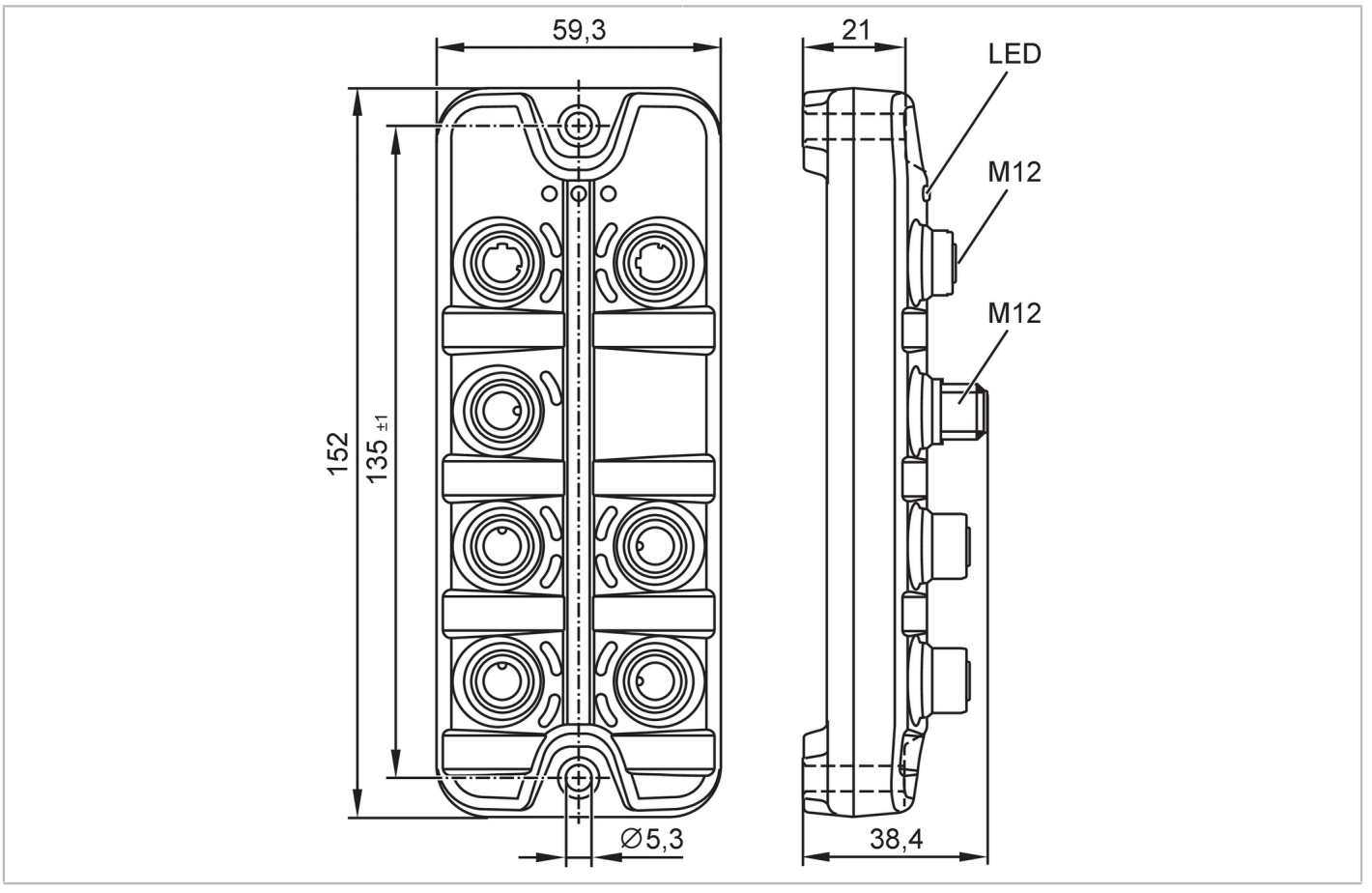
AL1351

IO-Link master with IoT interface

IO-Link Master DL IoT 4P IP69K



Application		
Application		hygienic systems; I/O modules for field applications
Daisy-chain function		fieldbus interface
Electrical data		
Operating voltage	[V]	20...28 DC; (US ; to SELV/PELV)
Current consumption	[mA]	300...3900; (US)
Protection class		III
Sensor supply US		
Max. current load total	[A]	3.6
Inputs / outputs		
Total number of inputs and outputs		8; (configurable)
Number of inputs and outputs		Number of digital inputs: 8; Number of digital outputs: 4
Inputs		
Number of digital inputs		8; (IO-Link Port Class A: 4 x 2)
Switching level high	[V]	11...30
Switching level low	[V]	0...5
Digital inputs protected against short circuits		yes

AL1351

IO-Link master with IoT interface

IO-Link Master DL IoT 4P IP69K

Outputs			
Number of digital outputs		4; (IO-Link Port Class A: 4 x 1)	
Max. current load per output	[mA]	300	
Short-circuit proof		yes	
Interfaces			
Communication interface		Ethernet; IO-Link	
Ethernet - Internet of Things			
Transmission standard		10Base-T; 100Base-TX	
Transmission rate		10 MBit/s; 100 MBit/s	
Protocol		MQTT JSON	
Factory settings		IP address: 169.254.x.x	
		subnet mask: 255.255.0.0	
		gateway IP address: 0.0.0.0	
		MAC address: see type label	
Note on interfaces		Safety protocol: HTTPS	
IO-Link Master			
Transmission type		COM1 (4,8 kBaud); COM2 (38,4 kBaud); COM3 (230,4 kBaud)	
IO-Link revision		1.1	
Number of ports class A		4	
Operating conditions			
Ambient temperature	[°C]	-25...60	
Storage temperature	[°C]	-25...85	
Max. relative air humidity	[%]	90	
Protection		IP 65; IP 66; IP 67; IP 69K; (operation with stainless steel protective caps: IP 69K)	
Protection rating (NEMA 250)		6P	
Pollution degree		2	
Chemical media		ISO 16750-5	HLP, CC, DB, DC, DD, CA
		NEMA 250 5.13.1	AA
Tests / approvals			
EMC		EN 61000-6-2	
		EN 61000-6-4	
Shock resistance		DIN EN 60068-2-27	
Vibration resistance		DIN EN 60068-2-64 2009-04	
		DIN EN 60068-2-6 2008-10	
MTTF	[years]	126	
Mechanical data			
Weight	[g]	304.5	
Housing		rectangular	
Type of mounting		Backplane mounting	
Dimensions	[mm]	152 x 59.3 x 38.4	
Material		housing: PA grey; Socket: stainless steel (1.4404 / 316L)	
Sealing material		EPDM	
Remarks			
Remarks		Further information is available in the manual.	

AL1351

IO-Link master with IoT interface

IO-Link Master DL IoT 4P IP69K

Pack quantity	1 pcs.

Electrical connection - Ethernet

Connector: 2 x M12; coding: D; Contacts: 4; sealing: EPDM

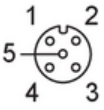


IoT X21, X22

1	TX +
2	RX +
3	TX -
4	RX -
5	not used

Electrical connection - IO-Link

Connector: 4 x M12; coding: A; Contacts: 4; sealing: EPDM



IO-Link Port Class A X01...X04

1	Sensor supply (US) L+
2	Digital input
3	Sensor supply (US) L-
4	C/Q IO-Link
5	not used

Electrical connection - voltage supply

Connector: 1 x M12; coding: A; Contacts: 4



X31

1	+ 24 V DC (US)
2	not used
3	GND (US)
4	not used

Operating instructions

IO-Link master with IoT interface
DataLine
4 ports
IP 65 / IP 66 / IP 67 / IP 69K

AL1351

Firmware: 3.1.x

English

80284129 / 04 10 / 2021

Contents

1 Preliminary note 5

1.1 Legal and copyright information 5
1.2 Purpose of the document 5
1.3 Explanation of Symbols 6
1.4 Change history 6

2 Safety instructions 7

2.1 General 7
2.2 Required background knowledge 7
2.3 Safety symbols on the device 7
2.4 IT security 8

3 Functions and features 9

4 Function 10

4.1 Communication, parameter setting, evaluation 11
4.1.1 IO-Link 11
4.1.2 Internet of Things (IoT) 11
4.1.3 Security mode 11
4.1.4 Parameter setting 11
4.1.5 Visual indication 11
4.2 Digital inputs 12
4.3 IO-Link supply 12

5 Mounting 13

5.1 Mount the device 13

6 Electrical connection 14

6.1 Notes 14
6.2 Connecting the IoT ports 15
6.3 IO-Link ports 16
6.3.1 Connecting IO-Link devices for Class A operation 16
6.4 Connecting the device 17

7 Operating and display elements 18

7.1 Overview 18
7.2 LED indicators 19
7.2.1 Status LEDs 19
7.2.2 IoT ports 19
7.2.3 IO-Link Ports (Class A) 20
7.2.4 Voltage supply 20

8 Set-up 21

9 Configuration 22

9.1 LR DEVICE 23
9.1.1 Remarks 24
9.1.2 IoT: Configure IP settings 25
9.1.3 IoT: Configure security mode 26
9.1.4 IoT: Configure the interface to LR AGENT or LR SMARTOBSERVER 27
9.1.5 IO-Link ports: Activate data transfer to LR AGENT or LR SMARTOBSERVER 28
9.1.6 IO-Link ports: Configure operating mode 29
9.1.7 IO-Link ports: Set the device validation and data storage 30
9.1.8 IO-Link ports: Configuration of fail-safe values 31
9.1.9 Info: Show device information 31
9.1.10 Firmware: Reset device to factory settings 32
9.1.11 Firmware: Reboot the device 32
9.1.12 Configure IO-Link devices 32
9.2 ifm IoT Core 34
9.2.1 Programmers' notes 35
9.2.2 First steps 39
9.2.3 General functions 39
9.2.4 IoT: Configuring access rights 43
9.2.5 IoT: Configuring IP settings 43
9.2.6 IoT: Configuring the LR AGENT or LR SMARTOBSERVER interface 44
9.2.7 IoT: Configuring security mode 44
9.2.8 IO-Link ports: Setting the operating mode of pin 4 (US) 47
9.2.9 IO-Link ports: Configuring device validation and data storage 47
9.2.10 IO-Link ports: Configuring data transfer to LR AGENT or LR SMARTOBSERVER 49
9.2.11 IO-Link ports: Reading / writing process data 49
9.2.12 IO-Link ports: Indicating port events 52
9.2.13 IO-Link devices: Accessing parameters 52
9.2.14 IO-Link devices: Reading an writing device information 54
9.2.15 IO-Link devices: Indicating IO-Link events 54
9.2.16 Gateway: Resetting, rebooting and localising the device 54
9.2.17 Gateway: Reading device information 55
9.2.18 Gateway: Reading status and diagnostic information 55
9.2.19 Gateway: Updating the firmware 56
9.2.20 Gateway: Setting the application tag 57
9.2.21 Subscribing to notifications 58
9.2.22 Using Web Socket 62
9.2.23 MQTT support 64
9.2.24 Using the IoT-Core Visualizer 68

10 Operation 75
10.1 Using web-based management 75

11 Maintenance, repair and disposal 76
11.1 Cleaning process 76
11.2 Updating the firmware 76
11.3 Replacing IO-Link device 76

12 Factory settings 77

13 Accessories 78

14 Appendix 79
14.1 Technical data 80
14.1.1 Application 80
14.1.2 Electrical data 80
14.1.3 Inputs / outputs 80
14.1.4 Inputs 81
14.1.5 Outputs 81
14.1.6 Interfaces 81
14.1.7 Operating conditions 82
14.1.8 Approvals / tests 82
14.1.9 Mechanical data 82
14.1.10 Electrical connection 83
14.2 ifm IoT Core 84
14.2.1 Overview: IoT profile 85
14.2.2 Overview: IoT types 92
14.2.3 Overview: IoT services 93

15 Index 107

1 Preliminary note

Content

Legal and copyright information 5
Purpose of the document 5
Explanation of Symbols 6
Change history 6

33203

1.1 Legal and copyright information

33117

© All rights reserved by ifm electronic gmbh. No part of this manual may be reproduced and used without the consent of ifm electronic gmbh.

All product names, pictures, companies or other brands used on our pages are the property of the respective rights owners:

- AS-i is the property of the AS-International Association, (→ www.as-interface.net)
- CAN is the property of the CiA (CAN in Automation e.V.), Germany (→ www.can-cia.org)
- CODESYS™ is the property of the CODESYS GmbH, Germany (→ www.codesys.com)
- DeviceNet™ is the property of the ODVA™ (Open DeviceNet Vendor Association), USA (→ www.odva.org)
- EtherNet/IP® is the property of the → ODVA™
- EtherCAT® is a registered trade mark and patented technology, licensed by Beckhoff Automation GmbH, Germany
- IO-Link® is the property of the → PROFIBUS Nutzerorganisation e.V., Germany (→ www.io-link.com)
- ISOBUS is the property of the AEF – Agricultural Industry Electronics Foundation e.V., Deutschland (→ www.aef-online.org)
- Microsoft® is the property of the Microsoft Corporation, USA (→ www.microsoft.com)
- Modbus® is the property of the Schneider Electric SE, France (→ www.schneider-electric.com)
- PROFIBUS® is the property of the PROFIBUS Nutzerorganisation e.V., Germany (→ www.profibus.com)
- PROFINET® is the property of the → PROFIBUS Nutzerorganisation e.V., Germany
- Windows® is the property of the → Microsoft Corporation, USA

1.2 Purpose of the document

34227

This document is only for device types "IO-Link master - IoT core gateway (DataLine) 4 port IP 65 / IP 66 / IP 67 / IP 69K" (art. no.: AL1351).

It is part of the device and contains information about the correct handling of the product.

► Read this document before using the device.

► Keep this document during the service life of the device.

1.3 Explanation of Symbols

34171

WARNING
Warning of serious personal injury.
Death or serious irreversible injuries may result.

CAUTION
Warning of personaly injury.
Slight reversible injuries may result.

NOTICE
Warning of damage to property

Important note
Non-compliance can result in malfunction or interference

Information
Supplementary note

► ... Request for action

> ... Reaction, result

→ ... "see"

abc Cross-reference

123 Decimal number
0x123 Hexadecimal number
0b010 Binary number

[...] Designation of pushbuttons, buttons or indications

1.4 Change history

42750

Version	Topic	Date
00	New creation of the document	04 / 2019
01	Correction of device status	05 / 2019
02	Correction: Technical data - current rating per output	09 / 2019
03	▪ Added: New IoT core functions ▪ Added: IoT Core Visualizer ▪ Correction: Description of the IoT Core Service getsubscriptioninfo	10 / 2020
04	Deleted: ifm IoT Core – DNS support	10 / 2021

2 Safety instructions

Content

General 7
Required background knowledge 7
Safety symbols on the device 7
IT security 8

28333

2.1 General

58525

- The device described is a subcomponent for integration into a system. The manufacturer is responsible for the safety of the system. The system manufacturer undertakes to perform a risk assessment and to create documentation in accordance with legal and normative requirements to be provided to the operator and user of the system. This documentation must contain all necessary information and safety instructions for the operator, the user and, if applicable, for any service personnel authorised by the manufacturer of the system.
- Read this document before setting up the product and keep it during the entire service life.
- The product must be suitable for the corresponding applications and environmental conditions without any restrictions.
- Only use the product for its intended purpose (→ **Functions and features** (→ S. 9)).
- If the operating instructions or the technical data are not adhered to, personal injury and/or damage to property may occur.
- The manufacturer assumes no liability or warranty for any consequences caused by tampering with the product or incorrect use by the operator.
- Installation, electrical connection, set-up, programming, configuration, operation and maintenance of the product must be carried out by personnel qualified and authorised for the respective activity.
- Protect units and cables against damage.

2.2 Required background knowledge

34185

This document is intended for specialists. Specialists are people who, based on their relevant training and experience, are capable of identifying risks and avoiding potential hazards that may be caused during operation or maintenance of the product.

The document contains information about the correct handling of the product.

2.3 Safety symbols on the device

34199



General warning
Observe instructions in chapter "Electrical connection" (→ **Electrical connection** (→ S. 14))!

2.4 IT security

54678

NOTICE!

If the device is operated in an unprotected network environment.

> Unauthorised read or write access to data is possible.

> Unauthorised manipulation of the device function is possible.

► Check and restrict access options to the device:

- Restrict access to authorised persons.
- Do not connect the device to open networks or the internet.

If access from the internet is inevitable:

► choose a safe method to connect with the device (e. g. VPN).

► Use encrypted data transmission (e. g. https / TLS).

3 Functions and features

58284

The device may only be used for the following purposes:

- as IO-Link master for configuration, management and operation of IO-Link devices
- as gateway between IO-Link devices and AS-i network

The device is designed for use without a control cabinet in the food industry.

► Only use the device within the limits of the technical data (→ **Technical data** (→ S. 80)).

4 Function

Content

Communication, parameter setting, evaluation 11
Digital inputs 12
IO-Link supply 12

33836

4.1 Communication, parameter setting, evaluation

Content

IO-Link 11
Internet of Things (IoT) 11
Security mode 11
Parameter setting 11
Visual indication 11

33860

4.1.1 IO-Link

34084

The device offers the following IO-Link functions:

- IO-Link master (IO-Link revision 1.0 and 1.1)
- 4 IO-Link ports for connection of IO-Link devices
- Provision of process data of the connected IO-Link devices for LR SMARTOBSERVER monitoring software (→ www.ifm.com)

4.1.2 Internet of Things (IoT)

54679

The device offers the following IoT functions:

- Gateway for the transmission of process, parameter and monitoring data between IO-Linkmaster / IO-Link devices and the IT network level
- REST-API to access process and parameter data
- Supported protocols: TCP/IP JSON, MQTT

4.1.3 Security mode

54697

The IoT interface offers the following optional sercurity functions:

- Secure data transfer via encrypted connection (Secure Layer Transport - TLS)
- Access protection via authentification

4.1.4 Parameter setting

34210

The device provides the following configuration options:

- Parameter setting of the IO-Link master of the AL1351 with LR DEVICE parameter setting software, IoT core projection software or ifm IoT-Core services.
- Parameter setting of the connected IO-Link devices (sensors, actuators) with LR DEVICE parameter setting software, IoT core projection software or ifm IoT-Core services
- Storage of parameter sets of the connected IO-Link devices for automatic recovery (data storage)

4.1.5 Visual indication

34192

The device has the following visual indicators:

- Status and error indication of the gateway, of the IoT core connection and of the system
- Status display of the voltage supply
- Status and activity display of the Ethernet connection

- Status, error and short circuit/overload indication of the IO-Link ports

4.2 Digital inputs

33817

The device has 4 additional digital inputs (type 2 according to EN 61131-2).

The digital inputs are on pin 2 of the ports X01...X04.

All inputs refer to the potential of the device supply (pin 3).

4.3 IO-Link supply

34077

The device has 4 supplies for IO-Link devices.

The IO-Link ports X01...X04 are ports class A.

Every supply provides short circuit monitoring.

The device ensures fire protection for the connected IO-Link devices by providing a power-restricted circuit at the IO-Link ports (according to IEC61010-1 and Class 2 according to UL1310).

5 Mounting

Content

Mount the device 13

34058

5.1 Mount the device

34059

- ► Disconnect the system from power before installation.
- ► For installation choose a flat mounting surface.
- ► Please observe the maximum tightening torque.

► Fix the unit to the mounting surface using 2 M5 mounting screws and washers.
  - Tightening torque: 1.8 Nm

► Ground the unit via the two mounting screws of the upper mounting lugs.

6 Electrical connection

Content

Notes 14
Connecting the IoT ports 15
IO-Link ports 16
Connecting the device 17

33805

6.1 Notes

51957

A qualified electrician must connect the unit.

► The national and international for regulations setting up electrical equipment must be complied with.

The unit is only suitable for operation using SELV/PELV voltages.

► Please note the information concerning IO-Link wiring!

This unit contains components that may be damaged or destroyed by electrostatic discharge (ESD).

► Please observe the required precautions against electrostatic discharge!

The M12 connection parts in the device comply with the ingress resistance requirements of the standard EN 61076-2-101. To adhere to the protection rating, only cables certified to this standard must be used. The system creators undertake to ensure ingress resistance for cables which they cut to length themselves.

► Carry out the fitting according to the indications of the cable manufacturer. A maximum of 0.8 Nm is permitted.

► During installation, place the M12 connector vertically so that the coupling nut will not damage the thread.

► Depending on the mounting conditions, cables must be provided with a strain relief to avoid unacceptable loads on the mounting points and M12 connections.

► Make sure that the M12 connection parts are correctly seated and mounted correctly. The specified protection rating can not be guaranteed if this is not observed.

For UL applications:

► To connect the IO-Link master and the IO-Link devices, only use UL-certified cables of the CYJV or PVVA category with a minimum temperature of 80 °C (75 °C in case of maximum ambient temperature of 40 °C).

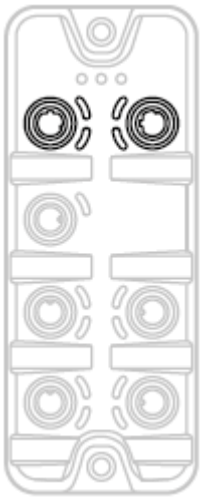
Wiring: → **Technical data** (→ S. 80)

By means of basic insulation according to EN61010-1, the circuits are separated from each other and from device surfaces that could be touched (secondary circuit with 28 V DC maximum, supplied from mains circuit up to 300 V overvoltage category II).

By means of basic insulation according to EN61010-1, the communication interfaces are separated from each other and from device surfaces that could be touched (secondary circuit with 28 V DC maximum, supplied from mains circuit up to 300 V overvoltage category II). They are designed for network environment 0 according to IEC TR62102.

6.2 Connecting the IoT ports

52272



- ► Connect the unit via the M12 socket X21 and/or X22 with the IoT core network (e.g. laptop/PC with LR DEVICE parameter setting software, laptop/PC with LR SMARTOBSERVER monitoring software, PC/laptop with http request compatible software)
- ► To connect the devices, use M12 connectors with protection rating IP 65 / IP 66 / IP 67 / IP 69K or higher (→ **Accessories** (→ S. 78)).
- ► Cover unused sockets with M12 protective caps (art. no.: E12542).

6.3 IO-Link ports

51958

The IO-Link ports of the AL1351 meet the requirements of the IO-Link specifications 1.0 to 1.1.2.

► Please note the information concerning IO-Link wiring!

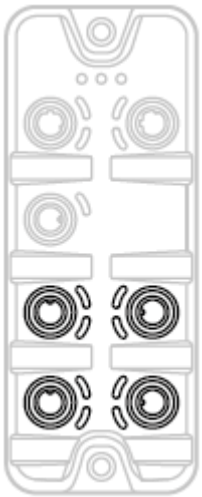
► Cover unused sockets with M12 protective caps (art. no.: E12542).

6.3.1 Connecting IO-Link devices for Class A operation

51959

Wiring information:

- The connected IO-Link devices must be supplied exclusively via the IO-Link master.
- The additional digital inputs IO-Link ports X01...X04 (pin 2) have a type 2 behaviour according to the standard EN61131-2. The connected electronics must be electrically suited for this.



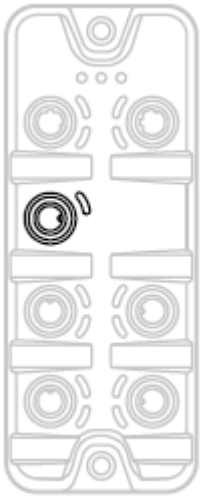
► Connect the connectors of the IO-Link devices with the M12 sockets of the IO-Link ports X01...X04.

  ▪ Maximum cable length per IO-Link port: 20 m

► To connect the devices, use M12 connectors with protection rating IP 65 / IP 66 / IP 67 / IP 69K or higher (→ **Accessories** (→ S. 78)).

6.4 Connecting the device

33882



- ► Disconnect power.
- ► Connect the IO-Link Master via M12 socket X31 to 24 V DC (20...28 V SELV/PELV; according to EN61010-1, secondary circuit with maximum 30 V DC supplied by mains circuit up to 300 V of overvoltage category II).
  - Maximum cable length: 25 m
- ► To connect the device, use M12 connectors with protection rating IP 65 / IP 66 / IP 67 / IP 69K or higher (→ **Accessories** (→ S. 78)).

When using cable length greater than 25 m keep in mind the voltage drop as well as the required minimum voltage supply of 20 V!

7 Operating and display elements

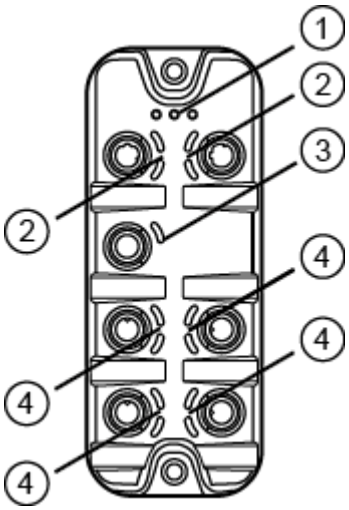
Content

Overview....18
LED indicators....19

34063

7.1 Overview

52274



1. Status LEDs RDY and IoT
   → **Status LEDs** (→ S. 19)
2. L/A and 10/100 MBits/s status LEDs of the IoT interface (X21/X22)
   → **IoT ports** (→ S. 19)
3. IOL and DI status LEDs of the IO-Link port (X01...X04)
   → **IO-Link Ports (Class A)** (→ S. 20)
4. PWR status LED of the voltage supply (X31)
   → **Voltage supply** (→ S. 20)

7.2 LED indicators

34047

The device only has the following LED indicators:

7.2.1 Status LEDs

52237

The RDY LED indicates the status of the gateway.
The IoT LED indicates the status of the connection to the LR SMARTOBSERVER.

Status LED			Description
RDY	green	on	Status: OK
		flashes 5 Hz	Status: Error
		flashes (200 ms on, 800 ms off)	Status: Firmware update is running
		off	Status: Gateway not running or gateway booting
IoT	green	off	No connection to the LR SMARTOBSERVER
		on	Connection to the LR SMARTOBSERVER activated

7.2.2 IoT ports

52238

Each IoT port has the 2 L/A and 10/100 MBits/s LEDs. The LEDs indicate the status of the Ethernet connection.

Status LED			Description
L/A	green	on	Ethernet connection established
		flashes	Data is transmitted via the Ethernet interface.
		off	No Ethernet connection
10/100 Mbits/s	yellow	on	100 MBit/s
		off	10 MBit/s

7.2.3 IO-Link Ports (Class A)

34074

Each IO-Link Port Class A has 2 LEDs labelled IOL and DI. The LEDs indicate the status of the IO-Link port.

Status LED			Description
IOL	yellow	Off	Port configured as DI/DO: pin 4 (C/Q) = OFF
		on	Port configured as DI/DO: pin 4 (C/Q) =ON
	green	flashing 1 Hz	Port configured asIO-Link: no IO-Link device found
		Flashing with 2 Hz	Port configured asIO-Link: Status PREOPERATE
		on	Port configured asIO-Link: Status OPERATE
	red	Flashing with 2 Hz	Port configuration error or short circuit / overload on US
		on	Transmission Error
DI	yellow	Off	Digital input: pin 2 = OFF
		on	Digital input: pin 2 = ON

7.2.4 Voltage supply

34191

The interface for voltage supply (X31) has the LED that is marked as US. The LED indicates the status of the voltage supply.

Status LED			Description
US	green	on	The supply voltage Us is applied.
		off	No supply voltage is applied or the applied supply voltage is too low.

8 Set-up

52239

When the supply voltage is switched on, the AL1351 starts with the factory settings. The display elements signal the current operating mode (→ **Operating and display elements** (→ S. 18)).

To enable parameter setting of the AL1351 via the IoT core network, the IoT interface must be configured according to the network environment.

► Connect AL1351 via the ports X21/X22 to the IoT core network.

► Configure the IoT interface (→ **IoT: Configure IP settings** (→ S. **Fehler! Textmarke nicht definiert.**)).

> IoT interface has valid IP settings.

> User can set the parameters of the AL1351.

Further steps:

- Set the parameters of the AL1351 (→ **Configuration** (→ S. 22)).

9 Configuration

Content

LR DEVICE....23
ifm IoT Core....34

33858

9.1 LR DEVICE

Content

Remarks 24
IoT: Configure IP settings .... **Fehler! Textmarke nicht definiert.**
IoT: Configure security mode 26
IoT: Configure the interface to LR AGENT or LR SMARTOBSERVER 27
IO-Link ports: Activate data transfer to LR AGENT or LR SMARTOBSERVER 28
IO-Link ports: Configure operating mode 29
IO-Link ports: Set the device validation and data storage....30
IO-Link ports: Configuration of fail-safe values 31
Info: Show device information 31
Firmware: Reset device to factory settings 32
Firmware: Reboot the device....32
Configure IO-Link devices 32

33692

On delivery, the AL1351 is configured with the factory settings (→ **Factory settings** (→ S. 77)).

Required software: LR DEVICE (1.5.0.x or higher) (art.-no.: QA0011/QA0012)

9.1.1 Remarks

Content

Offline parameter setting 24
VPN connection 24

34180

Offline parameter setting

34060

The AL1351 supports the offline parameter setting. In this context, the user creates and stores a configuration for the IO-Link master and the connected IO-Link devices without being connected to the AL1351 (OFFLINE mode). The configuration created in this way can be stored as a file (*.lrp) and loaded to the AL1351 and activated at a later date.



Further information about offline parameter setting: → Operating instructions LR DEVICE

VPN connection

34382



An active VPN connection blocks the access of the parameter setting software LR DEVICE to the IoT core interface of the AL1351.

► Deactivate the VPN connection in order to be able to access the AL1351 with the LR DEVICE.

9.1.2 IoT: Configure IP settings

34049

For access to the IO-Link master via the IT infrastructure the user has to set the IP settings of the IoT port.

To configure the IP settings with DHCP, a DHCP server has to be active in the IT network. If no DHCP server can be reached in the IT network, an IP address is automatically assigned to the IoT port with the Zeroconfig protocol (address range: → **Factory settings** (→ S. 77)).

To configure the IP settings of the IoT interface:

► Select [IoT] menu.

> The menu page shows the current settings.

► Set the following parameters as required:

Name	Description	Possible values	
[DHCP]	Activate/deactivate the DHCP client of the device	[Static IP]	IP settings were set by the user
		[DHCP]	IP settings are set by a DHCP server in the network.
[IP address]*	IP address of the IoT port	Factory setting: 169.254.X.X	
[Subnet mask]*	Subnet mask of the Ethernet network	Factory setting: 255.255.0.0	
[Default gateway IP address]*	IP address of the network gateway	Factory setting: 0.0.0.0	
[MAC address]	MAC address of the IoT port	The value is firmly set.	

* ... can only be edited if parameter [DHCP] = [Static IP]

► Save changed values on the device.

9.1.3 IoT: Configure security mode

54680

The IoT interface of the IO-Link offers a security mode. It enables secure data transmission via transport encryption and restriction of the access to IO-Link masters and IO-Link devices via user authentication.

To configure the security mode:

► Select [IoT] menu.

> The menu page shows the current settings.

► Set the following parameters as required:

Name	Description	Possible values	
[Security mode HTTPS]	Set the security mode	[Disabled]	Security mode disabled
		[Enabled]	Security mode enabled
[Security password]	Password Note: The set password is not displayed.		

► Save changed values on the device.

The security mode only protects the access to the device via the IoT interface.
The user name "administrator" cannot be changed.

The security mode can be enabled without setting the password. During the attempt to write to the device, LR DEVICE requires to enter and confirm the password.

After entering the password, the user has unrestricted access to IO-Link masters and connected IO-Link devices. The password will only be requested again if the current LR DEVICE session is over (e. g. after restarting the LR DEVICE).

To change the set password:

► Sign in with a valid password.

► Enter the new password in the field [Security password].

► Write changes to the device.

> The new password is set.

9.1.4 IoT: Configure the interface to LR AGENT or LR SMARTOBSERVER

34048

To enable transfer of process data from the IO-Link master to LR AGENT or LR SMARTOBSERVER, the interface has to be configured accordingly.

► Select [IoT] menu.

> The menu page shows the current settings.

► Set the following parameters as required:

Name	Description	Possible values	
[IP address LR Agent or SMARTOBSERVER]	IP address of LR AGENT or LR SMARTOBSERVER	Factory setting: 255.255.255.255	
[Port LR Agent or SMARTOBSERVER]	Port number that is used to send process data to LR AGENT or LR SMARTOBSERVER	0 ... 65535	Factory setting:: 35100
[Interval LR Agent or SMARTOBSERVER]	Cycle time for the transfer of the process data to LR AGENT or LR SMARTOBSERVER (value in milliseconds)	[Off]	no transfer
		500 ... 2147483647	500 ms ... 2147483647 ms
[Application Tag]	Source identifier of the IO-Link master in the structure of LR AGENT or LR SMARTOBSERVER (String32)	Factory setting: AL1351	

After changing the parameter [Port LR Agent or SMARTOBSERVER] or [Application Tag], it may take 120 seconds before the device establishes a new TCP connection.

To prevent the delay:

► Reboot the device after changing the the parameter.

► Save changed values on the device.

9.1.5 IO-Link ports: Activate data transfer to LR AGENT or LR SMARTOBSERVER

33690

The user can decide separately for each IO-Link port whether the process data of the connected IO-Link devices should be transferred to LR AGENT or LR SMARTOBSERVER.

To transfer process data the interface to the LR AGENT or LR SMARTOBSERVER has to be correctly configured (→ **IoT: Configure the interface to LR AGENT or LR SMARTOBSERVER** (→ S. 27)).

To activate / deactivate data transfer:

► Select [Port x] menu (x = 1...4).

> The menu page shows the current settings.

► Set the following parameters as required:

Name	Description	Possible values	
[Transmission to LR Agent or SMARTOBSERVER]	Transfer of process data of the connected IO-Link device to LR AGENT oder LR SMARTOBSERVER	[Disabled]	Transfer process data
		[Enabled]	Don't transfer process data

► Save changed values on the device.

9.1.6 IO-Link ports: Configure operating mode

33694

The IO-Link ports X01...X04 of the device support the following operating modes:

- Disabled: no data transfer at pin 4 (C/Q) of the IO-Link port
- Digital input (DI): binary input signal at pin 4 (C/Q) of the IO-Link port
- Digital output (DO): binary output signal at pin 4 (C/Q) of the IO-Link port
- IO-Link: IO-Link data transfer via pin 4 (C/Q) of the IO-Link port

The user can set the operating mode separately for each IO-Link port.
To set the operating mode of an IO-Link port:

► Select [Port x] menu (x = 1...4).

> The menu page shows the current settings.

► Set the following parameters as required:

Name	Description	Possible values	
[Mode Pin4 US]	Operating mode of pin 4 of the IO-Link port	[Disabled]	Port deactivated
		[DI]	Operation as digital input
		[DO]	Operation as digital output
		[IO-Link]	Operation as IO-Link interface
[Cycle time actual]**	Current cycle time of the data transfer between IO-Link master and IO-Link device on the port (value in microseconds)	Parameter can only be read	
[Cycle time preset]*	Cycle time of the data transfer between the IO-Link master and the IO-Link device at the port (value in microseconds)	0	The device automatically sets the fastest possible cycle time.
		1 ... 132800	1 microsecond ... 132800 microseconds
[Bitrate]**	Current transmission rate of the data transfer between the IO-Link master and the IO-Link device on the port	Parameter can only be read	

* ... Parameter only available if [Mode] = [IO-Link]
** ... Parameter only visible if the IO-Link device is connected to the IO-Link port.

► Save changed values on the device.

9.1.7 IO-Link ports: Set the device validation and data storage

33697

The user can choose how the IO-Link ports are to behave with regard to the device validation and the storage / recovery of parameter data of the connected IO-Link device.

The following options are available:

Option	Validation of the IO-Link device	Storage of the parameter values	Recovery of the parameter values
[No check and clear]	no	no	no
[Type compatible V1.0 device]	yes, test the compatibility with IO-Link standard V1.0	no	no
[Type compatible V1.1 device]	yes, test the compatibility with IO-Link standard V1.1	no	no
[Type compatible V1.1 device with Backup + Restore]	yes, test the compatibility with IO-Link standard V1.1 and identity of design (vendor ID and device ID)	yes, automatic storage of the parameter values; changes of the current parameter values will be stored	yes, recovery of the parameter values when connecting an identical IO-Link device with factory settings
[Type compatible V1.1 device with Restore]	yes, test the compatibility with IO-Link standard V1.1 and identity of design (vendor ID and device ID)	no, there is no automatic storage changes of the current parameter values will not be stored	yes, recovery of the parameter values when connecting an identical IO-Link device with factory settings

The options only apply if the IO-Link port is in the operating mode "IO-Link".

For options [Type compatible V1.1 device with Backup + Restore] and [Type compatible V1.1 device with Restore]: If the vendor ID and device ID are changed in the online mode, the data memory will be deleted and a new backup of the parameter values of the connected IO-Link device will be created in the IO-Link master.

To configure the device validation and the data storage:

► select [Port x] menu (x = 1...4).

> The menu page shows the current settings.

► Set the following parameters as required:

Name	Description	Possible values	
[Validation / Data Storage]	Supported IO-Link standard and behaviour of the IO-Link master when connecting a new IO-Link device at port x (x = 1...4)	[No check and clear]	
		[Type compatible V1.0 device]	
		[Type compatible V1.1 device]	
		[Type compatible V1.1 device with Backup + Restore]	
		[Type compatible V1.1 device with Restore]	
[Vendor ID]	ID of the manufacturer that is to be validated	0...65535	Factory setting: 0# ifm electronic: 310
[Device ID]	ID of the IO-Link device that is to be validated	0...16777215	Factory setting: 0

► Save changed values on the device.

9.1.8 IO-Link ports: Configuration of fail-safe values

52242

The user can set the fail-safe values of the outputs of the IO-Link ports X01...X04. The fail-safe values will be activated in case of an interruption of the IoT core connection.

To configure the fail-safe values:

► Select [Port x] menu (x = 1...4).

> The menu page shows the current settings.

► Set the following parameters as required:

Name	Description	Possible values	
[Fail-safe digital out]*	Fail-safe value of the output for operating mode "digital output (DO)"	[Reset]	OFF
		[Old]	Old value
		[Set]	ON
[Fail-safe IO-Link]*	Fail-safe value of the output for operating mode "IO-Link"	[Off]	No fail-safe
		[Reset]	Fail-safe: OFF
		[Old]	Fail-safe: old value
		[Pattern]	Fail-safe: byte sequence

* ... parameter can only be changed if the IoT core controller is disconnected

► Save changed values on the device.

9.1.9 Info: Show device information

34065

To read the general information of the ifm IO-Link master:

► Select [Info] menu.

> The menu page shows the current settings.

Name	Description	Possible values
[Product code]	Article number of the IO-Link master	AL1351
[Device family]	Device family of the IO-Link master	IO-Link master
[Vendor]	Vendor	ifm electronic gmbh
[SW-Revision]	Firmware of the IO-Link master	
[HW revision]	Hardware version of the IO-Link master	
[Bootloader revision]	Bootloader version of the IO-Link master	
[Serial number]	Serial number	

9.1.10 Firmware: Reset device to factory settings

33838

When the IO-Link master is reset, all parameters are set to the factory settings:

To reset the device to factory settings:

► Select [Firmware] menu.

> The menu page shows the current settings.

► Click on [Factory Reset] to reset the device.

> LR DEVICE sets the device to the factory settings.

9.1.11 Firmware: Reboot the device

33832

When rebooting the device, all settings are kept.

To restart the AL1351:

► Select [Firmware] menu.

> The menu page shows the current settings.

► Click on [Reboot] to reboot the device.

> LR DEVICE reboots the ifm IO-Link master.

9.1.12 Configure IO-Link devices

52351

To configure the IO-Link devices connected to the device with the LR DEVICE parameter setting software:

Requirements:

> IO-Link master is correctly installed and connected to the LR DEVICE parameter setting software.

> The IO-Link device is correctly connected to the AL1351.

> Operating mode of the IO-Link port is "IO-Link" (→ **IO-Link ports: Configure operating mode** (→ S. 29)).

1 Select IO-Link master

► Start LR DEVICE.

► Update IODD file library
OR:
Import IODD file of the IO-Link device manually.

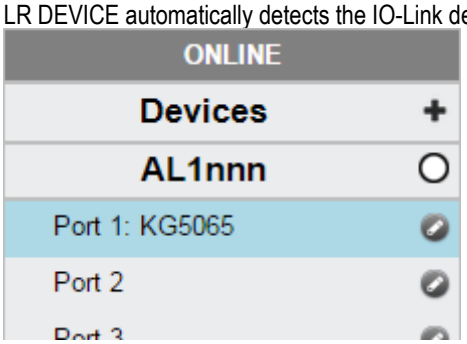
► Scan network for devices.

> LR DEVICE detects IO-Link master.

2 Add IO-Link device

► Under [ONLINE]: Click on the required IO-Link master.

> LR DEVICE automatically detects the IO-Link devices connected to the IO-Link master (e.g. ifm sensor KG5065).



3 Configure IO-Link device

► Mouse click on the port to which the IO-Link device is connected.

> LR DEVICE reads and shows the current parameter values of the IO-Link device.

► Configure IO-Link device.

Information about the available parameters of the IO-Link device: → IO Device Description (IODD) of the IO-Link device

► Save the changed configuration on the IO-Link device.

9.2 ifm IoT Core

Content

Programmers' notes 35
First steps 39
General functions 39
IoT: Configuring access rights 43
IoT: Configuring IP settings 43
IoT: Configuring the LR AGENT or LR SMARTOBSERVER interface 44
IoT: Configuring security mode 44
IO-Link ports: Setting the operating mode of pin 4 (US) 47
IO-Link ports: Configuring device validation and data storage 47
IO-Link ports: Configuring data transfer to LR AGENT or LR SMARTOBSERVER 49
IO-Link ports: Reading / writing process data 49
IO-Link ports: Indicating port events 52
IO-Link devices: Accessing parameters 52
IO-Link devices: Reading an writing device information 54
IO-Link devices: Indicating IO-Link events 54
Gateway: Resetting, rebooting and localising the device 54
Gateway: Reading device information 55
Gateway: Reading status and diagnostic information 55
Gateway: Updating the firmware 56
Gateway: Setting the application tag 57
Subscribing to notifications 58
Using Web Socket 62
MQTT support 64
Using the IoT-Core Visualizer 68

52244



General notes on the ifm IoT Core: → **Programmers' notes** (→ S. 35)

9.2.1 Programmers' notes

Content

IoT Core: General information 35
Access the ifm IoT Core 36
IoT Core: Diagnostic codes 38

34229

IoT Core: General information

52256

The DataLine device family has an IoT Core. The IoT Core allows the user to address the AL1351 from IT networks via a REST API and to integrate it into Internet-of-Things applications.

A device description is stored on the AL1351. This device description is a structured, machine-readable data object in JSON format. All current values of parameters, process data, diagnostic data and device information are mapped in this data object. These data values can be read and changed by means of services.

Access the ifm IoT Core

52257

The user can access the ifm IoT Core via HTTP requests. The following request methods are available.

GET request

33804

Using the GET method the user has read access to a data point.

The syntax of the request to the IoT Core is:

```
http://ip/datapoint/service
```

Parameter	Description
ip	IP address of the IoT interface
data_point	Data point which is to be accessed
service	Service

The syntax of the return of the IoT Core is:

```
{
"cid":id,
"data":{"value":resp_data},
"code":diag_code
}
```

Parameter	Description
id	Correlation ID for the assignment of request and return
resp_data	Value of the data point; depending on the data type of the data point
diag_code	Diagnostic code (→ **IoT Core: Diagnostic codes** (→ S. 38))

Example: GET request

54033

Request (via browser):

```
http://192.168.0.250/devicetag/applicationtag/getdata
```

Response:

```
{
"cid":-1,
"data":{"value":"AL1351"},
"code":200
}
```

POST request

54700

Using a POST request the user has read and write access to a data point.

The syntax of the request to the IoT Core is:

```
{
"code":"code_id",
"cid":id,
"adr":"data_point/service",
"data":{req_data},
"auth":{"user":"usr_id","passwd":"password"}
}
```

Field	Parameter	Description	
code	code_id	Service class	
		▪ request	Request
		▪ transaction	Transaction
		▪ event	Event
cid	id	Correlation ID for the assignment of request and response; ID freely assignable by the user	
adr	data_point	Data point of the element tree which is to be accessed	
	service	Service to be performed (→ **Overview: IoT services** (→ S. 93))	
data*	req_data	Data to be transferred to the IoT Core (e.g. new values); syntax depending on the service	
auth**	usr_id	user name (base64 coded); default value: administrator	
	password	password (base64 coded)	

* = optional; only required for services, that submit data to the IoT core (e. g. setdata)
** = optional; only required, if security mode is activated

The syntax of the return of the IoT Core is:

```
{
"cid":id,
"data":{resp_data},
"code":diag_code
}
```

Field	Parameter	Description
cid	id	Correlation ID for the assignment of request and response (see request)
data*	resp_data	Value of the data point; syntax depending on the service
code	diag_code	Diagnostic code (→ **IoT Core: Diagnostic codes** (→ S. 38))

* = optional; only required for services, that receive data from the IoT core (e.g. getdata)

Example: POST request

54035

Request:

```
{
"code":"request",
"cid":4711,
"adr":"devicetag/applicationtag/getdata"
}
```

Response:

```
{
"cid":4711,
"data":{"value":"AL1351"},
"code":200
}
```

IoT Core: Diagnostic codes

54688

Code	Text	Description
200	OK	Request successfully processed
230	OK but needs reboot	Request successfully processed; IO-Link master must be restarted
231	OK but block request not finished	Request successfully processed; blockwise request, but not yet finished
232	Data has been accepted, but internally modified	New values have been accepted, but were adjusted by the IO-Link master (Master cycle time)
233	IP settings (of IoT-Port) have been updated. Application needs to reload device. Wait at least 1 second before reloading device.	IP settings have been successfully changed, IO-Link master will be reloaded; wait for at least 1 second
400	Bad request	Invalid request
401	Unauthorized	Non authorised request
403	Forbidden	Forbidden request
500	Internal Server Error	Internal fault
503	Service Unavailable	The service is not available (e. g. IO-Link port in wrong operating mode; no IO-Link device at IO-Link port)
530	The requested data is invalid	Invalid process data
531	IO-Link error	Error in IO-Link Master / device
532	PLC connected Error	Error while setting data, because IO-Link master is still connected to fieldbus PLC

9.2.2 First steps

52245

To read the device description of the AL1351:

► Send the following POST request to the AL1351:
`{"code":"request","cid":-1,"adr":"gettree"}`

> AL1351 returns the device description as structured JSON object.

► Identify all substructures and the data points contained therein in the tree structure of the JSON object.

► Identify the applicable services for the access to substructures and the data points contained therein.

9.2.3 General functions

61148

The AL1351 has the type `device` (→ **Overview: IoT types** (→ S. 92)).

The following services can be used on the root element of the type `device`:

Service	Description
../gettree	Provide the complete tree or subtree of the device description (JSON)
../getidentity	Reading device information
../getdatamulti	Reading several parameter values sequentially
../getelementinfo	Reading detailed information of an element
../getsubscriberlist	Print a list of all active notification subscriptions
../querytree	Search device description for specific elements

Depending on the read and write access rights, the following services can be applied to elements of type `data`:

Service	Description
../getdata	Reading the value of the element
../setdata	Write the value of the element

Example: Reading properties of an element

59782

Task: Determine the data type and value range of the accessrights parameter.

Solution: Read the properties of the element `iotsetup/accessrights` of the getelementinfo service. The fields type (data type) and valuation (range of values) contain the required information.

- Request:

```
{
"code":"request",
"cid":4711,
"adr":"getelementinfo",
"data":{"adr":"iotsetup/accessrights"}
}
```

- Response:

```
{
"cid":4711,
"data":{
"identifier":"accessrights",
```

```
"type":"data",
"uid":null,
"profiles":["parameter"],
"format":{
"type":"enum",
"namespace":"json",
"encoding":"integer",
"valuation":{
"valuelist":{
"0":"Fieldbus + IoT",
"1":"Fieldbus + IoT (read-only)",
"3":"IoT only"}}}},
"code":200
}
```


The accessrights parameter has the data type ENUM with the valid values "Fieldbus + IoT", "Fieldbus + IoT (read only)" and "IoT only".

Example: output subtree

61149

Task: Output all direct sub-elements of the node `firmware`.

Solution: Use the service gettree to output the required subtree (root node: firmware, sub-levels to be shown: 1)

- Request:


```
{
"code":"request",
"cid":4711,
"adr":"gettree",
"data":{
"adr":"firmware",
"level":1}
}
```


- Response:


```
{
"cid":4711,
"data":{
"identifier":"firmware",
"type":"structure",
"profiles":[
"software","software/uploadablesoftware"],
"subs":[
{
"identifier":"version","type":"data","profiles":["parameter"],
"format":{"type":"string","namespace":"json","encoding":"UTF-8"}},
{
"identifier":"type","type":"data",
"format":{"type":"string","namespace":"json","encoding":"UTF-8"}},
{
"identifier":"install","type":"service"},
{
"identifier":"factoryreset","type":"service"},
{
"identifier":"signal","type":"service"},
{
"identifier":"container","type":"data",
"format":{"type":"binary","namespace":"json","encoding":"base64"}},
```

```
{
"identifier":"reboot","type":"service"}]
},
"code":200
}
```


Example: Read several parameter values of the IO-Link master simultaneously

33840

Task: The following current values are to be read by the IO-Link master: temperature, serial number

Solution: Read the current parameter values using the getdatamulti service (data point temperature: /processdatamaster/temperature; data point serial number: /deviceinfo/serialnumber)

- Request:


```
{
"code":"request",
"cid":4711,
"adr":"/getdatamulti",
"data":{"datatosend":["/processdatamaster/temperature","/deviceinfo/serialnumber"]
}
}
```


- Response:


```
{
"cid":4711,
"data":{"processdatamaster/temperature":{"code":200,"data":44},
"deviceinfo/serialnumber":{"code":200,"data":"000174210147"}},
"code":200
}
```


Example: Browsing device description

61150

Task: List all elements with the designation "status" and the profile "runcontrol".

Solution: Use the service querytree to browse the device description with the parameters "status" (name) and "runcorntrol" (profile)

- Request:


```
{
"cid":4711,
"adr":"querytree",
"code":"request",
"data":{
"profile":"runcontrol",
"name":"status"}
}
```


- Response:


```
{
"cid":4711,
"data":{
"adrList":[
"device/connections/mqttConnection/status",
"device/connections/mqttConnection/mqttCmdChannel/status"]},
"code":200
}
```

Setting the storage duration

61153

The IoT Core offers the possibility to set the storage duration of data and notifications. The Services **Service: setdata** (→ S. 102) and **Service: subscribe** (→ S. 105) therefore have the parameter "duration".

Example: Subscribing to notifications

61154

Task: The current values of the following parameters are to be sent regularly to a network server with IP address 192.168.0.4:

- Product name of the IO-Link Devices an IO-Link port X02
- Cyclic input data of the IO-Link Devices an IO-Link port X02
- Operating temperature of the IO-Link master.

The subscription is only to be active until the next restart of the IO-Link master.

Solution: Subscribe to the required data using the subscribe service.

- Request:

```
{
"code":"request",
"cid":4711,
"adr":"/timer[1]/counter/datachanged/subscribe",
"data":{
"callback":"http://192.168.0.4:80/temp",
"datatosend":[
"/iolinkmaster/port[2]/iolinkdevice/productname",
"/iolinkmaster/port[2]/iolinkdevice/pdin",
"/processdatamaster/temperature"],
"duration":"uptime"}
}
```

- Response:

```
{
"cid":4711,
"code":200
}
```

9.2.4 IoT: Configuring access rights

59785

Substructure: `iotsetup`

Available data points:

Name	Description	Access
../accessrights	Access rights to the IO-Link master	rw

rw ... read and write

If in IoT and IoT core projection software the parameter [Access Rights] is = [IoT core + IoT], the parameter values set in the IoT core projection software will always apply.

If in IoT the parameter [Access Rights] is = [IoT only], set the parameter [Access Rights] = [Keep settings] in the IoT core projection software.

If in LR DEVICE the parameter [Access Rigts] is = [EtherCAT + IoT (read-only)], write access to the device configuration via LR DEVICE and IoT core services is blocked. To enable write access again, set the parameter to [EtherCAT + IoT] via fieldbus configuration software.

Changes of the parameter [Access Rights] will only be effective after restarting the IO-Link master (→ **Firmware: Reboot the device** (→ S. 32)).

9.2.5 IoT: Configuring IP settings

61155

Substructure: `iotsetup`

Available data points:

Name	Description	Access
../network/dhcp	Configuration of the IP settings of the IoT port	rw
../network/ipaddress	IP address of the IoT port	rw
../network/subnetmask	Subnet mask of the network segment	rw
../network/ipdefaultgateway	IP address of the network gateway	rw

rw ... read and write

Applicable services:

Name	Description
../network/setblock	Write all values of the substructure blockwise

Change the IP parameters in the substructure network only blockwise with the setblock service!

9.2.6 IoT: Configuring the LR AGENT or LR SMARTOBSERVER interface

59786

Substructure: `iotsetup`

Avalable data points:

Name	Description	Access
../smobip	IP address of the LR SMARTOBSERVER	rw
../smobport	Port number of the LR SMARTOBSERVER	rw
../smobinterval	Cycle time for data transmission to LR SMARTOBERVER (value in milliseconds)	rw

rw ... read and write

9.2.7 IoT: Configuring security mode

54683

The access to the IoT interface of the IO-Link master can be protected with a security mode:

Substructure: `iotsetup`

Available data points:

Name	Description	Access
../security/securitymode	active security mode	rw
../security/password	Password for authentication (Base64 coded)	w

rw ... read and write
w ... write only

Valid character set for the Base64 coding / decoding of the password: UTF-8
Online tool for coding / decoding: → www.base64encode.org

Note: Security mode

54684

The security mode enables restricting access to the IO-Link master and the connected IO-Link devices from the IT network. In the activated security mode, the following restrictions apply:

- Access only with authentication (password-protected user account)
- Access only via secure https connection (Transport Layer Security - TLS)

The security mode only protects the access to the device via the IoT interface.
The standard value for users is: administrator
The set password cannot be read with getdata.

The current status of the security function can be read with the getidentity service (→ **Servicet: getidentity** (→ S. 96)).

For the authentication, the user must additionally provide the POST requests with a valid user name and password in the field "auth". The user name and the password will be shown as Base64-coded character strings (→ **Example: Request with authentication** (→ S. 46)).

The following requests can be done if the security mode is enabled, also without authentication:

- /getidentity
- /deviceinfo/vendor/getdata
- /deviceinfo/productcode/getdata

Example: Activate security mode

54701

Task: Activate the security mode of the IO-Link interface of the IO-Link master. Set the password "password" (Base64 coded: cGFzc3dvcmQ=)

Solution: The activation sonsists of 2 steps:

1 Activate security mode

Use service setdata with datapoint iotsetup/security/securitymode to activate the security mode.

- Request:

```
{
"code":"request",
"cid":-1,
"adr":"/iotsetup/security/securitymode/setdata",
"data":{"newvalue":"1"}
}
```

- Response:

```
{
"cid":-1,
"code":200
}
```

2 Set required password

Use service setdata with data point iotsetup/security/password to set the required password.

- Request:

```
{
"code":"request",
"cid":-1,
"adr":"/iotsetup/security/password/setdata",
```

```
"data":{"newvalue":"cGFzc3dvcmQ="}
}
```

- Response:

```
{
"cid":-1,
"code":200
}
```

Example: Request with authentication

54685

Task: The temperature of the IO-Link master is to be read. The security function is enabled (current password: password).

Solution: Read the data point processdatamaster/temperature with the getdata service. The request must be sent using https. The user name and the password are transferred as a Base64-coded character string ("administrator" = "YWRtaW5pc3RyYXRvcg==", "password" = "cGFzc3dvcmQ=")

- Request:

```
{
"code":"request",
"cid":-1,
"adr":"processdatamaster/temperature/getdata",
"auth":{"user":"YWRtaW5pc3RyYXRvcg==","passwd":"cGFzc3dvcmQ="}
}
```

- Response:

```
{
"cid":-1,
"data":{"value":37},
"code":200
}
```

Example: reset password

54686

Task: The existing password is to be reset.

Solution: To reset a password, disable the security mode. To disable it, enter the user name and the password (the fields "user" and "passwd").

- Request:

```
{
"code":"request",
"cid":-1,
"adr":"iotsetup/security/securitymode/setdata",
"data":{"newvalue":0},
"auth":{"user":"YWRtaW5pc3RyYXRvcg==","passwd":"SW9UNGlmbQ=="}
}
```

- Response:

```
{
"cid":-1,
"code":200
}
```

9.2.8 IO-Link ports: Setting the operating mode of pin 4 (US)

59793

Substructure: `iolinkmaster/port[n]` (n = 1...4).

Available data points:

Name	Description	Access
../mode	Operating mode of the IO-Link port	rw*
../mastercycletime_preset	Cycle time of the data transfer at the IO-Link port (value in ms)	rw*
../mastercycletime_actual	Current cycle time of the data transfer at the IO-Link port (value in ms)	r
../comspeed	Data transfer rate of the IO-Link port	r

r ... read only
rw ... read and write
* ... only changeable, if the <Feldsbus> plc is not in RUNNING state

9.2.9 IO-Link ports: Configuring device validation and data storage

59792

Substructure: `iolinkmaster/port[n]` (n = 1...4).

Available data points:

Name	Description	Access
../validation_datastorage_mode	Response of the IO-Link port when a new IO-Link device is connected	rw*
../validation_vendorid	IO-Link ID of the manufacturer that is to be validated	rw*
../validation_deviceid	IO-Link ID of the device that is to be validated	rw*
../datastorage	Structure for port data storage	rw
../datastorage/maxsize	Maximum size of the data storage content (in bytes)	r
../datastorage/chunksize	Size of a data segment (in bytes)	r
../datastorage/size	Size of the data storage content (in bytes)	r

r ... read only
rw ... read and write
* ... can only be changed if the IoT core PLC is not in RUNNING state

Applicable services:

Service	Description
../validation_useconnecteddevice	Validate the IO-Link device connected to the IO-Link port*
../datastorage/getblobdata	Reading the content of the data storage area
../datastorage/stream_set	Transfer an individual data segment*
../datastorage/start_stream_set	Start sequential transmission of several data segments*

* ... can only be changed if the IoT core PLC is not in the RUNNING state

Example: Clone the Data Storage of an IO-Link port

52344

Task: Save the Data Storage of IO-Link port X02 of IO-Link master 1 and restore the data at IO-Link master 2.

Solution: The cloning process consists of 2 steps. In the first step, the Data Storage of the IO-Link port of IO-Link master 1 is saved. In the second step, the saved data is restored at the Data Storage of port IO-Link port of IO-Link master 2.

Save Data Storage:

1 Preparations

- ► Read size of segments of Data Storage (h = number of bytes):
  {"code":"request", "cid": -1,"adr":"/iolinkmaster/port[2]/datastorage/chunksize/getdata"}
  Example: h = 256
- ► Read total size of Data Storage area (g = number of bytes):
  {"code": "request", "cid": -1, "adr": "/iolinkmaster/port[2]/datastorage/size/getdata"}
  Example: g = 550
- ► Calcuate the number of reading steps n: n = first integer value to which the following applies: g < n*h
  Example: n= 3, because 550 < 3*256

2 Read Data Storage of IO-Link port

- ► Read Data Storage segment by segment ("pos" is the byte offset, at which the reading process with length "length" starts).
  {"code": "request", "cid": -1, "adr": "/iolinkmaster/port[2]/datastorage/getblobdata", "data": {"pos": 0, "length": h}}
  {"code": "request", "cid": -1, "adr": "/iolinkmaster/port[2]/datastorage/getblobdata", "data": {"pos": h, "length": h}}
  {"code": "request", "cid": -1, "adr": "/iolinkmaster/port[2]/datastorage/getblobdata", "data": {"pos": 2*h, "length": h}}
  ...
  {"code": "request", "cid": -1, "adr": "/iolinkmaster/port[2]/datastorage/getblobdata", "data": {"pos": n*h, "length": h}}
  Example:
  1st read request: pos = 0, length = 256
  2nd read resquest: pos = 256, length =256
  3rd read request: pos = 512, length = 256
- > Each segment value will be returned as BASE64 coded string.
- ► Join segments.

Restore Data Storage:

1 Preparations

- ► Determine the size of the saved Data Storage value (n = number of bytes).
  Example: n = 550
- ► Read size of segments (s = number of bytes):
  {"code":"request", "cid": -1,"adr":"/iolinkmaster/port[1]/datastorage/chunksize/getdata"}
  Example: s = 256

2 Transfer Data Storage strings

- ► Start transfer of Data Storage string ("size" = size of Data Storage string):
  {"code":"request", "cid": -1, "adr":"/iolinkmaster/port[1]/datastorage/start_stream_set", "data": {"size": n}}
  Example: size = 550
- ► Transfer Data Storage string segment by segment ("value" = string value of length s):
  {"code": "request", "cid": -1, "adr": "/iolinkmaster/port[1]/datastorage/stream_set", "data": {"value": "aWZtfgIAAABBTDF4NXhfY25faXRfdDIuMi43Nw..."}

9.2.10 IO-Link ports: Configuring data transfer to LR AGENT or LR SMARTOBSERVER

59795

Substructure: `iolinkmaster/port[n]` (n = 1...4).

Available data points:

Name	Description	Access
../senddatatosmob	Process data to LR AGENT or LR SMARTOBSERVER	rw

rw ... read and write

9.2.11 IO-Link ports: Reading / writing process data

61156

Substructure: `iolinkmaster/port[n]` (n = 1...4)

Available data points:

Name	Description	Access
../pin2in	Value of the digital input on pin 2 of the IO-Link port	r
../iolinkdevice/pdin	Value of the IO-Link input on pin 4 of the IO-Link port	r
../iolinkdevice/pdout	Value of the IO-Link output on pin 4 of the IO-Link port	rw*

r ... read only
rw ... read and write
*... can only be changed if the fieldbus PLC is not in RUNNING state

Example: Read IO-Link process data (operating mode "IO-Link")

33842

Task: Read the current measured value of the ifm temperature sensor TN2531 at IO-Link port X02
Solution: Read the data point for the process input data with the `getdata` service.

- Request:

```
{
"code":"request",
"cid":4711,
"adr":"/iolinkmaster/port[2]/iolinkdevice/pdin/getdata"
}
```

- Response:

```
{
"cid":4711,
"data":{"value": "03C9"},
"code":200
}
```

The return value is given in hexadecimal format. Besides the temperature value the return value comprises additional information (→ IO Device Description (IODD) of the sensor). The temperature value is shown in bits 2 to 15.

0x03C9 = 0b**1111001**001

Temperature value: 0b11110010 = 242

Therefore: The current temperature value is 24.2 °C.

Example: Writing IO-Link value (operating mode "IO-Link")

59804

Task: Switch on the buzzer of DV2500 at IO-Link Port X2. The DV2500 operates in On/Off mode.

Solution: The IODD of the DV2500 shows the structure of the IO-Link process value (→ e.g. LED activity). The buzzer will be switched using bit 40 of the process value (OFF = 0, ON = 1).

To switch the buzzer:

1. Read the current process value (→ **Example: Read IO-Link process data (operating mode "IO-Link")** (→ S. 49)).
2. Set bit 40 of the read value to 1.
3. Write the process value to the IO-Link device.

Example:

Read process value:

0x0000 0000 004D = 0b0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0100 1101

New process value:

0b0000 000**1** 0000 0000 0000 0000 0000 0000 0000 0000 0100 1101 = 0x0100 0000 004D

- Request:

```
{
"code":"request",
"cid":10,
"adr":"iolinkmaster/port[2]/iolinkdevice/pdout/setdata",
"data":{"newvalue":"01000000004D"}
}
```

- Response:

```
{
"cid":10,
"code":200
}
```

Example: Writing digital output (operating mode "DO")

59803

Task: Set the output value of the IO-Link devices at IO-Link Port X1 to "ON". The operating mode of the IO-Link port is "Digital Output (DO)".

Solution: Write the value 1 to data point pdout. The value has to be written as hexadecimal value with a length of 1 byte (OFF = "00", ON = "01").

- Request:

```
{
"code":"request",
"cid":10,
"adr":"iolinkmaster/port[1]/iolinkdevice/pdout/setdata",
"data":{"newvalue":"01"}
}
```

- Response:

```
{
"cid":10,
"code":200
}
```

Example: Reading digital input (operating mode "DI")

59802

Task: Read the current input value of the IO-Link device at IO-Link port X5. The operating mode of the IO-Link port is "Digital Intput (DI)".

Solution: Read the value of data point pdin. The value will be returned as hexadecimal value with a length of 1 byte (OFF = "00", ON = "01"").

- Request:


```
{
"code":"request",
"cid":10,
"adr":"iolinkmaster/port[5]/iolinkdevice/pdin/getdata"
}
```


- Response:


```
{
"cid":10,
"data":{"value":"00"},
"code":200
}
```

9.2.12 IO-Link ports: Indicating port events

59796

Substructure: iolinkmaster/port[n] (n = 1...4).

Available data points:

Name	Description	Access
../portevent	Indication of the following events at IO-Link port n: ▪ plugging IO-Link device ▪ pulling IO-Link device ▪ changing operating mode of IO-Link port	r

r ... read only

Subscribing events: → **Subscribing to notifications** (→ S. 58)

9.2.13 IO-Link devices: Accessing parameters

59800

The ifm IoT Core supports the configuration of the connected IO-Link devices. A parameter is accessed via IO-Link index and subindex (→ IO Device Description (IODD) of the device).

Substructure: `iolinkmaster/port[n]/iolinkdevice` (n = 1...4)

Applicable services:

Service	Description
../iolreadacyclic	Read a parameter of an IO-Link device (acyclic)
../iolwriteacyclic	Write a parameter of an IO-Link device (acyclic)

Example: Read the parameter value of an IO-Link device

33847

Task: Read the serial number of the ifm temperature sensor TN2531 at IO-Link port X02

Solution: Read the serial number with the `iolreadacyclic` service from the IO-Link device (index: 21, subindex: 0)

- **Request:**

```
{
"code":"request",
"cid":4711,
"adr":"/iolinkmaster/port[2]/iolinkdevice/iolreadacyclic",
"data":{"index":21,"subindex":0}
}
```

- **Return:**

```
{
"cid":4711,
"data":{"value":"4730323134323830373130"},
"code":200
}
```

The returned value is given in hexadecimal format. The conversion of the HEX value in a STRING value is: G0214280710

Example: Change the parameter value of an IO-Link device

33844

Task: Set the output configuration OUT1 of the ifm temperature sensor TN2531 at IO-Link port X02 to the value "Hnc / hysteresis function, normally closed".

Solution: Change the parameter [ou1] of the sensor to the value 4 using the iolwriteacyclicdata service. The parameter can be accessed via IO-Link index 580, subindex 0 (→ IO-Link description of the sensor).

- Request:

```
{
"code":"request",
"cid":4711,
"adr":"/iolinkmaster/port[2]/iolinkdevice/iolwriteacyclic",
"data":{"index":580,"subindex":0,"value":"34"}
}
```

The value has to be given in hexadecimal format. The conversion of the STRING value in a HEX value is: 34.

- Response:

```
{
"cid":4711,
"code":200
}
```

9.2.14 IO-Link devices: Reading an writing device information

59797

Substructure: `iolinkmaster/port[n]/iolinkdevice` (n = 1...4)
Available data points:

Name	Description	Access
../status	Status of the connected IO-Link device	r
../vendorid	IO-Link ID of the vendor	r
../deviceid	IO-Link ID of the IO-Link device	r
../productname	Product name of the IO-Link device	r
../serial	Serial number of the IO-Link device	r
../applicationspecifictag	Device-specific identification (application tag)	rw

r ... read only
rw ... read and write

9.2.15 IO-Link devices: Indicating IO-Link events

59798

Substructure: `iolinkmaster/port[n]/iolinkdevice` (n = 1...4).
Available data points:

Name	Description	Access
../iolinkevent	Indication of IO-Link events	r

r ... read only

Subscribing events: → **Subscribing to notifications** (→ S. 58)

9.2.16 Gateway: Resetting, rebooting and localising the device

59790

Substructure: `firmware`
Applicable services:

Name	Description
../factoryreset	Reset IO-Link master to factory settings
../reboot	Reboot IO-Link master
../signal	Trigger the flashing of the status LED

9.2.17 Gateway: Reading device information

52254

Substructure: `deviceinfo`

Available data points:

Name	Description	Access
../productcode	Article number	r
../vendor	Manufacturer	r
../devicefamily	Device family	r
../hwrevision	Hardware revision	r
../serialnumber	Serial number	r
../revision	Firmware version	r
../bootloaderrevision	Bootloader version	r
../extensionrevisions	Firmware and bootloader version	r
../fieldbustype	Fieldbus	r

r ... read only

Additional information about the AL1351 can be read with the service `getidentity` (→ **Servicet: getidentity** (→ S. 96)).

9.2.18 Gateway: Reading status and diagnostic information

61157

Substructure: `processdatamaster`

Available data points:

Name	Description	Access
../temperature	Temperature of the IO-Link master (value in °C)	r
../voltage	Present voltage value of the supply voltage US (value in mV)	r
../current	Present current value of the sensor supply US (value in mA)	r
../supervisionstatus	Status of the device supply US	r

r ... read only

9.2.19 Gateway: Updating the firmware

59789

Substructure: `firmware`

Available data points:

Name	Description	Access
../version	Software version	r
../type	Software type	r
../container	Structure for updating the firmware	w
../container/maxsize	Maximum size of the container structure (in bytes)	r
../container/chunksize	Size of a data segment (in bytes)	r
../container/size	Size of the container content (in bytes)	r

r = only read
w = write only

Applicable services:

Name	Description
../install	Install firmware transferred to the IO-Link master
../container/stream_set	Transfer an individual data segment
../container/start_stream_set	Start sequential transmission of several data segments

Example: Update firmware

52252

Task:

Update the firmware of the device; size of the firmware file: 356676 bytes

Solution:

The firmware is transferred to the device in fragments (chunks). The size of the fragments depends on the size of the flash memory of the IO-Link master. To transfer the firmware, the firmware file must be converted into a character string using BASE64.

1 Preparations

► Determine the size of the fragments (g = number of bytes):
{"code":"request", "cid": -1, "adr":"/firmware/container/chunksize/getdata"}

► Convert the firmware file into a BASE64 string.

2 Start the transfer of the firmware

► Start the transfer of the firmware via the service start_stream_set (parameter "size": size of the firmware file):
{"code":"request", "cid": -1, "adr":"/firmware/container/start_stream_set", "data":{"size":356676}}

3 Load the firmware into the flash memory of the IO-Link master

► Send the BASE64 string of the firmware file to the IO-Link master fragment by fragment (value = string value with length g).
{"code": "request", "cid": -1, "adr": "/firmware/container/stream_set", "cid": -1, "data": {"value": "aWZtfgIAAABBTDF4NXhfY25faXRfdDIuMi43Nw..."}

► Repeat step 3 until all fragments of the firmware file have been sent to the IO-Link master.

> IO-Link master stores the segments received in the container area.

4 Install firmware

► Start the installation of the transmitted firmware.
{"code": "request", "cid": -1, "adr": "/firmware/install", "data": {}}

9.2.20 Gateway: Setting the application tag

59791

Substructure: `devicetag`

Available data points:

Name	Description	Access
../applicationtag	Name of the IO-Link master (application tag)	rw

rw ... read and write

For the storage of the applicationtag 32 bytes are available on the IO-Link master. If the memory area is exceeded during writing with setdata, the IoT core aborts the write process and returns the diagnostics code 400.

When writing the application tag, note the different memory requirements of the individual UTF-8 characters:

- characters 0-127: 1 byte per character
- characters >127: more than 1 byte per character

Example: Change name of the IO-Link master

a33823

Task: Set the name of the IO-Link master to AL1351 for the representation in the LR SMARTOBSERVER.

Solution: Change the parameter [Application Tag] with the setdata service to the value [AL1351].

The data point of the parameter [Application Tag] in the device description object is /devicetag/applicationtag.

- Request:

```
{
"code":"request",
"cid":4711,
"adr":"/devicetag/applicationtag/setdata",
"data":{"newvalue":"AL1351"}
}
```

- Response:

```
{"cid":4711,"code":200}
```

9.2.21 Subscribing to notifications

61159

If a data point has the sub-element `datachanged`, the user can subscribe to notifications on value and condition changes. Notifications can be triggered by the expiration of a timer or an event. The IoT Core supports the output of notifications in CSV or JSON format.

Available data points:

Name	Description	Access
timer[x]/counter	Timer for triggering a notification	rw
timer[x]/interval	Cycle time of the update of the subscribed values	rw
iolinkmaster/port[n]/portevent	Display of the following events on IO-Link port n: ▪ IO-Link device connected ▪ IO-Link device disconnected ▪ Operating mode of the IO-Link port changed	rw
iolinkmaster/port[n]/iolinkdevice/iolinkevent	Display of IO-Link events	rw

r ... read only
rw ... read and write
x = [1,2]
n = 1...4

Applicable services:

Name	Description
../datachanged/subscribe	Subsrscibe to notification
../datachanged/unsubsribe	Unsubscribe notification
../datachanged/getsubscriptioninfo	Show information about notifications

Additionally, the user can use **Service: getsubscriberlist** (→ S. 97) show all active subscriptions.

Example: Subscribing to notifications

61160

Task: The current values of the following parameters are to be sent regularly to a network server with IP address 192.168.0.4:

- cyclic input data of the IO-Link Devices an IO-Link port X02
- Operating temperature of the IO-Link master.

Solution: Subscribe to the required data using the subscribe service.

The following options are additionally available:

- via WebSockets (ws://): **Example: Subscribing notifications via WebSocket** (→ S. 62)
- via MQTT (mqtt://): **Example: Configuring the MQTT command channel** (→ S. 66)

- Request:

```
{
"code":"request",
"cid":4711,
"adr":"/timer[1]/counter/datachanged/subscribe",
"data":
{
```

```
"callback":"http://192.168.0.4:80/temp",
"datatosend":[
"/iolinkmaster/port[2]/iolinkdevice/pdin",
"/processdatamaster/temperature"]
}
}
```

In addition, the time interval of the timer[1] must be set to a value between 500 ms and 2147483647 ms.

- Request:

```
{
"code":"request",
"cid":4712,
"adr":"/timer[1]/interval/setdata",
"data":{"newvalue":500}
}
```

- Response:

```
{
"cid":4712,
"code":200
}
```

- Notification (JSON)

```
{
"code":"event",
"cid":4711,
"adr":"",
"data":{
"eventno":"6317",
"srcurl":"/timer[1]/counter/datachanged",
"payload":{
"/timer[1]/counter":{"code":200,"data":1},
"/processdatamaster/temperature":{"code":200,"data":39},
"/iolinkmaster/port[2]/iolinkdevice/pdin":{"code":200,"data":"03B0"}}}
}
```

Example: Changing a subscription

61161

Task: The existing subscription (**Example: Subscribing to notifications** (→ S. 58)) is to be changed. Instead of the temperature of the IO-Link master, the operating voltage applied is to be transmitted.

Solution: Overwrite the existing subscription. For this purpose, the parameter values for "cid" and "callback" in the request must be the same as those of the existing subscription.

- Request:

```
{
"code":"request",
"cid":4711,
"adr":"/timer[1]/counter/datachanged/subscribe",
"data":{
"callback":"http://192.168.0.4:80/temp",
"datatosend":[
"/iolinkmaster/port[2]/iolinkdevice/pdin",
"/processdatamaster/voltage"]}
}
```

Example: Subscribing to notifications in CSV format

61162

Task: Every 2 seconds, the current values of the following parameters are to be sent to a network server with the IP address 192.168.0.4

- cyclic IO-Link input data of the IO-Link device at port X02
- Operating temperature of the IO-Link master.

The data should be transmitted in CSV format (comma separator).

Solution:

► Use the subscribe service to subscribe to the required data and set the output format to "csv0".



Data in CSV format can only be sent via TCP protocol.

- Request:

```
{
"cid": 1,
"adr":"/timer[1]/counter/datachanged/subscribe",
"code":"request",
"callback":"tcp://192.168.50.59:1883/topic",
"codec":"csv0",
"data":{
"datatosend":[
"/iolinkmaster/port[2]/iolinkdevice/pdin",
"/processdatamaster/temperature"]}
}
```

► Set the interval of the timer to 2 seconds:

- Request:

```
{
"code":"request",
"cid":4712,
"adr":"/timer[1]/interval/setdata",
"data":{"newvalue":2000}
}
```

The cyclically sent notification has the following structure:

```
/timer[1]/counter/datachanged,6317,200,1,200,39,200,03B0
```

Example: Unsubscribing from notifications

61163

Task: The existing subscription (**Example: Subscribing to notifications** (→ S. 58)) is to be deleted.

Solution: Use the unsubscribe service to delete the subscription. For this purpose, the value of the parameter "callback" in the request must be equal to the value of the existing subscription.

```
{
"code":"request",
"cid":4711,
"adr":"/timer[1]/counter/datachanged/unsubscribe",
"data":{
"callback":"http://192.168.0.4:80/temp"
}
```

Example: Checking subscriptions

61164

Task: Information about the existing subscription (**Example: Subscribing to notifications** (→ S. 58)Show **Example: Subscribing to notifications** (→ S. 58)).

Solution: Use the service getsubscriptioninfo and the parameter values cid, "adr" and "callback" of the existing subscription to retrieve the information.

- Request:

```
{
"code":"request",
"cid":4711,
"adr":"/timer[1]/counter/datachanged/getsubscriptioninfo",
"data":{
"callback":"http://192.168.0.4:80/temp"}
}
```

- Response:

```
{
"cid." 4711,
"data":{
"callback":"http://192.168.0.4:80/temp",
"datatosend":[
"/iolinkmaster/port[2]/iolinkdevice/productname",
"/iolinkmaster/port[2]/iolinkdevice/pdin",
"/processdatamaster/temperature"]},
"code":200
}
```

9.2.22 Using Web Socket

61165

The IoT Core supports communication via WebSocket protocol. With Web Sockets, the user can establish a full-duplex communication channel via a TCP connection.

WebSockets can be used for the following services:

- subscribe / unsubscribe

Maximum number of WebSocket connections: 8
Fail-safe WebSocket connections (wss://) are not supported.

To transmit notifications via a WebSockets connection:

► Establish the WebSocket connection (e.g. "`ws://192.168.0.55:80/websocket`")

- Option 1: without parameter "callback"

► make subscribe/unsubscribe request without parameter "callback".

> IoT-Core sends notifications about existing WebSocket connections.

- Option 2. with parameter "callback"

► make subscribe/unsubscribe requests with parameter "callback" ("`ws:///myTopic`").

> IoT-Core sends notifications about existing WebSocket connections to the topic `myTopic`.

Example: Subscribing notifications via WebSocket

61166

Task: The current values of the following parameters are to be sent regularly to the data sink `myTopic` via an existing WebSocket connection:

- Product name of the IO-Link Devices an IO-Link port X02
- cyclic input data of the IO-Link Devices an IO-Link port X02
- Operating temperature of the IO-Link master.

Solution: Subscribe to the required data using the subscribe service.

- Request:

```
{
"code":"request",
"cid":4711,
"adr":"/timer[1]/counter/datachanged/subscribe",
"data":{
"callback":"ws:///myTopic",
"datatosend":[
"/iolinkmaster/port[2]/iolinkdevice/productname",
"/iolinkmaster/port[2]/iolinkdevice/pdin",
"/processdatamaster/temperature"]}
}
```

If the notifications are to be transmitted via the existing WebSocket connection, but without a special data sink, the callback parameter is not required.

- Request:

```
{
"code":"request",
"cid":4711,
"adr":"/timer[1]/counter/datachanged/subscribe",
"data":{
"datatosend":[
```

```
"/iolinkmaster/port[2]/iolinkdevice/productname",
"/iolinkmaster/port[2]/iolinkdevice/pdin",
"/processdatamaster/temperature"]}
}
```

9.2.23 MQTT support

61168

The IoT Core supports the MQTT protocol. The protocol allows an MQTT client to communicate with the IoT Core via an MQTT broker to request and receive data. The IoT Core can publish data via the MQTT connection.

Configuring the MQTT command channel

61169

To enable MQTT communication, the user needs to activate and configure an MQTT command channel.

Substructure: `connections/mqttConnection`

Name	Description	Access
../type	Type of the connection (MQTT)	r
../status	Global MQTT status	r
../status/preset	Presetting of the MQTT status; Basic settings: running	r
../MQTTSetup	Substructure for general MQTT settings	w
../MQTTSetup/QoS	Quality of Service of the MQTT communication ▪ 0: QoS Level 0 - PUBLISH (without confirmation) ▪ 1: QoS Level 1 - PUBLISH > PUBREC (one-time confirmation) ▪ 2: QoS Level 2 - PUBLISH > PUBREC > PUBREL > PUBCOMP (double confirmation)	rw
../MQTTSetup/version	MQTT version	r
../mqttCmdChannel	Substructure of the MQTT command channel	w
../mqttCmdChannel/type	Type of the MQTT command channel	r
../mqttCmdChannel/status	Status of the MQTT command channel	r
../mqttCmdChannel/status/preset	Presetting of the MQTT status; Basic setting: stopped	r
../mqttCmdChannel/mqttCmdChannelSetup	Structure for settings of the command channel	w
../mqttCmdChannel/mqttCmdChannelSetup/brokerIP	IP address of the MQTT broker	rw
../mqttCmdChannel/mqttCmdChannelSetup/brokerPort	Port number of the MQTT broker	rw
../mqttCmdChannel/mqttCmdChannelSetup/cmdTopic	Designation of the MQTT topic	rw
../mqttCmdChannel/mqttCmdChannelSetup/defaultReplyTopic	Standard response topic	rw

Applicable services:

Name	Description
../status/start	Enable MQTT
../status/stop	Deactivate MQTT
../status/reset	Reset MQTT
../mqttCmdChannel/status/start	Activate MQTT command channel
../mqttCmdChannel/status/stop	Deactivate MQTT command channel
../mqttCmdChannel/status/reset	Reset MQTT command channel

Notes on the states of an MQTT connection: **Note: Connection states** (→ S. 65)

To create an MQTT connection, perform the following steps in sequence:

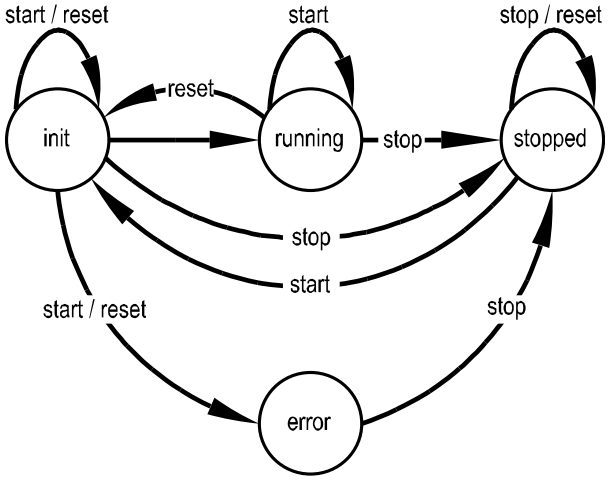
Ensure that the MQTT broker can be reached and that the selected port of the MQTT broker is enabled for data transmission.
Max. number of simultaneous MQTT connections: 10
Wildcards "+" and "#" in topics are not supported.

- ► Activate MQTT command channel.
- ► Set the IP address of the MQTT.
- ► Set the port number of the MQTT broker.
- ► Set topic.
- ► Set standard response topic.
- > The command channel is created with the selected properties.
- > The user can publish on the topic with the IoT Core.
- > MQTT clients can subscribe to the topic.

Note: Connection states

61170

The following status diagram shows the influence of the services "start", "stop" and "reset" on the status of an MQTT connection:



After the initialisation in the "init" state has been completed, the connection automatically changes to the "running" state.

The connection automatically switches to the "error" state if at least one of the following events occurs:

- no MQTT broker available

Example: Configuring the MQTT command channel

61171

Task: Configuring and activating the MQTT command channel (IP address MQTT broker: 192.168.82.100, port: 1883, topic: abc).

Solution:

► Check whether MQTT broker can be reached and the port has been released.

► Activate command channel

- Request:

```
{
"code":"request",
"cid":4711,
"adr":"/connections/mqttConnection/MQTTSetup/mqttCmdChannel/status/start"
}
```

► Set the IP address of the MQTT broker/server.

- Request:

```
{
"code":"request",
"cid":4712,
"adr":"/connections/mqttConnection/mqttCmdChannel/mqttCmdChannelSetup/brokerIP/set
data"
"data":{"192.168.82.100"}
}
```

► Set the port number of the MQTT broker/server.

- Request:

```
{
"code":"request",
"cid":4713,
"adr":"/connections/mqttConnection/mqttCmdChannel/mqttCmdChannelSetup/brokerPort/s
etdata"
"data":{"1883"}
}
```

► Set topic.

- Request:

```
{
"code":"request",
"cid":4714,
"adr":"/connections/mqttConnection/mqttCmdChannel/mqttCmdChannelSetup/cmdTopic/set
data"
"data":{"abc"}
}
```

► Set standard response topic.

- Request:

```
{
"code":"request",
"cid":4715,
"adr":"/connections/mqttConnection/mqttCmdChannel/mqttCmdChannelSetup/defaultReply
Topic/setdata"
"data":{"xyz"}
}
```

► Set the QoS.

- Request:

```
{
"code":"request",
"cid":4716,
"adr":"/connections/mqttConnection/MQTTSetup/QoS/setdata",
"data":{"QoS2"}
}
```

Example: Publish the temperature to an MQTT broker

54687

Task: Publish the temperature of the IO-Link master to an MQTT broker (IP address MQTT broker: 192.168.82.100, port: 1883, topic: abc

Solution:

- Request:

```
{
"code":"request",
"cid":-1,
"adr":"/timer[1]/counter/datachanged/subscribe",
"data":{
"callback":"mqtt://192.168.82.100:1883/abc",
"datatosend":["processdatamaster/temperature"}
}
```

- Response:

```
{
"cid":-1,
"code":200
}
```

9.2.24 Using the IoT-Core Visualizer

Content

Managing notifications 69
Searching for elements in the device tree 71
Configuring IO-Link the master 72
Reading and writing process data 73
Updating the firmware 74

61173

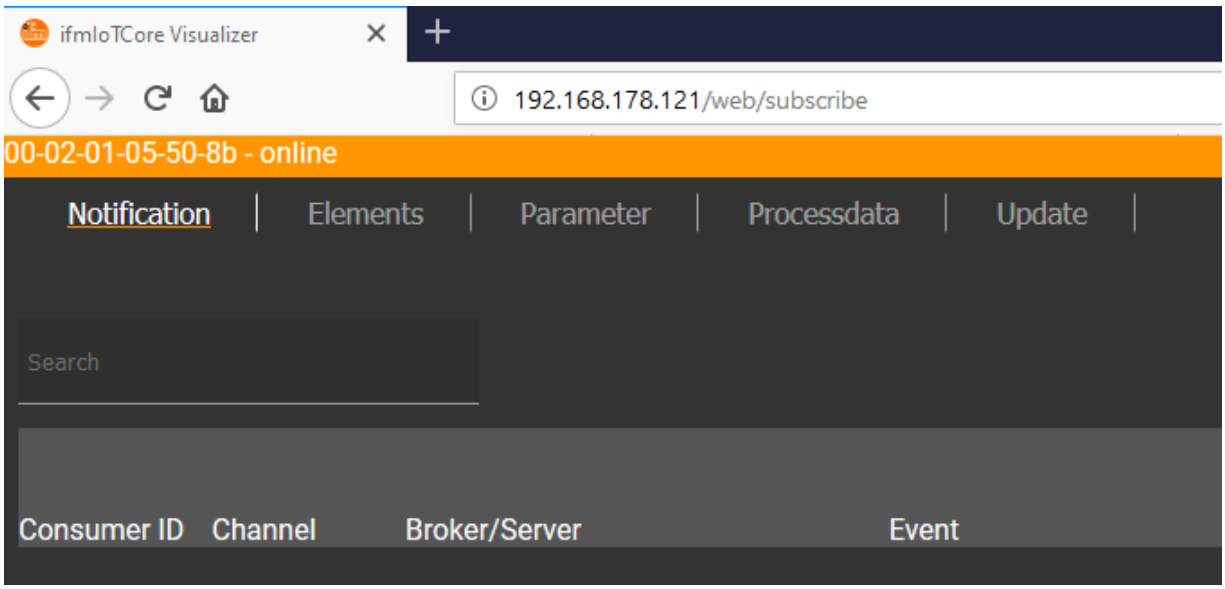
The ifm-IoT Core Visualizer of the IO-Link master provides a graphical user interface for accessing functions of the ifm-IoT Core.

To start the IoT Core Visualizer:

► Start web browser.

► Call the following address: http://ipaddress/web/subscribe

> Browser shows IoT Core Visualizer:



The navigation menu gives the user access to the following functions:

- [Notification]: Creating and managing notifications (subscribe / unsubscribe)
- [Elements]: Searching for elements in device description
- [Parameter]: Configuring IO-Link master
- [Processdata]: Reading and writing process data
- [Update]: Updating the firmware of the IO-Link master

Managing notifications

61174

The menu page allows you to perform the following functions

- Creating notifications
- Showing active notifications
- Deleting notifications (single, all)

Requirements:

- Iot-Core Visualizer has been started.

► Click on [Notification].

> The menu page for managing notifications appears.

> The menu page shows all registered notifications in a table

Creating a new notification

61175

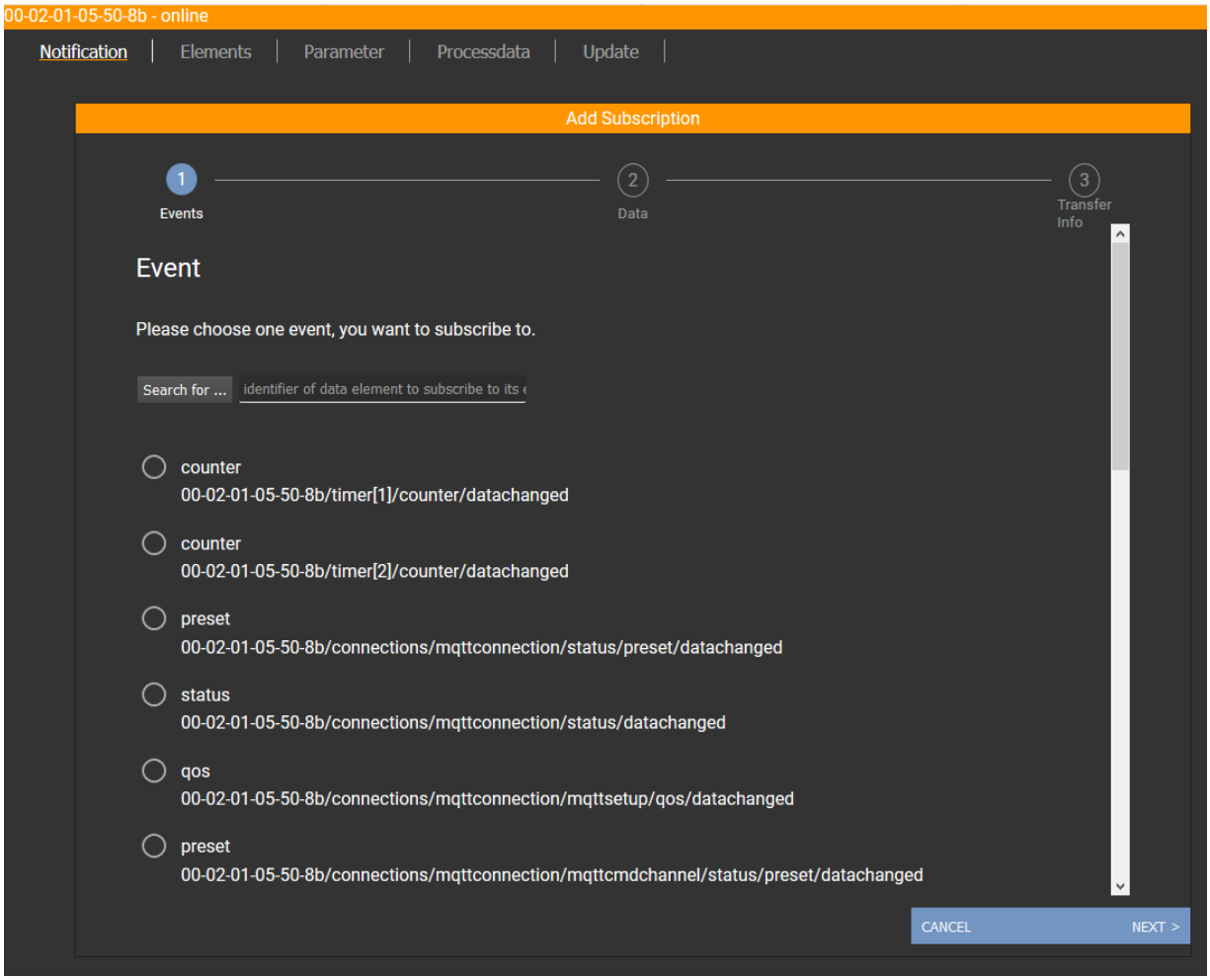
A wizard is used to register new notifications.

Requirements:

- The [Notification] menu page is open.

► Click on [+] on the right side of the table.

> The wizard for the creation of notifications appears.



► Use the wizard to enter the required notification parameters step by step.
> Created notification subscription is displayed in the table.

For cyclical notifications via timer[1] or timer[2], the user also needs to set the interval time of the timer in question.

Deleting a notification

61176

Requirements:

- The [Notification] menu page is open.
- At least one notification is active.

► Click on [x] in the column [Unsubscribe].
> The selected notification will be deleted (unsubscribe).

Searching for elements in the device tree

61177

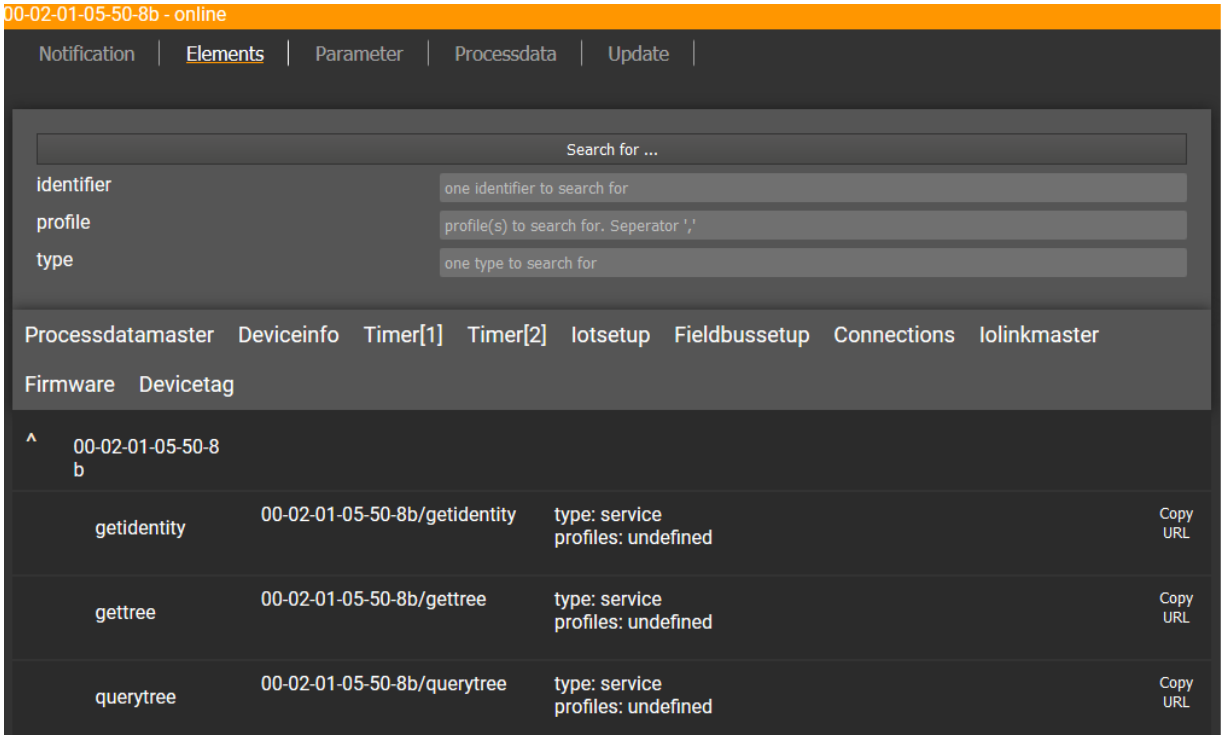
The [Elements] menu page allows you to search the device description for elements with specific properties (status, profile, name) and to output the results.

Requirements:

- Iot-Core Visualizer has been started.

► Click on [Elements].

> The input mask appears.



► Enter the search criteria of the required item in the [identifier], [profile] and [type] boxes.

► Click on [Search for ...].

> IoT-Core Visualizer searches device description for elements with selected search criteria.

> The result list shows all elements found.

Configuring IO-Link the master

61178

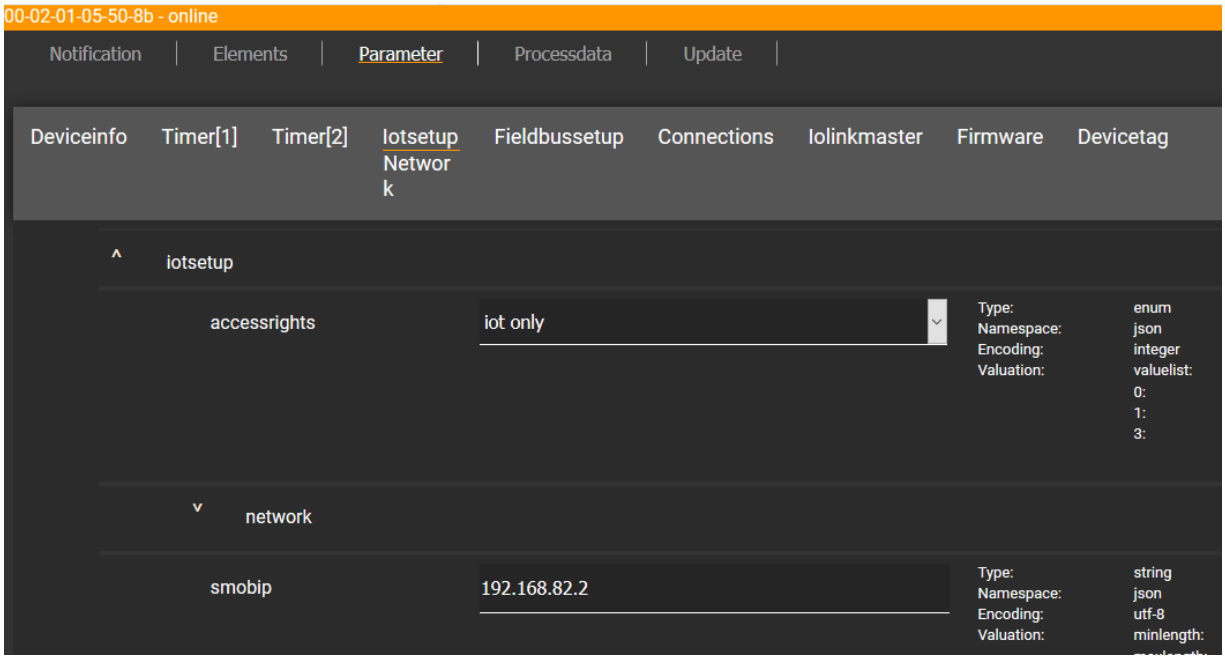
The [Parameter] menu page allows you to configure the IO-Link master.
Available options:

- Reading and writing individual parameters
- Backup and restore the current configuration of the machine.

Requirements:

- Iot-Core Visualizer has been started.

► Click on [Parameter].
> The menu page shows the available parameters of the IO-Link master.
> Current parameter values are displayed.
> Editable parameters can be changed.



To change a parameter:

► Navigate to the desired parameter in the device description.
► Changing the parameter value
► Click on the pencil icon to save the change on the IO-Link master.
> The changed parameter value is active.
► Optional: Repeat the procedure to change further parameter values.

Reading and writing process data

61179

The menu page allows the process data of the IO-Link master and the connected IO-Link devices to be read and written.

Requirements:

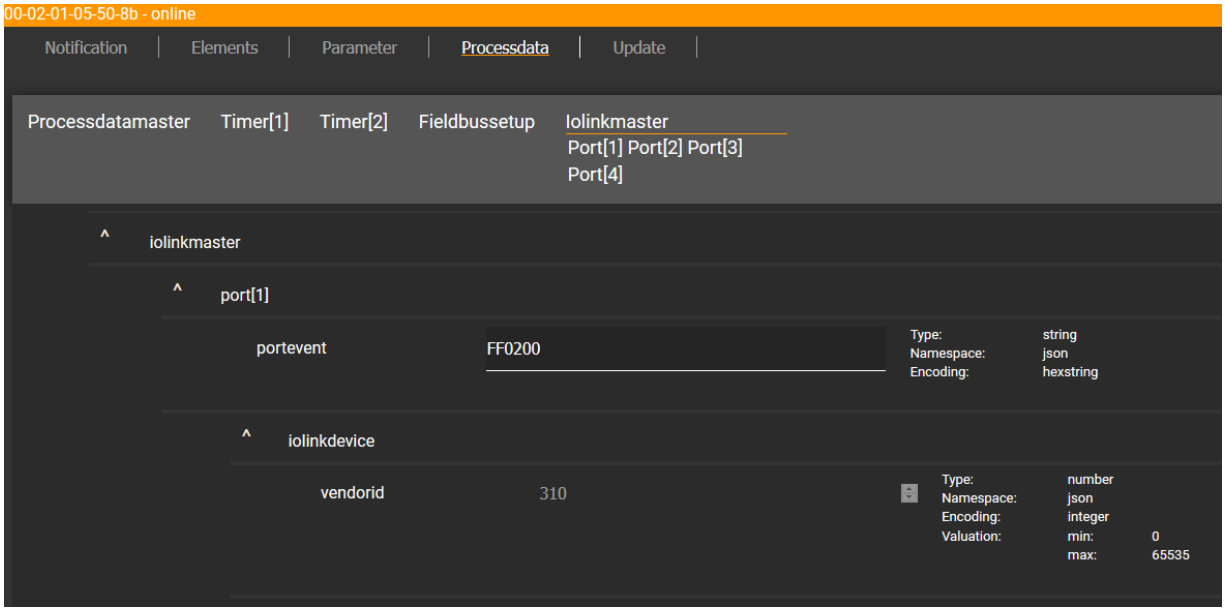
- Iot-Core Visualizer has been started.

► Click on [Processdata].

> Menu page shows the substructures of the device description that contain process data and events.

> The current process values are displayed.

> Editable process data can be changed.



To change the value of a process date:

► Navigate to the required process date in the device description.

► Change the process value.

► Click on the pencil icon to save the change on the IO-Link master.

> The changed process value is active.

► Optional: Repeat the procedure to change further process values.

Updating the firmware

61180

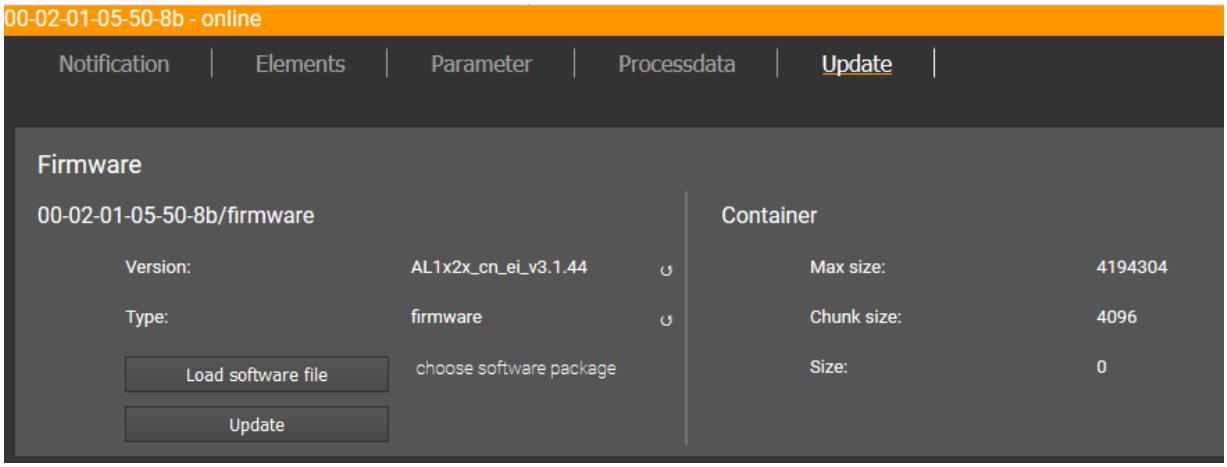
The [Update] menu page allows you to update the firmware of the IO-Link master:

Requirements:

- Iot-Core Visualizer has been started.

► Click on [Update].

> Menu page displays information about the current firmware version.



► Click on [Load software file] and select a new firmware file (*.bin).

► Click on [Update] to start the update process.

> The firmware of the IO-Link master will be updated.

> The area shows the progress bar.

> If the update process has been successful, the IO-Link master will restart automatically.

10 Operation

Content

Using web-based management.....75

34061

10.1 Using web-based management

61181

The device has an integrated web server The web server generates a website with the following data:

- Status information of the ports
- Access to product page of connected IO-Link devices (only ifm devices)
- Diagnostic information of the device
- Version information of the installed firmware components

To access the web interface of the IO-Link master:

► Connect the IO-Link master to the laptop / PC via the IoT port.

► Optional: Check the IP settings of the IoT interface.

► Start web browser.

► In the address field of the web browser, enter the IP address of the IoT interface and confirm with [ENTER].

> The web browser shows the website with the status and diagnostic information of the device.

11 Maintenance, repair and disposal

Content

Cleaning process....76
Updating the firmware....76
Replacing IO-Link device....76

51990

The operation of the unit is maintenance-free.

► Dispose of the unit in an environmentally friendly way in accordance with the applicable national regulations when it is no longer used.

11.1 Cleaning process

51991

► Clean the surface of the unit when necessary.
► Do not use any caustic cleaning agents for this!
► In case of severe soiling, use a damp cloth.
► Do not use any caustic cleaning agents for this!

11.2 Updating the firmware

61183

The firmware of the IO-Link master can be updated via the IoT Core Visualizer↘→ **Updating the firmware** (→ S. 74).

11.3 Replacing IO-Link device

34182

To replace an IO-Link device:

Requirement:

> New IO-Link device is with factory settings.
> New IO-Link device supports IO-Link standard 1.1 or higher.

1 Set data storage

► Set the following parameters of the IO-Link port
▪ Set Validation and Data Storage to [Type compatible V1.1 device with Restore] or [Type compatible V1.1. device with Backup + Restore]
▪ Set correct values to [Vendor ID] and [Device ID] according to properties of the IO-Link device.
► Save changes.

2 Replace IO-Link device

► Disconnect old IO-Link device from IO-Link master.
► Connect new IO-Link device with the same IO-Link port of the AL1351.
> IO-Link master copies parameter values from the data memory to the new IO-Link device.

12 Factory settings

33849

In the factory settings, the device has the following parameter settings:

Parameter	Factory setting
[IP address] (IoT interface)	169.254.X.X
[Subnet mask] (IoT interface)	255.255.0.0
[IP gateway address] (IoT interface)	0.0.0.0
[Host name]	blank
Data storage	empty

13 Accessories

33870

List of accessories of AL1351: → www.ifm.com > Product page > Accessories

14 Appendix

Content

Technical data 80
ifm IoT Core 84

33879

14.1 Technical data

Content

Application 80
Electrical data 80
Inputs / outputs 80
Inputs 81
Outputs 81
Interfaces 81
Operating conditions 82
Approvals / tests 82
Mechanical data 82
Electrical connection 83

34188

14.1.1 Application

33878

Application	
Application	Hygienic systems; I/O modules for field applications
Daisy-chain function	Fieldbus interface

14.1.2 Electrical data

33808

Electrical data	
Operating voltage [V]	20...28 DC; (US; to SELV/PELV)
Current Consumption [mA]	300...3900; (US)
Protection class	III
Sensor supply US	
Max. current load total [A]	3.6

14.1.3 Inputs / outputs

34068

Inputs / outputs	
Total number of inputs and outputs	8; (configurable)
Number of Inputs and Outputs	Number of digital inputs: 8; Number of digital outputs: 4

14.1.4 Inputs

34069

Inputs	
Number of digital inputs	8; (IO-Link Port Class A: 4 x 2)
Switching level high [V]	11...28
Switching level low [V]	0...5
Digital inputs protected against short circuits	yes

14.1.5 Outputs

34053

Outputs	
Number of digital outputs	4; (IO-Link Port Class A: 4 x 1)
Max. current load per output [mA]	300
Short-circuit protection	yes

14.1.6 Interfaces

52260

Interfaces	
Communication interface	Ethernet; IO-Link
Communication interface	IO-Link; TCP/IP; TCP/IP JSON
Ethernet	
Transmission standard	10Base-T; 100Base-TX
Transmission rate [MBit/s]	10; 100
Protocol	DCP, DHCP, Auto IP
Factory settings	▪ IP address: 169.254.X.X ▪ Subnet mask: 255.255.0.0 ▪ Gateway IP address: 0.0.0.0 ▪ MAC address: see type label
IO-Link master	
Type of transmission	COM 1 / COM 2 / COM 3
IO-Link revision	V1.1
Number of ports class A	4

14.1.7 Operating conditions

34062

Operating conditions	
Applications	Indoor use
Ambient temperature [°C]	-25...60
Storage temperature [°C]	-25...85
Max. perm. relative air humidity [%]	90
Max. height above sea level [m]	2000
Protection rating	IP 65; IP 66; IP 67; IP 69K; (operation with stainless steel protective caps: IP 69K)
Pollution Degree	2

14.1.8 Approvals / tests

33877

Approval / tests	
EMC	▪ EN 61000-6-2 ▪ EN 61000-6-4
MTTF [Years]	90

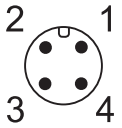
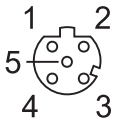
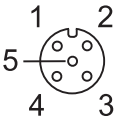
14.1.9 Mechanical data

34050

Mechanical data	
Weight [g]	304,5
Materials	Housing: PA grey; socket: 1.4404 (stainless steel / 316L)

14.1.10 Electrical connection

52261

Voltage supply IN X31		
Plug and socket connection	M12	
Wiring		1: + 24 V DC (US) 2: - 3: GND (US) 4: -
IoT port X21, X22		
Plug and socket connection	M12	
Wiring		1: TX + 2: RX + 3: TX - 4: RX - 5: -
Process connection IO-Link ports class A X01...X04		
Plug and socket connection	M12	
Wiring		1: Sensor supply (US) L+ 2: DI 3: Sensor supply (US) L- 4: C/Q IO-Link 5: -

14.2 ifm IoT Core

Content

Overview: IoT profile....85
Overview: IoT types....92
Overview: IoT services....93

33803

14.2.1 Overview: IoT profile

Content

Profile: blob 85
Profile: deviceinfo 86
Profile: devicetag 86
Profile: iolinkdevice_full 87
Profile: iolinkmaster 87
Profile: mqttCmdChannel 88
Profile: mqttCmdChannelSetup 88
Profile: mqttConnection 88
Profile: mqttSetup 89
Profile: network 89
Profile: parameter 90
Profile: processdata 90
Profile: runcontrol 90
Profile: service 90
Profile: software 90
Profile: software/uploadedablesoftware 91
Profile: Timer 91

34054

Profile: blob

52264

Element (identifier)	Properties	Mandatory	Comment
blobname	▪ type = data ▪ profiles = blob		labels element as device information
../size	type = data	mandatory	
../chunksize	type = data	mandatory	
../setblobdata	type = service	optional	
../getblobdata	type = service	optional	
../start_stream_set	type = service	optional	
../stream_set	type = service	optional	
../clear	type = service	optional	
../getcrc	type = service	optional	
../getmd5	type = service	optional	
../getdata	type = service	optional	
../setdata	type = service	optional	

Profile: deviceinfo

34207

Element (identifier)	Properties	mandatory	Comments
deviceinfo	▪ type = structure ▪ profile = deviceinfo		characterises the element as device information
../devicename	type = data	optional	
../devicefamiliy	type = data	optional	
../devicevariant	type = data	optional	
../devicesymbol	type = data	optional	
../deviceicon	type = data	optional	
../serialnumber	type = data	mandatory	
../productid	type = data	optional	
../productname	type = data	optional	
../productcode	type = data	mandatory	
../producttext	type = data	optional	
../ordernumber	type = data	optional	
../productiondate	type = data	optional	
../productioncode	type = data	optional	
../hwrevision	type = data	mandatory	
../swrevision	type = data	mandatory	
../bootloaderrevision	type = data	optional	
../vendor	type = data	optional	
../vendortext	type = data	optional	
../vendorurl	type = data	optional	
../vendorlogo	type = data	optional	
../productwebsite	type = data	optional	
../supportcontact	type = data	optional	
../icon	type = data	optional	
../image	type = data	optional	
../standards	type = data	optional	

Profile: devicetag

34206

Element (identifier)	Properties	mandatory	Comments
devicetag	▪ type = structure ▪ profile = devicetag		
../applicationtag	type = data	mandatory	
../applicationgroup	type = data	optional	
../machinecode	type = data	optional	
../tenant	type = data	optional	

Profile: iolinkdevice_full

52265

Element (identifier)	Characteristics	Mandatory	Comments
iolinkdevice	▪ type = structure ▪ profile = iolinkdevice_full		Structure of an IO-Link device
../vendorid	type = data	mandatory	
../deviceid	type = data	mandatory	
../productname	type = data	mandatory	
../serial	type = data	mandatory	
../applicationspecifictag	type = data	mandatory	
../pdin	type = data	mandatory	
../pdout	type = data	mandatory	
../status	type = data	mandatory	
../iolreadacyclic	type = data	mandatory	
../iolwriteacyclic	type = data	mandatory	
../iolinkevent	type = data	mandatory	

Profile: iolinkmaster

34205

Element (identifier)	Properties	Mandatory	Comments
masterport	▪ type = structure ▪ profile = iolinkmaster		Executable service
../mode	▪ type = data ▪ profile = parameter	mandatory	
../comspeed	▪ type = data ▪ profile = parameter	mandatory	
../mastercycletime_actual	▪ type = data ▪ profile = parameter	mandatory	
../mastercycletime_preset	▪ type = data ▪ profile = parameter	mandatory	
../validation_datastorage_mode	▪ type = data ▪ profile = parameter	mandatory	
../validation_vendorid	▪ type = data ▪ profile = parameter	mandatory	
../validation_deviceid	▪ type = data ▪ profile = parameter	mandatory	
../additionalpins_in	▪ type = data ▪ profile = processdata	optional	
../additionalpins_out	▪ type = data ▪ profile = processdata	optional	
../portevent	▪ type = data	mandatory	
../iolinkdevice	▪ type = structure ▪ profile = iolinkdevice_full	mandatory	

Profile: mqttCmdChannel

61186

Element (identifier)	Properties	Mandatory	Comment
mqttCmdChannel	▪ type = structure ▪ profile = commChannel		Profile of the MQTT command channel
../type	▪ type = data ▪ data type = STRING	mandatory	Protocol type of the interface
../status	▪ type = data ▪ data type = STRING	mandatory	Status of the MQTT command channel (possible values: init, running, stopped, error)
../mqttCmdChannelSetup	type = profile		Sub-profile: **Profile: mqttCmdChannelSetup** (→ S. 88)

Profile: mqttCmdChannelSetup

61187

Element (identifier)	Properties	Mandatory	Comment
mqttCmdChannelSetup	▪ type = structure ▪ profile = mqttCmdChannelSetup		Settings of the MQTT command channel
../brokerIP	▪ type = datat ▪ data type = STRING	optional	
../brokerPort	▪ type = data ▪ data type = STRING	optional	
../cmdTopic	▪ type = data ▪ data type = STRING	optional	
../defaultReplyTopic	▪ type = data ▪ data type = STRING	optional	

Profile: mqttConnection

61188

Element (identifier)	Properties	Mandatory	Comment
mqttConnection	▪ type = structure ▪ profile = commInterface		MQTT connection in the IoT Core
../type	▪ type = data ▪ data type = STRING	mandatory	Protocol type of the interface
../status	▪ type = data ▪ data type = STRING	mandatory	global status of the MQTT (possible values: init, running, stopped, error)
../mqttSetup	type = profile		Sub-profile: **Profile: mqttSetup** (→ S. 89)
../mqttCmdChannel	type = profile		Sub-profile: **Profile: mqttCmdChannel** (→ S. 88)

Profile: mqttSetup

61189

Element (identifier)	Properties	Mandatory	Comment
mqttSetup	▪ type = structure ▪ profile = mqttSetup		Settings of the MQTT command channel
../QoS	▪ type = data ▪ data type = Number	mandatory	Quality of Service of the MQTT connection
../version	▪ type = data ▪ data type = STRING	mandatory	

Profile: network

52266

Element (identifier)	Characteristics	Mandatory	Comments
network	▪ type = structure ▪ profiles = deviceinfo		Characterises the element as device information
../macaddress	▪ type = data ▪ profile = parameter	mandatory	
../ipaddress	▪ type = data ▪ profile = parameter	optional	
../ipv6address	▪ type = data ▪ profile = parameter	mandatory	
../subnetmask	▪ type = data ▪ profile = parameter	mandatory	
../ipdefaultgateway	▪ type = data ▪ profile = parameter	mandatory	
../dhcp	▪ type = data ▪ profile = parameter	optional	
../ipversion	▪ type = data ▪ profile = parameter	optional	
../hostname	▪ type = data ▪ profile = parameter	optional	
../autonegotiation	▪ type = data ▪ profile = parameter	optional	
../portspeed	▪ type = data ▪ profile = parameter	optional	
../enablenetwork	type = service	optional	
../disablenetwork	type = service	optional	

Profile: parameter

34215

The profile is used to mark the elements of type `data` as parameters (acyclic data). The profile defines no substructure.

Profile: processdata

34225

The profile is used to mark the elements of type `data` as process data (cyclic data). The profile does not define a substructure.

Profile: runcontrol

61190

Element (identifier)	Properties	Mandatory	Comment
runcontrol	▪ type = profile ▪ profile = runcontrol		Control of the MQTT command channel
../start	type = service	mandatory	**Service: start** (→ S. 103)
../stop	type = service	mandatory	**Service: stop** (→ S. 104)
../reset	type = service	mandatory	**Service: Reset** (→ S. 101)

Profile: service

34224

Element (identifier)	Properties	mandatory	Comments
service	▪ type = service ▪ profile = service		Executable service

Profile: software

34223

Element (identifier)	Properties	mandatory	Comments
software	▪ type = structure ▪ profile = software		characterises the element as software
../version	type = data	mandatory	
../type	type = data	mandatory	
../status	type = structure	optional	
../diag	type = structure	optional	

Profile: software/uploadedablesoftware

52267

Element (identifier)	Characteristics	Mandatory	Comments
software	▪ type = structure ▪ profiles = software/uploadablesoftware		Software that can be loaded to the device via the IoT Core
../lastinstall	type = data	optional	
../installhistory	type = data	optional	
../container	▪ type = data ▪ profile = blob	mandatory	
../preinstall	type = service	optional	
../install	type = service	mandatory	
../postinstall	type = service	optional	
../abortinstall	type = service	optional	
../installstatus	type = data	optional	

Profile: Timer

34226

Element (identifier)	Properties	Mandatory	Comment
timer	▪ type = structure ▪ profile = timer		
../counter	▪ type = data ▪ profile = parameter	mandatory	
../interval	▪ type = data ▪ profile = parameter	optional	
../start	type = service	optional	
../stop	type = service	optional	

14.2.2 Overview: IoT types

34055

The ifm IoT Core uses the following element types:

Name	Description
structure	Element is a structure element (like a folder in a file system)
service	Element is a service that can be addressed from the network
event	Element is an event that can be started by the firmware and sends messages.
data	Element is a data point
device	Root element a device represents

14.2.3 Overview: IoT services

Content

Service: factoryreset....93
Service: getblobdata....94
Service: getdata....94
Service: getdatamulti....95
Service: getelementinfo....95
Servicet: getidentity....96
Service: getsubscriberlist....97
Service: getsubscriptioninfo....98
Service: gettree....99
Service: install....100
Service: iolreadacyclic....100
Service: iolwriteacyclic....100
Service: querytree....101
Service: reboot....101
Service: Reset....101
Service: setblock....102
Service: setdata....102
Service: signal....103
Service: start....103
Service: start_stream_set....104
Service: stop....104
Service: stream_set....104
Service: subscribe....105
Service: unsubscribe....106
Service: validation_useconnecteddevice....106

34056

Service: factoryreset

34184

Name: factoryreset

Description: The service sets the parameters of the device to the factory settings.

Request data (field "data"): none

Response data (field "data"): none

Example:

```
{
"code": "request",
"cid": 4711,
"adr": "/firmware/factoryreset"
}
```

Service: getblobdata

52345

Name: getblobdata
Description: The service reads a binary large object (blob).
Applicable to: datastorage
Request data (field "data"):

Data field	Required field	Data type	Default	Description
pos	mandatory	number	0	Byte position
length	mandatory	number	-	Size of the object (number of bytes)

Return data (field "data"):

Data field	Required field	Data type	Default	Description
data	mandatory	STRING	0	Data to be decoded (BASE64 coded)
crc	optional	HEX STRING		CRC of the data after decoding
md5	optional	HEX STRING		MD5 checksum of the data after decoding

Service: getdata

34183

Name: getdata
Description: Service reads the value of a data point and provides it.
Request data (field "data"): none
Return data (field "data"):

Data field	Required field	Data type	Description
value	mandatory	STRING	Value of the element/data point

Example:

```
{
"code":"request",
"cid":4711,
"adr":"devicetag/applicationtag/getdata"
}
```

Service: getdatamulti

34174

Name: getdatamulti

Description: The service sequentially reads the values of several data points and provides them. The value and the diagnostic code are provided for each data point.

Request data (field "data"):

Data field	Required field	Data type	Description
datatosend	mandatory	ARRAY OF STRINGS	List of data points to be requested; data points must support the service getdata ("datatosend":["url1","url2",...,"urlx"])

Response data (field "data"): for each requested data point

Data field	Required field	Data type	Description
url	mandatory	STRING	Data point request
code	mandatory	INT	Diagnostic code of the request
data	mandatory	STRING	Value of the data point

Service: getelementinfo

52269

Name: getelementinfo

Description: The service reads the properties of an element of the IoT tree.

Applicable to: Objects of the type `device`

Request data (field "data"):

Data field	Required field	Data type	Default	Description
adr	mandatory	STRING		URL of the element, which properties to be changed

Return data (field "data"):

Data field	Required field	Data type	Default	Description
identifier	mandatory	STRING		Identifier of the element
type	mandatory	STRING		Type of the element
format	optional	JSON object	blank	Format of the data or the service content
uid	optional	STRING	blank	
profiles	optional	JSON array	blank	
hash	optional	STRING	--	

Servicet: getidentity

54690

Name: getidentity
Description: The service reads the device information of the AL1351 and issues it.
Request data ("data" field): none
Return data ("data" field):

Data field	Required field	Data type	Description	
iot		Device	Device description as JSON object	
iot.name	mandatory	STRING		
iot.uid	optional	STRING		
iot.version	mandatory	STRING		
iot.catalogue	optional	ARRAY OF OBJECTS		
iot.deviceclass	optional	ARRAY OF STRING		
iot.serverlist	optional	ARRAY OF OBJECTS		
device	optional		AL1351	
device.serialnumber	optional		Serial number	
device.hwrevision	optional		Hardware version	
device.swrevision	optional		Software version	
device.custom	optional			
Security	optional		Security options	
security.securitymode	optional	ENUM	shows if the security mode is activated	
security.authscheme	optional	ENUM	shows the active authentication scheme	
security.ispasswordset	optional	BOOL	shows whether a password has been set	
security.activeconnection	optional	ENUM	shows the currently used communication interface	
			▪ tcp_if	unencrypted http connection at the IoT interface, port 80
			▪ tls_if	encrypted https connection at the IoT interface, port 443
			▪ fb_if	unencrypted http connection at the fieldbus interface, port 80

Service: getsubscriberlist

61191

Name: getsubscriberlist
Description: The service provides a list of all active subscriptions.
Request data ("data" field): none
Return data ("data" field): Array with the following data

Data field	Mandatory field	Data type	Description
adr	mandatory	STRING	Data source
datatosend	mandatory	ARRAY OF STRINGS	List with URLs of the subscribed data points
cid	mandatory	NUMBER	ID of the subscription
callbackurl	mandatory	STRING	Address to which IoT Core event notifications are to be sent;
duration	mandatory	STRING	Storage duration of the value

Example:

- **Request object:**

```
{
"code":"request",
"cid":4711,
"adr":"/getsubscriberlist"
}
```

- **Return object:**

```
{
"cid":4711,
"data":[
{
"adr":"/timer[1]/counter/datachanged/subscribe",
"datatosend":["/iolinkmaster/port[2]/iolinkdevice/pdin"],
"cid":1,
"callbackurl":"http://192.168.0.45:80/temp",
"duration":"lifetime"},
{
"adr":"/timer[1]/counter/datachanged/subscribe",
"datatosend":["/processdatamaster/temperature","/processdatamaster/voltage"],
"cid":2,
"callbackurl":"http://192.168.0.44:80/temp",
"duration":"lifetime"}
]
"code":200
}
```

Service: getsubscriptioninfo

61192

Name: getsubscriptioninfo
Description: The service provides information about an existing subscription (subscribe).

The following parameters of the existing subscription are to be used for the query:

- Value of the identifier cid (e.g. 4711)
- Number of the timer (e.g. timer[1])
- Name of the callback topic (e.g. B. temp)

Request data ("data" field):

Data field	Mandatory field	Data type	Description
callback	mandatory	STRING	Address to which IoT Core event notifications are to be sent; complete URL: http://ipaddress:port/path

Return data ("data" field):

Data field	Mandatory field	Data type	Description
subscription	mandatory	BOOL	Status of the transferred subscription parameter
datatosend	mandatory	ARRAY OF STRINGS	List with subscribed data points
cid	mandatory	NUMBER	ID of the subscribe request
callbackurl	mandatory	STRING	Address to which IoT Core event notifications are to be sent; complete URL: http://ipaddress:port/path

Example:

- **Request object:**

```
{
"code":"request",
"cid":4711,
"adr":"/timer[1]/counter/datachanged/getsubscriptioninfo",
"data":{
"callback":"http://192.168.0.44:80/temp"}
}
```

- **Return object:**

```
{
"cid": 4711,
"data":{
"subscription": true,
"datatosend":[
"/iolinkmaster/port[2]/iolinkdevice/productname",
"/iolinkmaster/port[2]/iolinkdevice/pdin",
"/processdatamaster/temperature"],
"callbackurl":"http://192.168.0.44:80/temp",
"duration":"lifetime"},
"code":200
}
```

Service: gettree

61193

Name: gettree

Description: The service reads the device description of the IO-Link master and outputs it as a JSON object. The output can be limited to a subtree of the device description.

Request data ("data" field):

Data field	Mandatory field	Data type	Description
adr	optional	STRING	Root element of the subtree
level	optional	STRING	max. level up to which the subtree is output ▪ no entry: all levels will be displayed ▪ 0: do not display sub-elements ("subs") ▪ 1: display sub-elements ▪ 2: display sub-elements up to the 2nd level ▪ 3: display sub-elements up to the 3rd level ... ▪ 20: display sub-elements up to the 20th level

Return data ("data" field):

Data field	Mandatory field	Data type	Description
identifier	mandatory	STRING	Identifier of the root element
type	mandatory	STRING	Type of the element
format	optional	JSON Object	Format of the data content
uid	optional	STRING	
profiles	optional	JSON-Array	
subs	mandatory	JSON-Array	Sub-elements
hash	optional	STRING	

Examples:

- output the complete device description

```
{
"code":"request",
"cid":4,
"adr":"/gettree"
}
```

- output the subtree counter[2] of the device description up to the 2nd level

```
{
"code":"request",
"cid":4,
"adr":"/gettree"
"data":{
"adr":"counter[2]",
"level":2}
}
```

Service: install

52343

Name: install
Description: The service installs the firmware stored in the container area of the device.
Applicable to: container
Request data (data): none
Return data (data): none

Service: iolreadacyclic

34178

Name: iolreadacyclic
Description: The service acyclically reads the parameter value of an IO-Link device. The parameter is accessed via IO-Link index and subindex.
Request data (field "data"):

Data field	Required field	Data type	Description
index	mandatory	NUMBER	IO-Link index of the parameter
subindex	mandatory	NUMBER	IO-Link subindex of the parameter

Response data (field "data"):

Data field	Required field	Data type	Description
value	mandatory	STRING	Value of the parameter; Value in hexadecimal format

Service: iolwriteacyclic

34177

Name: iolwriteacyclic
Description: The service acyclically writes the parameter value of an IO-Link device. The parameter is accessed via IO-Link index and subindex.
Request data (field "data"):

Data field	Required field	Data type	Description
index	mandatory	NUMBER	IO-Link index of the parameter
subindex	mandatory	NUMBER	IO-Link subindex of the parameter
value	mandatory	STRING	New value of the parameter; Value in hexadecimal format

Response data (field "data"): none

Service: querytree

61194

Name: querytree

Description: The service searches a device tree for the criteria profile, type and name and outputs a list with the URLs of the elements found. At least one of the search criteria must be specified. The service can only be executed on the root node of the machine.

Return data ("data" field):

Data field	Mandatory field	Data type	Description
profile	optional	STRING	Profile of the searched element
type	optional	STRING	Type of the searched element
name	optional	STRING	Type of the searched element

Return ("data" field):

Data field	Mandatory field	Data type	Description
urlList	mandatory	Array	Array with URLs of the found elements; URLs are separated by commas

Service: reboot

34176

Name: reboot

Description: The service reboots the device.

Request data (field "data"): none

Return data (field "data"): none

Example:

```
{
"code":"request",
"cid":4,
"adr":"firmware/reboot"
}
```

Service: Reset

61195

Name: Reset

Description: The service resets a connection to the initialisation state.

Request data ("data" field): none

Return data ("data" field): none

Example:

```
{
"code":"request",
"cid":4711,
"adr":"/connections/mqttConnection/MQTTSetup/mqttCmdChannel/status/reset"
}
```

Service: setblock

34186

Name: setblock

Description: The service simultaneously sets the values of several data points of a structure.

Request data (field "data"):

Data field	Required field	Data type	Description
datatoset	mandatory	ARRAY OF OBJECTS	List of data points and their new values; data points must support the service setdata
consistent	optional	BOOL	

Response data (field "data"): none

Example:

Request:

```
{
"code":"request",
"cid":4711,
"adr":"iotsetup/network/setblock",
"data":{
"datatoset":{
"ipaddress":"192.168.0.6",
"subnetmask":"255.255.255.0",
"ipdefaultgateway":"192.168.0.250",
"dhcp":0}
}
}
```

Response:

```
{
"cid":4711,
"code":233
}
```

Service: setdata

34195

Name: setdata

Description: The service sets the value of the data point.

Request data ("data" field):

Data field	Mandatory field	Data type	Description
newvalue	mandatory	STRING	New value of the element/data point
duration	mandatory	STRING	Duration of value storage ▪ lifetime: Value is saved with IoT Core; Value remains valid even after restart of the device ▪ uptime: Value is saved until the next restart of the device

Return data ("data" field): none

Example:

```
{"code": "request",
"cid":4711,
"adr": "devicetag/applicationtag/setdata",
```

```
"data":{
"newvalue": "ifm IO-Link master
"duration": "lifetime"}
}
```

Service: signal

33819

Name: signal
Description: The service starts the flashing of the status LEDs of the AL1351.
Request data (field "data"): none
Return data (field "data"): none

Example:

```
{
"code":"request",
"cid":4711,
"adr":"firmware/signal"
}
```

Service: start

61196

Name: start
Description: The service starts a connection.
Request data ("data" field): none
Return data ("data" field): none

Example:

```
{
"code":"request",
"cid":4711,
"adr":"/connections/mqttConnection/MQTTSetup/mqttCmdChannel/status/start"
}
```

Service: start_stream_set

52342

Name: start_stream_set
Description: The service starts the sequential transfer of multiple data segments.
Applicable to: Objects of type data
Request data (data):

Data field	Required field	Data type	Default	Description
size	mandatory	STRING		Total size of data to be transfered (number of bytes)

Return data (data): none

Service: stop

61197

Name: stop
Description: The servicestops a connection.
Request data ("data" field): none
Return data ("data" field): none

Example:

```
{
"code":"request",
"cid":4711,
"adr":"/connections/mqttConnection/MQTTSetup/mqttCmdChannel/status/stop"
}
```

Service: stream_set

52341

Name: stream_set
Description: The service transfers a data segment.
Applicable to: Objects of type data
Request data (data):

Data field	Required field	Data type	Default	Description
value	mandatory	BIN (BASE64)	*	Segment of binary data (BASE64 coded)

Return data (data): none

Service: subscribe

61198

Name: subscribe

Description: The service subscribes to the values of data points. The data points to be subscribed are transferred as a list. The IoT Core sends changes to the data sink defined in callback.

CSV formatted notifications can only be transmitted using the TCP protocol via an activated and configured MQTT channel.

Request data ("data" field):

Data field	Mandatory field	Data type	Description
callback	mandatory	STRING	Address to which IoT Core event notifications are to be sent; URL format: ▪ JSON: http://ipaddress:port/path ▪ JSON: ws:///path ▪ JSON: mqtt://ipadress:port/topic ▪ CSV: tcp://ipaddress:port/path
datatosend	mandatory	ARRAY OF STRINGS	List from URLs of data elements; Elements must support getdata
codec	optional	STRING	Format of the returned data ▪ json: JSON formatted ▪ csv: CSV with standard separator (,) ▪ csv0: CSV formatted with comma separator (,) ▪ csv1: CSV formatted with semicolon separator (;)
DURATION	mandatory	STRING	Duration of value storage ▪ lifetime: Value is saved with IoT Core; Value remains valid even after restart of the device ▪ uptime: Value is saved until the next restart of the device ▪ once: send only one notification, user must unsubscribe immediately

Return data ("data" field): none

Notification: JSON

```
{
"code":"event",
"cid":4711,
"adr":"",
"data":{
"eventno":"EventNo",
"srcurl":"SrcURL",
"payload":{
"eventurl":{"code":EventStatus,"data":EventData},
"datapointurl_1":{"code":DataStatus_1,"data":DataValue_1},
"datapointurl_2":{"code":DataStatus_2,"data":DataValue_2},
...}}
}
```

Notification: CSV

```
SrcURL,EventNo,EventStatus,EventData,DataStatus_1,DataValue_1,DataStatus_2,DataVal
ue_2,...
```

- `SrcURL`: Source of the event (data point on which subscribe command was listed)
- `EventNo`: Event number

- `EventStatus`: Status code of the event
- `EventData`: Event data
- DataStatus_1: Status code of the 1st element in list datatosend
- `DataValue_1`: Value of the 1st element in list datatosend
- `DataStatus_2`: Status code of the 2nd element in list datatosend
- `DataValue_2`: Value of the 2nd element in list datatosend
- ...

Service: unsubscribe

34197

Name: unsubscribe

Description: The service deletes an existing subscription. The service unsubcribe is successful if cid and the callback address are registered for an active subscription (subscribe). If the STRING "DELETE" is provided in callback, the IO-Link master deletes all active subscriptions.

Request data (field "data"):

Data field	Required field	Data type	Description
callback	mandatory	STRING	Address to which IoT Core event notifications are to be sent; complete URL: http://ipaddress:port/path

Response data (field "data"): none

Service: validation_useconnecteddevice

52340

Name: validation_connecteddevice

Description: The service checks, whether Device ID and Vendor ID of the connected IO-Link device match with the values of the datapoints `../validation_vendorid` and `../validation_deviceid`.

Applicable to: Objects of type `stucture`

Request data (data): none

Return data (data): none

15 Index

A

Access the ifm IoT Core 36
Accessories 78
Appendix 79
Application 80
Approvals / tests 82

C

Change history 6
Cleaning process 76
Communication, parameter setting, evaluation 11
Configuration 22
Configure IO-Link devices 32
Configuring IO-Link the master 72
Configuring the MQTT command channel 64
Connecting IO-Link devices for Class A operation 16
Connecting the device 17
Connecting the IoT ports 15
Creating a new notification 69

D

Deleting a notification 70
Digital inputs 12

E

Electrical connection 14, 83
Electrical data 80
Example
- Activate security mode 45
- Browsing device description 41
- Change name of the IO-Link master 57
- Change the parameter value of an IO-Link device 53
- Changing a subscription 59
- Checking subscriptions 61
- Clone the Data Storage of an IO-Link port 48
- Configuring the MQTT command channel 66
- GET request 36
- output subtree 40
- POST request 38
- Publish the temperature to an MQTT broker 67
- Read IO-Link process data (operating mode 49
- Read several parameter values of the IO-Link master simultaneously 41
- Read the parameter value of an IO-Link device 52
- Reading digital input (operating mode 50
- Reading properties of an element 39
- Request with authentication 46
- reset password 46
- Subscribing notifications via WebSocket 62
- Subscribing to notifications 42, 58
- Subscribing to notifications in CSV format 60
- Unsubscribing from notifications 60
- Update firmware 56
- Writing digital output (operating mode 50
- Writing IO-Link value (operating mode 49

Explanation of Symbols 6

F

Factory settings 77
Firmware
- Reboot the device 32
- Reset device to factory settings 32

First steps 39
Function 10
Functions and features 9

G

Gateway
- Reading device information 55
- Reading status and diagnostic information 55
- Resetting, rebooting and localising the device 54
- Setting the application tag 57
- Updating the firmware 56

General 7
General functions 39
GET request 36

I

ifm IoT Core 34, 84
Info
- Show device information 31

Inputs 81
Inputs / outputs 80
Interfaces 81
Internet of Things (IoT) 11
IO-Link 11
IO-Link devices
- Accessing parameters 52
- Indicating IO-Link events 54
- Reading an writing device information 54

IO-Link ports 16
- Activate data transfer to LR AGENT or LR SMARTOBSERVER 28
- Configuration of fail-safe values 31
- Configure operating mode 29
- Configuring data transfer to LR AGENT or LR SMARTOBSERVER 49
- Configuring device validation and data storage 47
- Indicating port events 52
- Reading / writing process data 49
- Set the device validation and data storage 30
- Setting the operating mode of pin 4 (US) 47

IO-Link Ports (Class A) 20
IO-Link supply 12
IoT
- Configure IP settings 25
- Configure security mode 26
- Configure the interface to LR AGENT or LR SMARTOBSERVER 27
- Configuring access rights 43
- Configuring IP settings 43
- Configuring security mode 44
- Configuring the LR AGENT or LR SMARTOBSERVER interface 44

IoT Core
- Diagnostic codes 38
- General information 35

IoT ports 19
IT security 8

L

LED indicators 19
Legal and copyright information 5
LR DEVICE 23

M

Maintenance, repair and disposal 76
Managing notifications 69

Mechanical data 82
Mount the device 13
Mounting 13
MQTT support 64

N

Note
- Connection states 65
- Security mode 45

Notes 14

O

Offline parameter setting 24
Operating and display elements 18
Operating conditions 82
Operation 75
Outputs 81
Overview 18
- IoT profile 85
- IoT services 93
- IoT types 92

P

Parameter setting 11
POST request 37
Preliminary note 5
Profile
- blob 85
- deviceinfo 86
- devicetag 86
- iolinkdevice_full 87
- iolinkmaster 87
- mqttCmdChannel 88
- mqttCmdChannelSetup 88
- mqttConnection 88
- mqttSetup 89
- network 89
- parameter 90
- processdata 90
- runcontrol 90
- service 90
- software 90
- software/uploadedablesoftware 91
- Timer 91

Programmers' notes 35
Purpose of the document 5

R

Reading and writing process data 73
Remarks 24
Replacing IO-Link device 76
Required background knowledge 7

S

Safety instructions 7
Safety symbols on the device 7
Searching for elements in the device tree 71
Security mode 11
Service
- factoryreset 93
- getblobdata 94
- getdata 94
- getdatamulti 95
- getelementinfo 95
- getsubscriberlist 97
- getsubscriptioninfo 98
- gettree 99
- install 100
- iolreadacyclic 100
- iolwriteacyclic 100
- querytree 101
- reboot 101
- Reset 101
- setblock 102
- setdata 102
- signal 103
- start 103
- start_stream_set 104
- stop 104
- stream_set 104
- subscribe 105
- unsubscribe 106
- validation_useconnecteddevice 106

Servicet
- getidentity 96

Setting the storage duration 42
Set-up 21
Status LEDs 19
Subscribing to notifications 58

T

Technical data 80

U

Updating the firmware 74, 76
Using the IoT-Core Visualizer 68
Using Web Socket 62
Using web-based management 75

V

Visual indication 11
Voltage supply 20
VPN connection 24