table 86 – Object ParameterValue

Object	**ParameterValue**			
Property	**Type**	**Value**	**Remark**	**M/O/C**
`value`	array of numbers	see 4.5.8		C[1,4]

`value`	boolean, string, number		Simple types	C^{2,4}
`value`	object	Object defined in Table 72 – Object complexParameterValue	Complex types	C^{3,4}
NOTE 1 this is format "byteArray" NOTE 2 this is for format "iodd" NOTE 3 this is for format "iodd" but not for subindex/subparameter access NOTE 4 only one `value` property can be present				

Table 87 – Object SubIndexAddress

Object	**SubIndexAddress**			
Property	**Type**	**Value**	**Remark**	**M/O/C**
`index`	number	integer		C^{1}
`subIndex`	number	integer	NOTE 3	C^{1}
`parameterName`	string	{parameterName}		C^{2}
`subParameterName`	string	{subParameterName}	NOTE 3	C^{2}
NOTE 1 this is format "byteArray" NOTE 2 this is format "iodd" if IODD is supported NOTE 3 `subIndex` or sub`ParameterName` are only needed if sub parameters are addressed				

Table 88 – Object ParameterResult

Object	**ParameterResult**			
Property	**Type**	**Value**	**Remark**	**M/O/C**
`parameterExchangeResult`	string	**Enumeration:** "WRITE_SUCCESS" "READ_SUCCESS" "ERROR"		M
`content`	object	Object defined in Table 86 – Object ParameterValue		C^{1}
`iolinkError`	object	Object defined in Table 114 – Object iolinkError		C^{2}
NOTE 1 when `direction` is read NOTE 2 when parameterExchangeResult is ERROR				

Example:

Example 1: Read Request (format byteArray)

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

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

Response (request format is byteArray)

```
{
  [
    {
      "identifier": {"index": 123},
      "result":
        {
          "parameterExchangeResult": "READ_SUCCESS",
          "content": {"value": [15,232,22]}
        }
    },
    {
      "identifier": {"index": 233,"subIndex": 2},
      "result":
        {
          "parameterExchangeResult": "READ_SUCCESS",
          "content": {"value": [23,149,206]}
        }
    }
  ]
}
```

Example 2: Read Request (format is iodd)

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

{
  "direction": "read",
  "parameters": [
    {
      "identifier": {"parameterName": "applicationTag"}
    },
    {
       "identifier": {"parameterName": "hystersis","subParameterName": "channelB" }
    }
  ]
}
```

Response

```
{
  [
    {
      "identifier": {"parameterName": "Application_tag"},
      "result":
        {
          "parameterExchangeResult": "READ_SUCCESS",
          "content": {"value": "Level 2, row 3"}
        }
     },
     {
       "identifier": {"parameterName": "Hysteresis","subParameterName":
"Channel_B"},
       "result":
         {
           "parameterExchangeResult": "READ_SUCCESS",
           "content": {"value": 123}
         }
      }
   ]
}
```

Example 3: Write Request (format is byteArray)

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

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

Response

```
[
  {
    "identifier": {
      "index": 123
    },
    "result": {
      "parameterExchangeResult": "WRITE_SUCCESS"
    }
  },
  {
    " identifier": {
      "index": 233
    },
    "result": {
      "parameterExchangeResult": "WRITE_SUCCESS"
    }
  }
]
```

Example 4: Write Request (format iodd)

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

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

Response

```
[
  {
    "identifier": {
      "parameterName": "Application_tag"
    },
    "result": {
      "parameterExchangeResult": "WRITE_SUCCESS"
    }
  },
  {
    "identifier": {
      "parameterName": "Hysteresis",
      "subParameterName": "Channel_B"
    },
    "result": {
      "parameterExchangeResult": "WRITE_SUCCESS"
    }
  }
]
```

5.7.21 GET /events

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

Table 89 – GET /events

	Description
Description	Read the EventLog for a specific Device.
System Behavior	Nothing will be changed or modified.
Path	/devices/{deviceAlias}/events (see Table 116)
Query parameters	Device Event log query parameters (see Table 90)
Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK
Request body	–
Response body	Object defined in Table 20 – Object GatewayEventLog

Table 90 – Device Event log query parameters

Query parameter	**Value**	**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 m 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 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 91 – 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/file`	5.8.2	POST	Store or update an IODD	M
`/iodds/file`	5.8.4	GET	Get a specific IODD	O
`/iodds`	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 92 – GET /iodds

	Description
Description	Get a list of all IODDs (representations) that are available on the Gateway.
System Behavior	Nothing will be changed or modified.
Path	/iodds
Query parameters	?vendorId={vendorId}&deviceId={deviceId}&revision={iolinkRevision} all query parameters are optional
Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK
Request body	–
Response body	Array of objects defined in Table 93 – Object IODDIdentification

Table 93 – 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/file

Store or update an IODD (representation) on the Gateway. Application Type is XML

Table 94 – POST /iodds/file

	Description
Description	Store or update an IODD on the Gateway.
System Behavior	If there are no restrictions, IODD will be updated.
Path	/iodds/file
Query parameters	–
Errors	(see A.2), (see 4.5.10)
Success	HTTP 204 - No Content
Request body	XML file encoded with the MIME type application/xml
Response body	–

5.8.3 DELETE /iodds

Delete IODDs (representation) on the Gateway. The IODDs (representation) are addressed by query parameters.

Table 95 – DELETE /iodds

	Description
Description	Delete IODDs on the Gateway.
System Behavior	If there are no restrictions, IODD will be deleted.
Path	/iodds
Query parameters	?vendorId={vendorId}&deviceId={deviceId}&revision={iolinkRevision} all query parametes are optional
Errors	(see A.2), (see 4.5.10)
Success	HTTP 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/file

Read a specific IODD (XML file) from the Gateway. The IODD (representation) is addressed by query parameters.

Table 96 – GET /iodds/file

	Description
Description	Read a specific IODD (XML file) from the Gateway.
System Behavior	Nothing will be changed or modified.
Path	/iodds/file
Query parameters	?vendorId={vendorId}&deviceId={deviceId}&revision={iolinkRevision} all query parameters are mandatory
Errors	(see A.2), (see 4.5.10)
Success	HTTP 200 - OK
Request body	–
Response body	XML file encoded with the MIME type application/xml

Example:

Request

```
GET /iolink/v1/iodds/file?vendoId=26&deviceId=34654&revision="1.1"
```

5.9 MQTT

This section is related to the configuration of the MQTT client (publisher) defining publishing topics and configurations of the MQTT server (broker). Referenced MQTT Version is [3].

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

Table 97 – 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 98 – GET /mqtt/configuration

	Description
Description	Read the MQTT configuration.
System Behavior	Nothing will be changed or modified.
Path	/mqtt/configuration
Query parameters	–
Errors	(see A.2)
Success	HTTP 200 - OK
Request body	–
Response body	Object defined in Table 99 – Object MQTTConfiguration

Table 99 – Object MQTTConfiguration

Object	**MQTT Base Configuration**			
Property	**Type**	**Value**	**Description**	**M/O/C**
`clientMode`	string	**Enumeration:** “ACTIVE” “INACTIVE”		M
`serverAddress`	string		url of the server	C[1]
`username`	string			C[1]
`password`	string			C[1]
`lastWill`	object	Object defined in Table 100 – Object LastWill		C[1]

`keepAliveTime`	number		Time in seconds see [3]	C[1]
NOTE 1 Mandatory if the `clientMode` is "ACTIVE"				

Table 100 – 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:

Request

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

Response

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

5.9.2 POST /mqtt/configuration

Write the MQTT configuration to the Gateway by updating the object "MQTT Configuration"

Table 101 – POST /mqtt/configuration

	Description
Description	Write the MQTT configuration to the Gateway.
System Behavior	If there are no restrictions the configuration will be updated.
Path	/mqtt/configuration
Query parameters	–
Errors	(see A.2)
Success	HTTP 204 - OK
Request body	Object defined in Table 99 – Object MQTTConfiguration
Response body	–

Example:

Request

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

5.9.3 GET /mqtt/topics

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

Table 102 – GET /mqtt/topics

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

Table 103 – Object MQTTtopic

Object	**MQTTtopic**			
Property	**Type**	**Value**	**Description**	**M/O/C**
`topicId`	number		The topic number as a reference	M[2]
`topicName`	string			O[3]
`qos`	string	**Enumeration:** “0_ONLY_ONCE” “1_AT_LEAST_ONCE” “2_EXACTLY_ONCE”		M
`deviceAlias`	string	{deviceAlias}		M
`processData`	object	MQTT ProcessData topic (see Table 104)		C[1]
`event`	object	null		C[1]
NOTE 1 Either one of processData or event shall be used NOTE 2 This is not applicable for POST NOTE 3 If the topicName is not specified the default structure of building topic names is like following:				

`<deviceAlias>/processData` for `process data`
`<deviceAlias>/events` for `event`
The topicName is bound to a device

Table 104 – Object MQTT ProcessDataTopic

Object	**MQTT ProcessDataTopic**			
Property	**Type**	**Value**	**Description**	**M/O/C**
`direction`	string	**Enumeration:** “getData” “setData” “getSetData”		M
`format`	string	**Enumeration:** “byteArray” “iodd”		M
`interval`	object	“CycleTime” (see Table 112)		C[1]
`onChange`	Boolean	true, false		C[1]
NOTE 1 Either one of interval or onChange				

Example:

Request

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

Response

```
[
  {
    "topicId": 1,
    "topicName": "Sensor34/processDate",
    "qos": "1_AT_LEAST_ONCE",
    "deviceAlias": "DT35",
    "processData": {
      "direction": "getData",
      "format": "iodd",
      "interval": {
        "value": 10,
        "unit": "ms"
      }
    }
  },
  {
    "topicId": 2,
    "topicName": "Sensor34/event",
    "qos": "2_EXACTLY_ONCE",
    "deviceAlias": "TAD081",
    "event": null
  },
  {
    "topicId": 3,
    "topicName": "PD",
    "qos": "0_ONLY_ONCE",
    "deviceAlias": "BNI_IOL",
    "processData": {
      "direction": "getSetData",
      "format": "iodd",
      "onChange": true
    }
  }
]
```

5.9.4 POST /mqtt/topics

Create one new MQTT topic on the Gateway

Table 105 – POST /mqtt/topics

	Description
Description	Create a new MQTT topic on the Gateway.
System Behavior	If there are no restrictions the object will be modified.
Path	/mqtt/topics
Query parameters	–
Errors	(see A.2)
Success	HTTP 204 - OK
Request body	Object defined in Table 103 – Object MQTTtopic
Response body	–

Example:

Request

```
POST /iolink/v1/mqtt/topics
{
  "qos": "1_AT_LEAST_ONCE",
  "deviceAlias": "DT35",
  "processData": {
    "direction": "getData",
    "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 106 – DELETE /mqtt

	Description
Description	Deletes a MQTT topic.
System Behavior	If there are no restrictions the topic will be deleted.
Path	/mqtt/topics/{topicId}
Query parameters	–
Errors	(see A.2)
Success	HTTP 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 107 – GET /mqtt/topics

	Description
Description	Read one specific MQTT topic.
System Behavior	Nothing will be changed or modified.
Path	/mqtt/{topicId}
Query parameters	–
Errors	(see A.2)
Success	HTTP 200 - OK
Request body	–
Response body	Object defined in Table 103 – Object MQTTtopic

Example:

Request

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

Response

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

5.9.7 GET /mqtt/connectionstatus

Read the connection status of the MQTT client to the MQTT server.

Table 108 – GET /mqtt/connectionstatus

	Description
Description	Read the connection status of the MQTT client to the MQTT server.
System Behavior	Nothing will be changed or modified.
Path	/mqtt/connectionstatus

Query parameters	–
Errors	(see A.2)
Success	HTTP 200 - OK
Request body	–
Response body	Object defined in Table 109 – Object MQTTConnectionStatus

Table 109 – Object MQTTConnectionStatus

Object	**MQTT Base Configuration**			
Property	**Type**	**Value**	**Description**	**M/O/C**
`connectionStatus`	string	**Enumeration:** “CLIENT_INACTIVE” “CONNECTION_ACCEPTED” “UNACCEPTABLE_PROTOCOL_VERSION” “IDENTIFIER_REJECTED” “SERVER_UNAVAILABLE” “BAD_USERNAME_OR_PASSWORD” “NOT_AUTHORIZED”		M
`serverAddress`	string		url of the server	M
`upTime`	number		Time in seconds	M

6 MQTT topics format

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

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

Example: (Process data topic)

```
{
 "getData": {
   "iolink": {
     "valid": true,
     "value": {
       "Distance": 55,
       "Quality": 12
     }
   },
   "iqValue": true
  }
}
```

For event log entries the topic format is as defined Table 20 – Object GatewayEventLog. Only one EventLog entry shall be published at the time the event was entered into the EventLog. One MQTT message shall contain only one event.

Example: (Event Log entry topic)

```
{
  "time": "2018-05-18T07:31:54.123Z",
  "severity": "WARNING",
  "origin": {
    "masterNumber": 1,
    "portNumber": 1,
      "device": "Temp_sensor_1"
  },
  "message": {
    "msgCode": 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 by the status code.

Table 110 – HTTP Status codes

Status Code		Description
200	OK	The request is completed successfully.
201	Created	A new resource has been created successfully.
204	No Content	Response code for POST requests without content and for DELETE requests.
400	Bad Request	Processing of request is rejected
403	Forbidden	The user is not permitted to perform the requested operation.
404	Not Found	The requested resource did not exist.
500	Server error	Request cannot be processed successfully due to issues on server side

A.2 JSON Errors

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

Table 111 – JSON Error codes

Error Code	HTTP Status code	JSON Message	Remark
General errors			
101	500	Internal server error	
102	500	Internal communication error	
103	404	Operation not supported	
104	400	Action locked by another client	Fieldbus controller or another gateway protocol has claimed priority
105	501	IODD feature not supported	
106	501	MQTT feature not supported	
150	403	Permission denied	due to user management restrictions
JSON parsing errors			
201	400	JSON parsing failed	Error while parsing the incoming JSON value)
202	400	JSON data value invalid	Error while parsing a specific JSON value, e.g. malformed IP address
203	400	JSON data type invalid	e.g. `string` instead of `number`
204	400	Enumeration value unknown	
205	400	JSON data value out of range	Exceeds the minimum or maximum value

206	400	JSON data value out of bounds	An array/string was accessed exceeding its maximum length
207	400	deviceAlias is not unique	
208	400	POST request without content	
Resource access errors			
301	404	Resource not found	e.g. wrong URL
302	404	masterNumber not found	
303	404	portNumber not found	
304	404	deviceAlias not found	
305	400	Query parameter name invalid	
306	400	Query parameter value invalid	
307	400	Port is not configured to IO-Link	e.g. not in `IOLINK_MANUAL` or `IOLINK_AUTOSTART` mode
308	404	IO-Link Device is not accessible	e.g. 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	The additional *iolinkErrorCode* and *iolinkErrorMessage* fields contain the IO-Link error code and the incident text from the ErrorTypes table. See [1]
312	404	IO-Link parameter name is not unique	Please use the [name]_[index] format. See 4.5.6
Data Storage errors			
401	400	Data storage mismatch	Mismatch between configured device and data storage meta data
Process Data handling errors			
501	400	I/Q is not configured as DIGITAL_OUTPUT	Writing processdata to I/Q is not possible
502	400	C/Q is not configured as DIGITAL_OUTPUT	Writing processdata to C/Q is not possible
503	400	IO-Link device has no output process data	
IODD errors			
601	400	IODD (representation) is not available	IODD representation for this IO-Link device is not available
602	500	IODD upload failed. IODD XML invalid	
603	400	Uploaded file is no valid IODD XML. Upload rejected	
604	400	IODD upload failed. CRC error	
605	400	IODD upload failed. Parsing error	
Data content errors			
701	400	Data set incomplete	
702	400	Data set not applicable	whole data set is rejected
703	400	Data set combination incompatible	whole data set is rejected

Annex B
(normative)

JSON base objects

B.1 General JSON objects

B.1.1 Cycle time object

Table 112 – 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 113 – Object Error

Object	**Error**			
Property	**Type**	**Value**	**Description**	**M/O/C**
`code`	number		Integer value	M
`text`	string		See A.2, See [1]	M
`iolinkError`	object	Object defined in Table 114 – Object iolinkError		O

Table 114 – Object iolinkError

Object	**Error**			
Property	**Type**	**Value**	**Description**	**M/O/C**
`code`	number		Integer value	M
`text`	string		See A.2, See [1]	M

Example:

```
{
    "code": 150,
    "text": "Permission denied"
}
```

B.1.3 Power Supply object

Table 115 – 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"
}
```

Annex C
(normative)

Path Paramters

C.1 Path Parameters

Table 116 – Path Parameters

Path Parameters			
Name	**Type**	**Value range**	**Description**
`masterNumber`	number	1 - 255	see 4.5.2
`portNumber`	number	1 - 255	see 4.5.3
`deviceAlias`	string		
`index`	number	0 - 65535	See [1]
`subindex`	number	0 - 255	See [1]
`parameterName`	string		IODD defines the maximum length of a name to 64 characters. Due to name conversion rules (see 4.5.6) the length can be extended.
`subParameterName`	string		See 4.5.6
`topicId`	number	Vendor specific	See Table 103 – Object MQTTtopic

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] http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[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

[8] IETF RFC 7231 https://tools.ietf.org/html/rfc7231

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

[10] https://io-link.com/share/Downloads/Profiles/IOL_ProfileIDOverview_V10_Mar2019.pdf

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/