



Device manual
RFID compact device
IoT Protocol

UK

DTE105
DTE605
DTE8x5
DTE9x5

Contents

1.1	Licenses and trademarks	4
2	Preliminary note	5
2.1	Symbols used	5
3	Safety instructions	5
4	Functions and features	5
4.1	RFID evaluation unit DTE105	5
4.2	RFID compact devices DTE605, DTE8x5, DTE9x5	6
4.3	Configuration via Ethernet interface	6
4.4	Functions for commissioning	6
5	Function	7
5.1	RFID evaluation unit DTE105	7
5.1.1	Connection	7
5.1.2	AUX" voltage supply	8
5.1.3	Field bus connection Ethernet port 1 / port 2	8
5.1.4	Process connections "IO-1 ... IO-4"	8
5.1.5	Factory setting of the Ethernet parameters	9
5.1.6	Reset to factory settings	9
5.2	RFID compact devices DTE605, DTE8x5, DTE9x5	10
5.2.1	Connection	10
5.2.2	"PWR" voltage supply and process connections C/Qi or C/Qo	11
5.2.3	Field bus connection ETH1 / ETH2	11
5.2.4	Factory setting of the Ethernet parameters	11
5.2.5	Reset to factory settings	12
5.3	Allowed network infrastructure	13
5.3.1	Operating of the devices in an unprotected Ethernet network structure	13
5.3.2	Linear structure	13
5.3.3	Ring structure	13
5.3.4	Star structure	14
6	Installation	15
6.1	Devices DTE8x5/DTE9x5	15
7	Operating and display elements	16
7.1	LED indicators DTE105	16
7.1.1	LED AUX	16
7.1.2	LED Ethernet port 1 / port 2	16
7.1.3	LED SF (system failure)	16
7.1.4	LED BF (bus failure)	16
7.1.5	LEDs IO1 ... IO4	17
7.1.6	Special device LED indications	17
7.2	LED indicators DTE605, DTE8x5, DTE9x5	18
7.2.1	LED PWR	18
7.2.2	LED Signal DTE605	18
7.2.3	LED Signal DTE8x5/DTE9x5	18
7.2.4	LED LINK/ACT on interface ETH1/ETH2	18
7.2.5	Special device LED indicators	19

7.2.6	LED SF System failure.....	19
7.2.7	LED BF Bus failure.....	19
8	Putting into operation.....	20
9	Web interface.....	21
9.1	Verify and set the IP address of the PC.....	21
9.2	Tab Home.....	22
9.3	Tab "Network".....	23
9.4	Tab "Firmware – firmware update".....	24
9.5	Tab "Firmware – Read-/Write head update".....	25
9.5.1	Update of the Read-/Write head or the UHF module.....	26
9.6	Tab "IO-Port".....	27
9.6.1	Read-/Write head.....	27
9.6.2	IO-PORT IO-1 and IO-2.....	28
9.7	Tab "Monitor".....	28
9.8	Tab "System".....	29
9.9	Tab "SNTP".....	30
9.10	Tab "Info".....	31
9.11	Open Source license information.....	33
10	RFID IOT Core protocol.....	34
10.1	General information about the different RFID devices with IoT Core.....	34
10.1.1	Required firmware version of the device DTE105.....	34
10.1.2	Required firmware versions of the device DTE605.....	34
10.1.3	Required firmware versions of the devices DTE8x5, DTE9x5.....	34
10.2	IoT Core general informations.....	35
10.2.1	HTTP protocol format.....	35
10.2.2	TCP Port of the RFID devices with IoT Core.....	35
11	Access the ifm IoT Core.....	36
11.1	JSON objects.....	37
11.1.1	JSON requests.....	37
11.1.2	JSON Response.....	38
11.2	HTTP request call.....	39
11.2.1	GET request.....	39
11.2.2	POST request.....	40
11.3	IoT general functions.....	41
11.4	IoT Services.....	41
11.4.1	First possibility, call of request over URL and POST method.....	41
11.4.2	Second possibility, direct call of request over URL.....	42
11.4.3	First steps.....	43
12	General IoT Core services.....	44
12.1	Get Identity of the device.....	44
12.2	Read diagnostics from the IO channel.....	46
12.3	Subscribe to event.....	48
12.4	Unsubscribe from event.....	50
12.5	Get all subscriptions of the device.....	52
12.6	Connect the IoT Webserver with the device.....	53
13	RFID IoT Core profile.....	55

13.1	Configure IO-Ports	55
13.2	Access data of the IO ports	58
13.2.1	Read input C/Qi of the IO channel	58
13.2.2	Receive C/Qi from IO port automatically	60
13.2.3	Read input I/Q of the IO channel	62
13.2.4	Write output C/Qo of the IO channel	63
13.3	Access data of the HF ID tag	64
13.3.1	Read UID of HF ID tags	64
13.3.2	Receive UID from HF ID tag automatically	66
13.3.3	Read User memory of HF ID tags	69
13.3.4	Write User memory of HF ID tags	70
13.4	Access data of the UHF ID tag	71
13.4.1	Read EPC data of UHF ID tags	71
13.4.2	Receive EPC data of UHF ID tags automatically	73
13.4.3	Read memory bank of the UHF ID tags tags	75
13.4.4	Write User memory of the UHF ID tags	77
13.4.5	Write EPC data of the UHF ID tags	78
14	IoT Core HTTP status codes	79
15	Error codes of the device	80
15.1	Error group ID tag (F1FE)	80
15.2	Error group compact device (F4FE)	81
15.3	Error group Communication User – compact device (F5FE)	82
15.4	Error group Command error – compact device (F6FE)	83
16	Glossary	84

1.1 Licenses and trademarks

Microsoft® and Internet Explorer® are registered trademarks of Microsoft Corporation.

CoDeSys® is a registered trademark licensed by 3S-Smart Software Solutions GmbH.

All trademarks and company names are subject to the copyright of the respective companies.

2 Preliminary note

2.1 Symbols used

- Instruction
- > Reaction, result
- [...] Designation of pushbuttons, buttons or indications
- Cross-reference, Important note
-  Non-compliance can result in malfunction or interference
-  Information Supplementary note

3 Safety instructions

Please read the operating instructions prior to setup of the device. Ensure that the device is suitable for your application without any restrictions. If the operating instructions or the technical data are not adhered to, personal injury and/or damage to property can occur.

4 Functions and features

4.1 RFID evaluation unit DTE105

The RFID evaluation unit integrates an Ethernet interface and 4 channels for the connection of field devices . Each channel can be used either for the connection of a HF read/write head, e.g. ANT512, ANT513, ANT600 ... or as input/output to IEC 61131 .

The device

- controls the data exchange to the read/write heads or the sensor/actuator level .
- communicates with the higher control level via Ethernet .
- allows device configuration via a web server . Application examples:
- Material flow control in production lines
- Warehouse management by the automatic detection of stored products
- Tank management, order picking or product tracking

Application examples:

- Material flow control in production lines
- Warehouse management by the automatic detection of stored products
- Tank management, order picking or product tracking

4.2 RFID compact devices DTE605, DTE8x5, DTE9x5

The RFID Compact device integrates UHF Read-/Write head and a TCP/IP interface to communicate with a PC or Controller with integrated Ethernet interface. Additionally, 2 IO channels are available which can be configured as input/output to IEC 61131.

The device

- DTE605 can detect one HF ID tag according to RFID-Standard ISO 15693
- DTE8x5/DTE9x5 can detect up to 16 UHF ID tags according RFID-Standard EPC Class1 GEN2/ISO 18000-63 simultaneously
- control up to 2 IEC 61131 input /outputs
- communicates with the higher control level over Ethernet.
- allows device configuration via a web server.

Application examples:

- Material flow control in production lines
- Warehouse management by the automatic detection of stored products
- Tank management, order picking or product tracking

4.3 Configuration via Ethernet interface

- 10 Mbps and 100 Mbps
- Protocol MQTT JSON over TCP / IP - Transport Control Protocol / Internet Protocol
- IT functionality: HTTP server
- M12, twisted pair

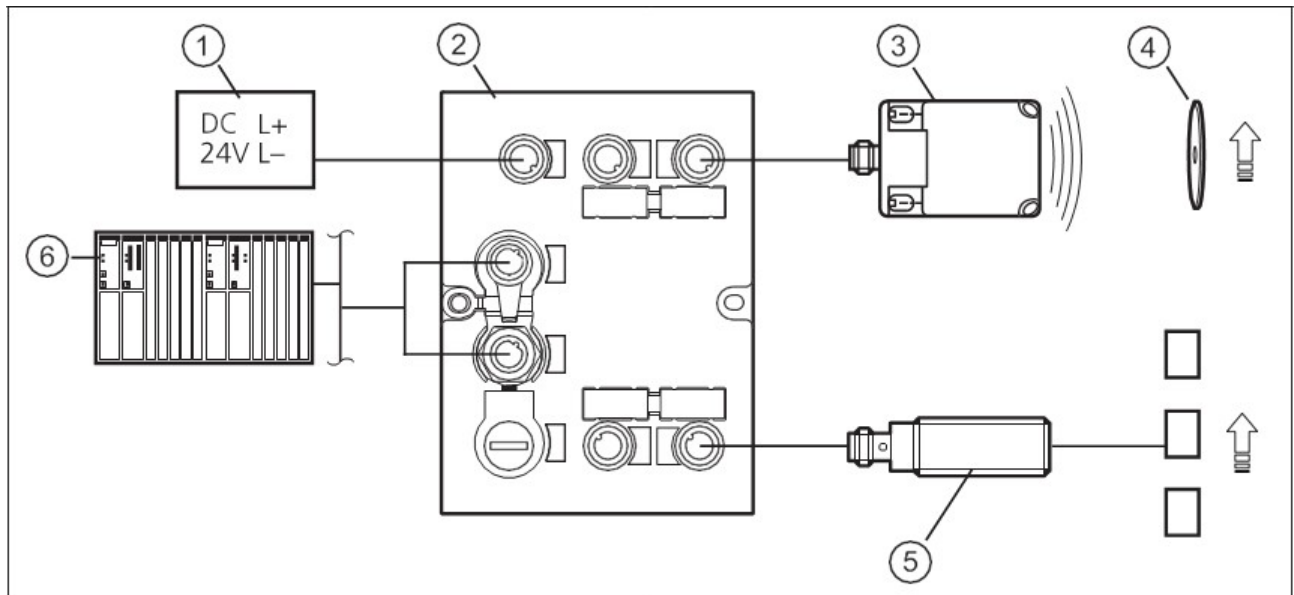
4.4 Functions for commissioning

Via the integrated Webserver it is possible to

- read and write the UID data of the HF ID tag with the device DTE605
- read and write the EPC data of the UHF ID tag with the DTE8x5/DTE9x5 devices
- read the TID data of the UHF ID tag with DTE8x5/DTE9x5 devices
- read and write the User data area of the HF/UHF ID tag
- write to the User data area of the HF / UHF ID tag
- read the input of the IO channels
- write to the output of the IO channels
- read the device information of the device
- read the device information of the integrated Read-/Write head
- update the firmware of the integrated Read-/Write head

5 Function

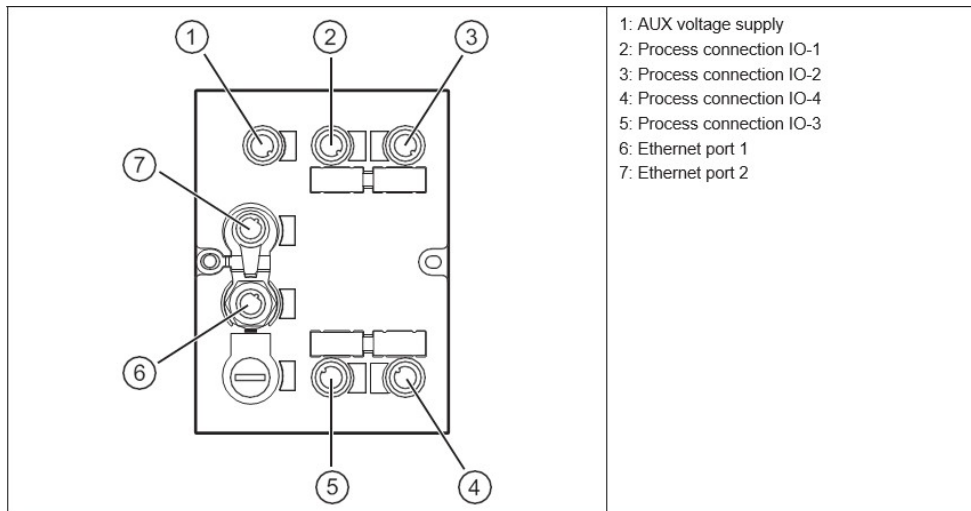
5.1 RFID evaluation unit DTE105



- | | |
|--|------------------|
| 1: Power supply | 4: Tag |
| 2: Evaluation unit | 5: Sensor |
| 3: Read/write head type ANT51x /ANT41x | 6: Ethernet host |

The evaluation unit processes data of up to 4 read/write heads (type ANT51x/ANT41x/ANT60x) or IEC 61131 inputs / outputs . The mode of operation for each port can be set individually via the Ethernet controller .

5.1.1 Connection



5.1.2 **AUX" voltage supply**

► Connect the device to the voltage supply using an M12 connection cable .

	Pin	Connection"
	1	24 V DC
	2	not used
	3	0 V
	4	not used
	5	not used

5.1.3 **Field bus connection Ethernet port 1 / port 2**

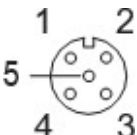
► Connect the device to the Ethernet host using a suitable M12 Ethernet connection cable .

	Pin	Connection
	1	TD+
	2	RD+
	3	TD-
	4	RD-



5.1.4 **Process connections "IO-1 ... IO-4"**

Each process connection can be used as input/output to IEC 61131 or for connection of an RFID read/write head type ANT51x/ANT41x .

	Pin	Connection
	1	L+
	2	Switching input (I/Q)
	3	L-
	4	Switching output (C/Qo) or input (C/Qi)
	5	not used



 Please note that the total current consumption of the device must not exceed 3 A.

You can find information about the matching read/write heads on our website at:
<https://www.ifm.com> → Data sheet search → e.g. ANT4, ANT5 or ANT6

5.1.5 Factory setting of the Ethernet parameters

The following values are preset on delivery of the device:

Parameters	Factory setting
IP address	192 .168 .0 .79
Gateway address	192 .168 .0 .100
Subnet mask	255 .255 .255 .0
Auto-negotiation	on
DHCP	off



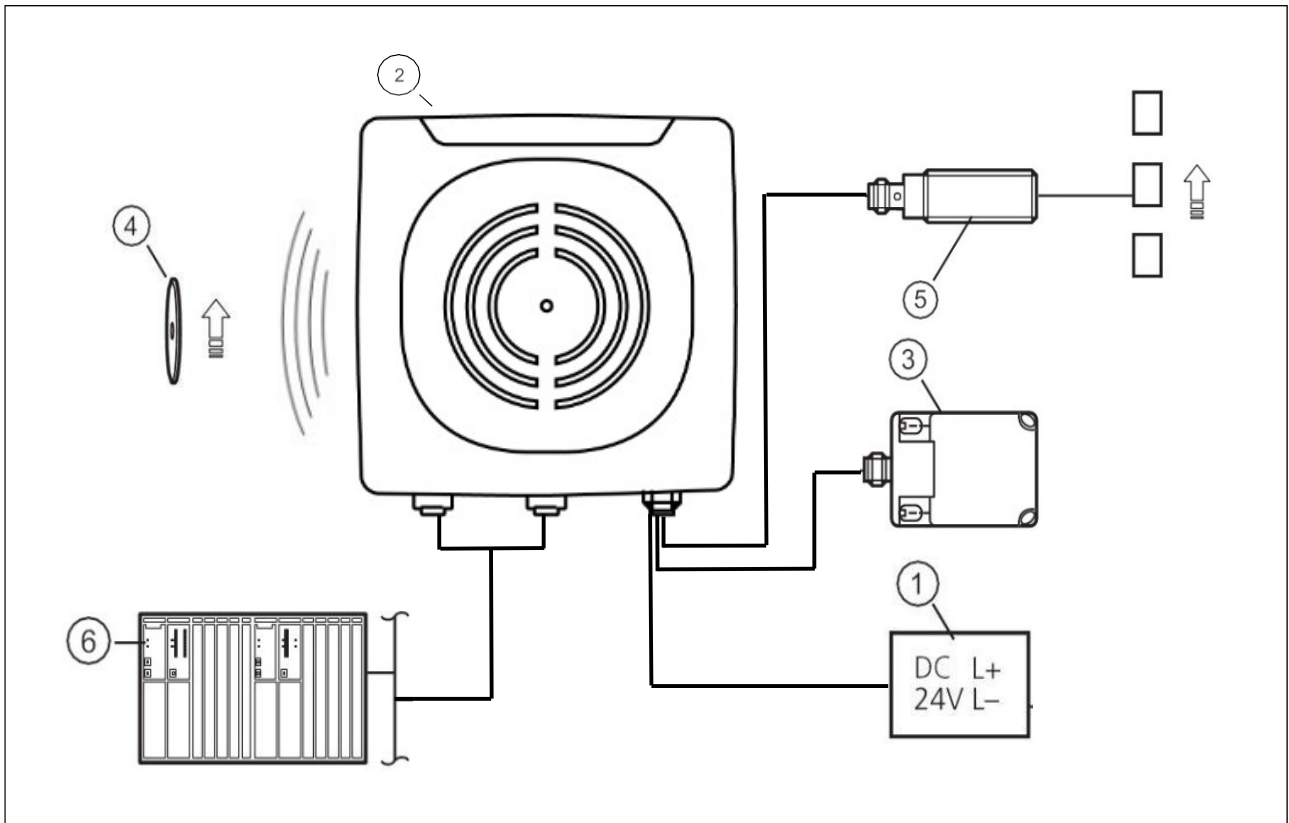
The settings can be changed via the web server of the device or via the Ethernet host.

5.1.6 Reset to factory settings

The Ethernet parameters and the password of the device can be reset to the factory settings. Take the following steps:

- ▶ Remove all cable connections from the device.
- ▶ Insert an electrically conductive bridge between pin 2 and pin 4 on the process connection voltage supply PWR.
- ▶ Connect the unit to the voltage supply.
 - The LEDs of the signal bar (yellow) are on one after the other. Then LED 4 of the signal bar (yellow) flashes at 8 Hz.
- ▶ As soon as the LEDs of the signal bar (yellow) flash at 8 Hz, disconnect the unit
- ▶ Remove the bridge.
- ▶ Connect the device to the voltage.
- ▶ The settings are reset.

5.2 RFID compact devices DTE605, DTE8x5, DTE9x5

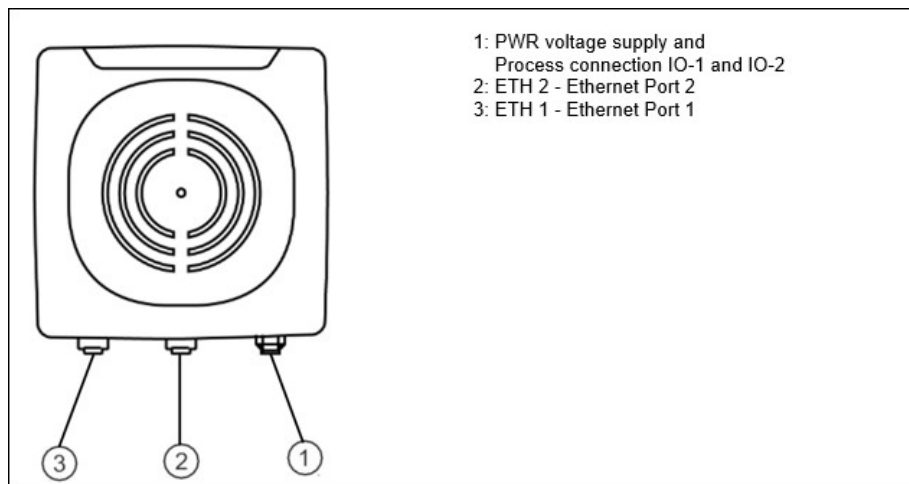


- 1: Voltage supply
- 2: DTE8x4/DTE9x4 device unit
- 3: Actuator
- 4: ID tag

- 5: Sensor
- 6: Ethernet host

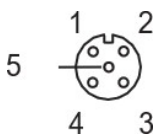
The device processes data from the integrated Read-/Write head and the two IEC 61131 inputs / outputs. The mode of operation for the two IO channels can be set individually via the Ethernet host or the integrated Webserver.

5.2.1 Connection



5.2.2 "PWR" voltage supply and process connections C/Qi or C/Qo

- ▶ Connect the device to the voltage supply using an M12 connection cable.

	Pin	Connection
	1	24V DC
	2	Switching output (C/Qo) or input (C/Qi) 2
	3	0V
	4	Switching output (C/Qo) or input (C/Qi) 1
	5	not used

 Please note that the total current consumption of the device must not exceed the value of 0,5 A.

5.2.3 Field bus connection ETH1 / ETH2

- ▶ Connect the device to an Ethernet host using a suitable M12 Ethernet connection cable.

	Pin	Connection
	1	TD+
	2	RD+
	3	TD-
	4	RD

 Screened connection cable required.

5.2.4 Factory setting of the Ethernet parameters

The following values are preset on delivery of the device:

Parameters	Factory setting
IP address	192.168.0.79
Gateway address	192.168.0.100
Subnet mask	255.255.255.0
Auto-negotiation	on
DHCP	Off

 The settings can be changed via the web server of the device or via the Ethernet host.

5.2.5 Reset to factory settings

The Ethernet parameters and the password of the device can be reset to the factory settings. Take the following steps:

- ▶ Remove all cable connections from the device.
- ▶ Insert an electrically conductive bridge between pin 2 and pin 4 on the process connection voltage supply PWR.
- ▶ Connect the unit to the voltage supply.
 - The LEDs of the signal bar (yellow) are on one after the other. Then LED 4 of the signal bar (yellow) flashes at 8 Hz.
- ▶ As soon as the LEDs of the signal bar (yellow) flash at 8 Hz, disconnect the unit
- ▶ Remove the bridge.
- ▶ Connect the device to the voltage.
- ▶ The settings are reset.

5.3 Allowed network infrastructure

The devices can be integrated in the Ethernet network infrastructures described below if the following safety instructions are implemented and observed

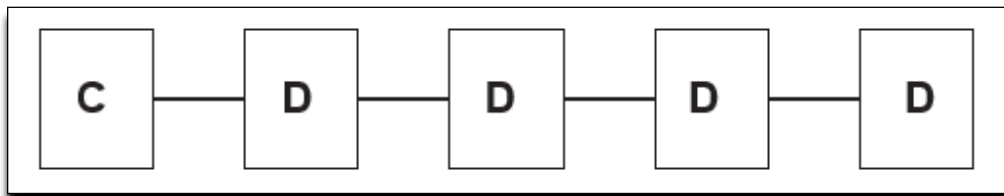
5.3.1 Operating of the devices in an unprotected Ethernet network structure

- ⚠ Check and restrict access options to the device:
 - Restrict access to authorized persons.
 - Do not connect the device to open networks or the internet.

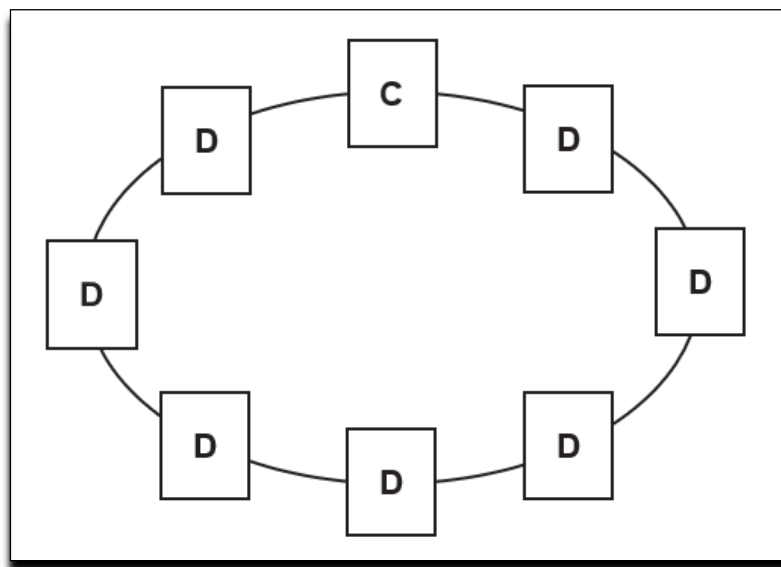
- ⚠ If access from the internet is inevitable:
 - ▶ choose a safe method to connect with the device (e. g. VPN).
 - ▶ Use encrypted data transmission (e. g. https / TLS).

- 👤 If the device is operated in an unprotected network environment, notice that
 - > unauthorized read or write access to data is possible.
 - > unauthorized manipulation of the device function is possible.

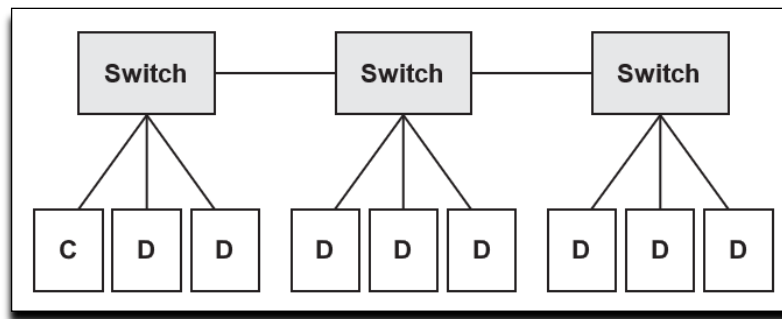
5.3.2 Linear structure



5.3.3 Ring structure



5.3.4 Star structure



C = Controller D = Device

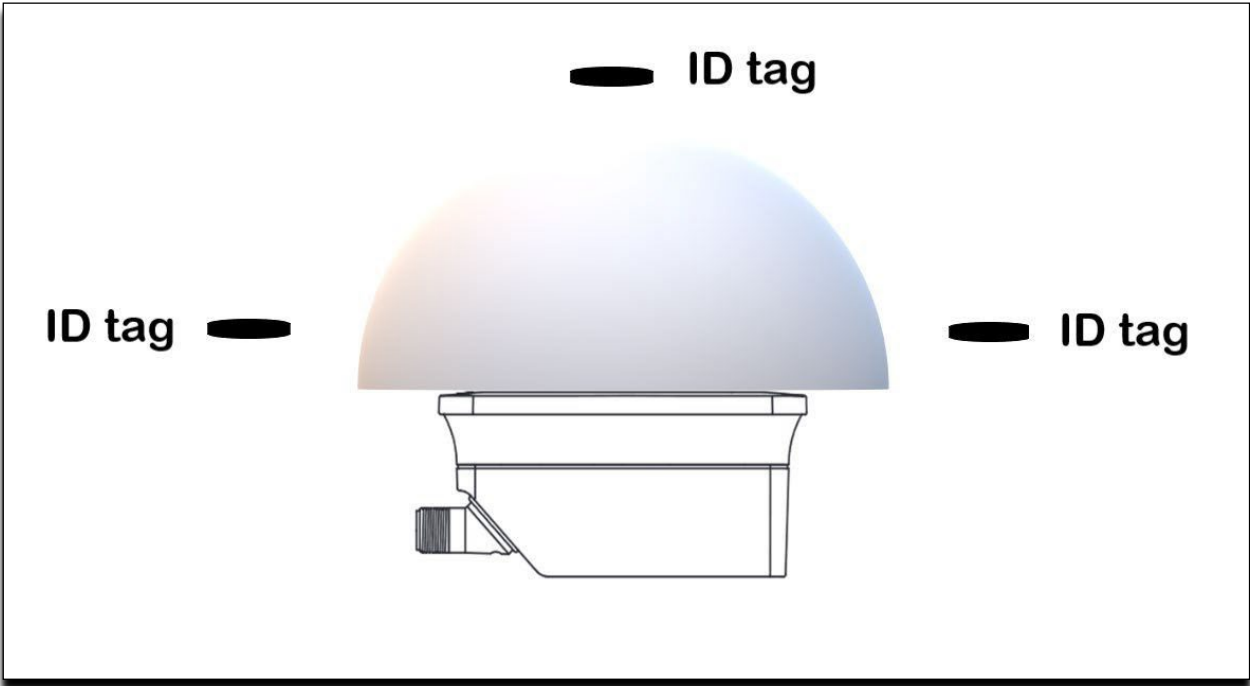
6 Installation

You can find information about installation and electrical connection in the operating instructions for the devices at <https://www.ifm>

6.1 Devices DTE8x5/DTE9x5

-  To avoid malfunctions of the UHF functions, do not place any electrically conductive objects within 10 cm of the built-in UHF antenna.
-  The ID tags should be placed outside a hemisphere, see table below, so that an error-free reading or writing is possible.

Device	Radius for ID Tags	Remark
DTE8x4/DTE9x4	$r = 10\text{ cm}$	Placement within the hemisphere may exceed the limits for the ID tag



7 Operating and display elements

7.1 LED indicators DTE105

The device indicates the current status of the interfaces via the status LEDs .

7.1.1 LED AUX

LED green	LED yellow	Status	Note
Off	Off	No voltage supply	UAUX < 5 V
On	Flashes at 2 Hz	Voltage supply too low	$5\text{ V} \leq \text{UAUX} \leq 18\text{ V}$
On	Flashes at 8 Hz	Firmware update running	Do not switch off power supply
On	Off	Voltage supply OK	$18\text{ V} \leq \text{UAUX} \leq 36\text{ V}$

7.1.2 LED Ethernet port 1 / port 2

LED green	LED yellow	Status	Note
Off	Off	No connection to another Ethernet counterpart	Link status "no link"
On	Off	Connection to Ethernet counterpart is established, no data exchange	Link status "link", "no traffic"
On	Flashes sporadically	Connection to Ethernet counterpart is established, data exchange running	Link status "link", "traffic"

7.1.3 LED SF (system failure)

LED red	LED green	Status	Note
Off	Off	No voltage supply	Check the voltage supply
Off	On	Normal operation	-
Flashes	Off	Error at channel level	- Overload - Temperature - Internal fault
On	Off	Error at device level	- Undervoltage - Temperature
Flashes	Flashes	Self-test	Start phase of the device

7.1.4 LED BF (bus failure)

LED red	LED green	Status	Note
Off	Off	No voltage supply	Check the voltage supply
Off	Flashes	Connection to the host controller is established, there is no data exchange	-
Off	On	Connection to the host controller is established, there is data exchange	-
Flashing	Off	Connection to the host controller is established, no valid configuration	Check configuration
On	Off	No connection to the host controller	Check connection
Flashes	Flashes	Self-test	Start phase of the device

7.1.5 LEDs IO1 ... IO4

The LED indications of the process connections are different for each connection configuration .

Use as input to IEC 61131

LED green	LED yellow	Status	Note
Off	Off	Interface deactivated	Interface via Ethernet host not configured
On	Off	Interface activated, input on L level (0 V)	-
On	On	Interface activated, input on H level (24 V)	-
Flashes at 8 Hz	Flashes at 8 Hz	Overload or short circuit	-

Use as output to IEC 61131

LED green	LED yellow	Status	Note
off	off	Interface deactivated	Interface via Ethernet host not
On	off	Interface activated, output L-active (0 V)	-
On	On	Interface activated, output H-active (24 V)	-
off	off	Interface deactivated	Interface via Ethernet host not

Use with read/write heads

LED green	LED yellow	Status	Note
Off	Off	Interface deactivated	Interface via Ethernet host not configured
Flashes at 2 Hz	Off	Interface activated, antenna off	-
On	Off	Interface activated, tag not in the field	-
On	On	Interface activated, tag in the field	-
Flashes at 8 Hz	Flashes at 8 Hz	Overload, short-circuit or communication error	-

7.1.6 Special device LED indications

LED	Status	Note
Green AUX LED on Yellow AUX LED flashes at 8 Hz Yellow IO1 ... IO4 LEDs flash at 8 Hz	Device is in the service mode "emergency system started" .	A firmware update is necessary and can be executed via the web server .
Green AUX LED on Yellow AUX LED flashes at 8 Hz Green IO1 ... IO4 LEDs flash at 8 Hz Yellow IO1 ... IO4 LEDs flash at 8 Hz	Major error, device has to be returned .	Hardware fault or permanent data in the device is corrupt .
Green AUX LED on Yellow AUX LED flashes at 8 Hz Yellow IO3 LED flashes at 8 Hz	Reset to factory settings	-

7.2 LED indicators DTE605, DTE8x5, DTE9x5

The device indicates the current status of the interfaces via the status LEDs .

7.2.1 LED PWR

LED PWR (1x green)	State	Note
On	Voltage supply OK	$18\text{ V} \leq U_{\text{PWR}} \leq 36\text{ V}$
Flashing at 2 Hz	Antenna (HF field) is deactivated	In this case the

7.2.2 LED Signal DTE605

LED signal bar LED 1	LED signal bar LED 2	LED signal bar LED 3	LED signal bar LED 4	State
Off	Off	Off	Off	No ID tag detected
On	Off	Off	Off	One ID tag detected $0 \leq \text{RSSI value} \leq 1$
On	On	Off	Off	One ID tag detected $2 \leq \text{RSSI value} \leq 3$
On	On	On	Off	One ID tag detected $4 \leq \text{RSSI value} \leq 5$
On	On	On	On	One ID tag detected RSSI value > 5

7.2.3 LED Signal DTE8x5/DTE9x5

LED signal bar LED 1	LED signal bar LED 2	LED signal bar LED 3	LED signal bar LED 4	State
Off	Off	Off	Off	No ID tag detected
On	Off	Off	Off	1 ID tag detected
On	On	Off	Off	2 ID tag detected
On	On	On	Off	3 ID tag detected
On	On	On	On	4 or more ID tags detected

7.2.4 LED LINK/ACT on interface ETH1/ETH2

LED green	LED yellow	Status	Note
Off	Off	No connection to another Ethernet counterpart	Link status "no link"
On	Off	Connection to Ethernet counterpart exists, no data exchange	Link status "link", "no traffic"
On	Flashes sporadically	Connection to Ethernet counterpart exists, data exchange running	Link status "link", "traffic"

7.2.5 Special device LED indicators

LED	State	Note
PWR LED (green) ON LEDs of signal bar (yellow) flashing at 8 Hz	Device is in the service mode "emergency system started"	A firmware update is necessary and can be executed via the web server
PWR LED (green) On LEDs of signal bar (yellow) flashing at 8 Hz	Major error, device has to be returned	Hardware fault or permanent data in the device are corrupt
PWR LED (green) On The LEDs of the signal bar (yellow) are on one after the other. Then LED 4 of the signal bar (yellow) flashes at 8 Hz	Reset to factory settings	-

7.2.6 LED SF System failure

LED red	LED green	State	Note
Off	Off	No voltage supply or fault state	Check the voltage supply.
Off	On	Normal operation	Connection to the Ethernet host is established. The device is configured. The data transfer is running.
Flashes	Off	Connection fault	A connection to the Ethernet host was not established. ► Check the voltage supply. ► Check the configuration of the device.
On	Off	Device fault	Overtemperature: ► Try to reduce the ambient temperature or assemble a heatsink on the back of the device. Hardware error of the device: ► Restart the device. If the error remains, send the device for service
Flashes	Flashes	Self test	Starting phase of the device

7.2.7 LED BF Bus failure

LED red	LED green	State	Note
Off	Off	No IP address or no voltage supply	► Check the voltage supply. ► If DHCP is activated, check presence of a DHCP server in the network.
Off	Flashes	Link established, but no data exchange	At least an Ethernet link from one of the Ethernet ports was detected.
Off	On	Connection to host established, data exchange running	Operational Mode of the device
Flashes	Off	No valid configuration	A timeout was found with one of the existing Ethernet connections. ► Check the status of the connection in the Ethernet host.
On	Off	No connection to host	► Check Ethernet cable or power supply of the Ethernet switches
Flashes	Flashes	Self-test	Starting phase of the device

8 Putting into operation

- ▶ Connect the device according to the operating instructions.



After connecting the operating voltage, the device is ready for use.
The green LED

- AUX for DTE105 device
- PWR LED for DTE605, DTE8x5, DTE9x5 devices

with LED LINK/ACTIVITY will light up after the device is ready and connected to the Ethernet network.

9 Web interface

The devices are equipped with an integrated web server that allows to

- read and write data of the ID tag
- read from the User data area of the ID tag
- write to the User data area of the ID tag
- update the firmware of the device
- update the firmware of the built-in Read-/Write head
- update the firmware of the connected Read/Write heads

The settings are made via a web browser, e.g. Microsoft Internet Explorer® as from V7.0
To access the web server, connect the device to a PC using a suitable M12 Ethernet connection cable.

Please note that the device and the PC must be set to the same IP address range.
Default after factory reset: 192.168.0.79, see → [Verify and set the IP address of the PC.](#)

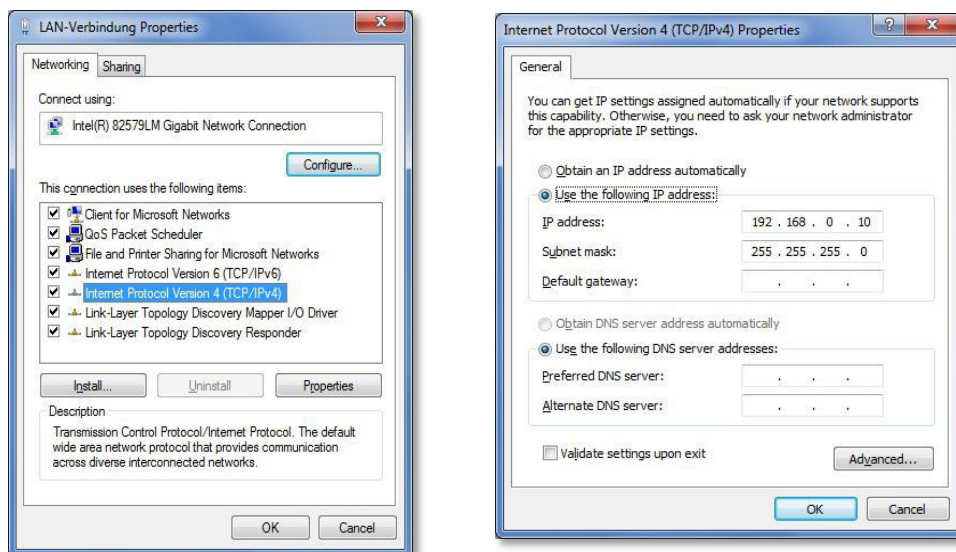


9.1 Verify and set the IP address of the PC

- ▶ Activate menu "Internet Protocol Version 4 (TCP/IPv4) Properties".

The Windows menu "Internet protocol (TCP/IP) Properties" is accessible for example via:
Start → Control Panel → Network and Sharing Center → Change adapter settings → Local Area Connection → Properties.

- ▶ Select the menu item "Use the following IP address".
- ▶ Verify and set the IP address, if necessary (here e.g. 192.168.0.10).
- ▶ Enter the subnet mask (255.255.255.0).
- ▶ Leave default gateway blank.
- ▶ Confirm the settings with [OK].



Changes in the network settings of the PC require extended user rights. Contact your system administrator.

9.2 Tab Home



This page slightly differs in its appearance depending of the device.



- This is the main menu from where all functions of the device can be accessed. The user can select the language of the device web interface.

9.3 Tab "Network"

Network settings:

TCP/IP parameter	Current settings	New settings
IP address	192.168.0.79	. . .
Subnet mask	255.255.255.0	. . .
Default gateway	192.168.0.100	. . .
DHCP	off	☐ on ☒ off
BOOTP	off	☐ on ☒ off
Port 1 parameter		
Autonegotiation	on	☒ on ☐ off
Port speed	Duplex 100MB	Duplex 100MB ▾
Port 2 parameter		
Autonegotiation	on	☒ on ☐ off
Port speed	Simplex 10MB	Simplex 10MB ▾

WARNING: Changing the IP parameters may cause a loss of connection.

This menu allows different settings Ethernet interface the device to be changed. Ask what settings for the integration of the device into the existing network needed your network administrator.

If the device is connected directly to the PC, a static IP address setting is recommended. Check → [Verify and set the IP address of the PC](#) .



Make sure that the network address of the PC is located within the address range of the device and that it differs from it.

Example:

Red marked address settings must be the same, if Subnet Mask of the device is 255.255.255.0

Ethernet IP address of the PC: 192.168.0.10

IP address of the evaluation: 192.168.0.79



Attention: Be careful when setting the IP address of the device. This address should not exist in the Ethernet network.

9.4 Tab "Firmware – firmware update"



This page slightly differs in its appearance depending of the device.

Firmware
Firmware identification

Name	Number	Version	Date
DTE Firmware	113	V4.	2021-
Emergency System	112	4.	-
Bootloader	-	1.	-

Firmware update

Choose the new firmware file (.nxf) you want to install:

No file selected.

Submit your file by clicking on 'transfer'. The transfer will take a few seconds.

WARNING: Do not interrupt power or disconnect the network cable while the transfer is in progress!

Read-/ Write head firmware

[For update of the Read-/Write head firmware click here.](#)



Please update the Read-/Write head firmware always before update the device firmware !

This menu allows to update the firmware of the device.

- Open the "Firmware" tab on the browser interface.
- Choose firmware file of the device and commit via button [transfer].
- Note: For update of the device firmware only files with extension ".nxf" are accepted.
- For update of the Read-/Write head firmware select the Link "For update of the Read-/Write head firmware click here". (Check → Tab "Firmware – Read-/Write head update")



Do not power off the device while the firmware update is running. A firmware update can take up to 3 minutes.

In case of an incomplete firmware update, the device will start up with the Emergency system.

From there a new firmware update can be started.

After the firmware update process is finished following information is displayed.

- To take effect of the new firmware set the checkbox and press the button "Reset".

Firmware

Firmware transfer and update succeeded!

You should reset the device to apply the changes.
Firmware is being restarted and connections may be interrupted or time out!

☐ Please confirm you want to reset the device.

Reset

- !
- Please note that the device needs up to 2 minutes to restart after a firmware update. During this time the LEDs of the device are switched off.

9.5 Tab "Firmware – Read-/Write head update"

- !
- This page slightly differs in its appearance depending of the device.

Read-/Write head information:

RWH		
Device type:	DTE	
Hardware version:	1	
IDLink SW:	12	
Firmware version:	1.026	Update
Reader Type:	RS500	1-1740 4515
Reader Firmware Version:	1.8.12.240	Update
Reload		

This menu shows under section "Read-/Write head information" following information.

- Device type
- Hardware and IDLINK software version
- Firmware version
- Reader type
- Reader firmware version

Button	Function
Update	Update the firmware of the Read-/Write head or update the firmware of the UHF module (Reader firmware) Check → Update of the Read-/Write head or the UHF module
Reload	Reload of this page

9.5.1 Update of the Read-/Write head or the UHF module

Antenna firmware

Firmware identification

Type of antenna	Firmware version
DTE	1.
RS500	1.8.

Firmware update

Choose the new antenna firmware file (.afw) you want to install:

Durchsuchen...

DTE0X_1_026.afw

Submit your file by clicking on the button. The transfer will take a few seconds.

WARNING: Do not interrupt power or disconnect the network cable while the transfer is in progress!

Transfer

Button	Function
Browse	Select the firmware file for the update process Note: For update of the Read-/Write firmware only files with extension “.afw” are accepted. For update of the UHF module firmware only files with extension “.rfw” are accepted.
Transfer	Start the firmware update process

After the firmware update process is finished following information is displayed.

Antenna firmware

Firmware transfer and update succeeded!

 If the firmware update fails or the Read-/Write head is not detected by the device at the selected IO-port, the Read-/Write head is accessible via web browser and the following URL:

http://<IP-ADDRESS>/rwhupdate?ioport=<IO-CHANNEL>1&anttype=<ANTENNA_TYPE>&fwVersion=<NUMBER>&setLng=<LANGUAGE>

Parameter name	Description	Remark
IP-ADDRESS	IP address of the device [XXX.XXX.XXX.XXX]	IPv4 address
IO-CHANNEL	IO-Channel number [1...4]	-
ANTENNA_TYPE	Article number of the Read-/Write head [e.g. ANT51x, DTE60x, DTE80x, DTE90x]	6-digit article number
NUMBER	Firmware number [01]	2 digit number. Shall be set to “01”
LANGUAGE	Language of the website [de, en, es, fr, it, ko, pt, ru, zh]	-

Example of URL for ANT513 connected at IO port 1 at DTE105 :

<http://192.168.0.79/rwhupdate?ioport=1&anttype=ANT600&fwVersion=1&setLng=en>

Example of URL for DTE605 :

<http://192.168.0.79/rwhupdate?ioport=1&anttype=DTE60X&fwVersion=1&setLng=en>

Example of URL for DTE805 :

<http://192.168.0.79/rwhupdate?ioport=1&anttype=DTE80X&fwVersion=1&setLng=en>

Example of URL for DTE905 :

<http://192.168.0.79/rwhupdate?ioport=1&anttype=DTE90X&fwVersion=1&setLng=en>



After finishing the firmware update of the Read-/Write head, enter the URL of the device to return to the main menu (→ Tab "Home").

9.6 Tab "IO-Port"

IO-Port Configuration		
Global	Current settings	New settings
Fail-safe	off	☐ on ☒ off
Mode	RWH	RWH
Data hold time	0	0 ms
IO-1	Current settings	New settings
Mode	Off (0 bytes)	Input
Data hold time	0	0 ms
IO-2	Current settings	New settings
Mode	Off (0 bytes)	Output
Data hold time	0	0 ms
		Activate and save Cancel

This menu allows to configure the Read-/Write head and the IO-Ports of the device.

9.6.1 Read-/Write head

The user data length of the read / write head cannot be configured. The maximum length can be up to 1024 characters.



Parameter "Data hold time" define how long the RFID data are kept stable. This is helpful if the time interval, in which the ID tag data are available, is shorter than the Ethernet host can read these over the fieldbus.

9.6.2 IO-PORT IO-1 and IO-2

The IO-channels can be configured to mode “Off”, “Input” or “Output”.

Mode	Function
Off	No function, inactive
Input	IEC 61131 input, payload size 20 bytes
Output	IEC 61131 output, payload size 20 bytes
Input (2 bytes)	IEC 61131 input, payload size 2 bytes
Output (2 bytes)	IEC 61131 output, payload size 2 bytes

Parameter “Data hold time” define how long the IO input data are kept stable. This is helpful if the time interval, in which the IO input is sampled, is shorter than the Ethernet host can read these over the fieldbus.

Button	Function	Remark
Activate and save	The settings are activated and stored non-volatile	After next power-on the stored settings are activated. If the host connect to the device and write a new IO-Port configuration the stored values are overwritten
Cancel	Discard changes	-

9.7 Tab "Monitor"

Port monitoring					Fieldbus state: 
RWH					
No	Length	EPC	RSSI	Readings	
1	8	eb 80 b0 44 3c 00 00 1a	-51	54268	
2	16	e2 00 40 80 71 11 00 00 00 00 00 30 18 30 58 33	-45	54271	
3	8	e0 04 01 00 0b ad f6 8a	-39	54250	
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					

This menu shows the data of the Read-/Write head and the state of IO ports of the device.

9.8 Tab "System"

System settings

Password protection configuration

Parameter	Current settings	New settings
Password protection:	off	☒ on ☐ off

Parameter	Value
User name:	admin
Old password:	
New password:	
Confirm new password:	

Submit

Cancel

This menu allows to define a password to protect the device against unauthorized access. To enable the password protection the button "New settings" has to be set to "on".

Parameter	Setting	Note
Username	admin	User name could not be changed
Old password	XXXXXX	Default password is "admin". For setting the password for the first, please enter the default password
New password	XXXXXX	Up to 10 characters are allowed
Confirm new password	XXXXXX	Must correspond to the parameter setting "New password"

- ▶ Click [Submit] to save the password
- ▶ Click [Cancel] to delete all parameter settings



If the password is lost the default password "admin" can be retrieved by a reset to factory settings (→ [Reset to factory settings](#)).

9.9 Tab “SNTP”

SNTP settings

To change the settings fill out the form in the table below and press 'submit'.

Parameter	Current settings	New settings
NTP support:	off	☒ on ☐ off
IP address from NTP server:	0.0.0.0	172 . 19 . 55 . 37
Offset to UTC:	UTC	UTC +0 v
State	not used	
Last sync	-	

Submit

If an SNTP server is present in the Ethernet network the internal clock of the device can be synchronized with the external time server.

► Click on button [Submit] to transmit the settings to the device.

SNTP settings

Parameter	Current settings
NTP support:	on
IP address from NTP server:	172.19.55.37
Offset to UTC:	UTC

9.10 Tab "Info"

This menu shows the following information about the device.

Hardware:

Parameter	Value
Power supply state:	fully operable
Temperature:	55°C 131°F
Reader temperature:	45°C 113°F
System time:	00:27:26.308
System date:	2011-01-01

- Temperature of the device



At temperatures > 85° C / 194° F, a heat sink must be mounted on the back of the device.

- Temperature of the UHF module
 - For a proper operation of the device the UHF module temperature shall not exceed 85°C.



At temperatures > 85° C / 194° F, reduce the HF power read and write.

- System time
 - Time gone since device powered up.
 - In case of setting time by SNTP service, the System date and System time is set automatically.

Production parameters:

Parameter	Value
Product article number:	DTE
Production number:	12345678
Version PermData:	1

- Show the ifm article and production number. Version PermData is used for compatibility checks if newer software is installed

Read-/Write head information:

RWH		
Device type:	DTE	
Hardware version:	1	
IDLink SW:	12	
Firmware version:	1.026	<button>Update</button>
Reader Type:	RS500	1-1740 4515
Reader Firmware Version:	1.8.12.240	<button>Update</button>
<button>Reload</button>		

- Shows the information about the integrated Read-/Write head and the UHF module. Additionally, the firmware update of both units can be done here.

Details see → [Update of the Read-/Write head or the UHF module](#) .

9.11 Open Source license information

The licensed free software is of average quality and suitable for normal use on common personal computers. According to the present state of software technology no guarantee can be assumed for the correct operation or absence of computer viruses nor for the removal of any fault which may occur. In case of an incorrect program ifm electronic gmbh cannot be held liable for cost incurred at the customer (e.g. maintenance, repair or rectification of faults). The exclusion does not apply to damage for which liability is mandatory according to peremptory legal provisions. Copyright by ifm electronic gmbh, Essen. No copies may be made without the written permission of ifm electronic gmbh. This also goes for printing images or text for business use. We reserve the right to make technical alterations.
[license text](#)



The licensed free software information is displayed and can be downloaded from the device.

Extract of the license text file:

DTEXxX Open Source licenses information
ifm electronic GmbH

Sections included in this document:

List of licenses included in cJSON by package

List of copyrights included in cJSON by file

List of licenses included in newlib 1.14.0 (ARM syscalls) by package

List of copyrights included in newlib 1.14.0 (ARM syscalls) by file

List of licenses included in MD5 by package

List of copyrights included in MD5 by file

List of licenses included in linenoise by package

List of copyrights included in linenoise by file

List of licenses included in Lua 5.1.4 by package

List of copyrights included in Lua 5.1.4 by file

List of licenses included in pthread by package

List of copyrights included in pthread by file

List of licenses included in STM32 ADC/USART by package

List of copyrights included in STM32 ADC/USART by file

List of licenses included in the STM32 HAL by package

List of copyrights included in the STM32 HAL by file

List of licenses included in web documents by package

List of copyrights included in web documents by file

[Complete licenses Text](#)

Portions of this document were generated with ScanCode, see the corresponding [acknowledge](#) for more information

List of licenses included in cJSON by package

Back to top

/cjson/JSON, MIT License granted by MIT

/cjson/JSON/cJSON/README, MIT License granted by MIT

/cjson/JSON/cJSON, MIT License granted by MIT

10 RFID IOT Core protocol

10.1 General information about the different RFID devices with IoT Core

The devices DTE105, DTE605, DTE805, DTE9x5 have different hardware versions so that the physical access to HF and UHF ID tags are possible. For this reason, there are also different IoT core services when accessing the data of the ID tags. The following description indicates which services are available for the individual devices.

10.1.1 Required firmware version of the device DTE105

Following, minimum firmware revisions are required.

Runtime firmware
V4.1. 7

10.1.2 Required firmware versions of the device DTE605

Following, minimum firmware revisions are required.

Runtime firmware	R/W head firmware	UHF firmware
V4.1. 6	V1. 031	1.8.12. 240

10.1.3 Required firmware versions of the devices DTE8x5, DTE9x5

Following, minimum firmware revisions are required.

Runtime firmware	R/W head firmware	UHF firmware
V4.1. 6	V1. 031	1.8.12. 240

10.2 IoT Core general informations

The IoT Core allows the user to address the devices from IT networks via a **RE**presentational **S**tate **T**ransfer **AP**plication Interface (REST API) and to integrate it into Internet-of-Things (IoT) applications.

A device description is stored in the devices and can be obtained via the command `gettree()`.

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.

10.2.1 HTTP protocol format

The devices with RFID IoT profile support only the non secure HTTP protocol.

The IoT Core command requests are sent over the HTTP methods "GET" and "POST".
The command and response itself are code in JSON format.



All data, that is not in JSON format, has to be coded in BASE64 format.

10.2.2 TCP Port of the RFID devices with IoT Core

The IoT Core of the devices with integrated RFID IoT profile can be accessed only over TCP port **8080** only.



The access to the integrated Webserver is possible over TCP port 80.

11 Access the ifm IoT Core

The user can access the ifm IoT Core of the RFID devices over Port 8080 via HTTP requests. The requests itself are send and received as JSON objects.

http://ip/name/data_point/service

Parameter	Description	Remark
ip	IP address of the IoT interface	IPV4 address of the target device e.g.: 192.168.0.79
Name	IoT Core Device name	UTF-8 string e.g. "dte105-0000034652" ⁽¹⁾
data_point	Data point which is to be accessed	UTF-8 string e.g. "processdata"
service	Service	UTF-8 string e.g. "getidentity"

(1) Can be obtained from the Webserver on Tab "info"

<http://192.168.0.79/awe/main.sht?target=info>

IoT Core device name = <Article number>-"<10 digit Serial number>

Example:

Hardware information

Article: DTE105
Hardware version: 5
Firmware version: E1.0.99
Serial number: 34652
Production date: 2021-08-23 15:25
MAC-address: 00:02:01:42:87:5C

IoT Core Device name:

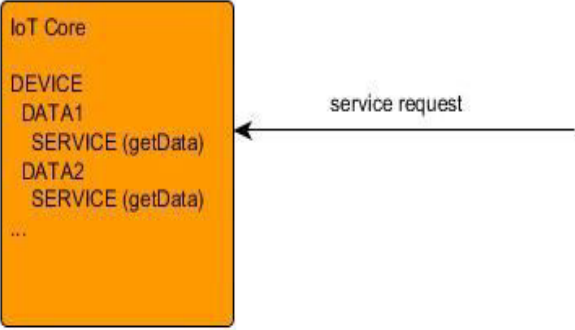
DTE105-**0000034652**

Note: The "Name" parameter can be skipped for the RFID IoT Core devices cause per device only one tree is supported.

11.1 JSON objects

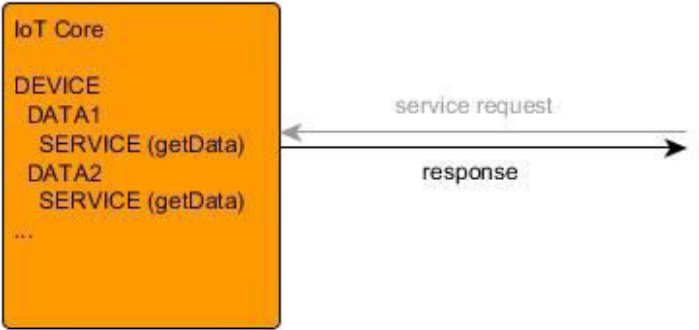
11.1.1 JSON requests

The ifm IoT Core interact with services. Each interaction is a "request" from the host over a URL with a service element of the device.

JSON request object	
<pre>{ "code" : "request", "cid" : number, "adr" : service_adr, "reply" : "reply_data", "data" : "serialized_data", "auth" : {authinfo} }</pre>	
Field explanation	
code	"request"
cid	The issuer of a call can give a number, that will be incorporated into the response. The correlation identifier is provided by the issuer of the request and is mirrored back to him in the response. allowed values: [0 ... 32768]
adr	The address of the service that should be called in the IoT Core device.
reply	Optional: The reply data of the response. If this field is used, the "reply_data" will be sent in the response.
data	Optional: Data payload for the call. This is specific to the service that is called. Getdata services does not need any parameters. Setdata services need additional parameters.
auth	Optional: Authentication information depending on authScheme (see getidentity service). In "standard" auth scheme, auth should contain a sub object "user" with based64 encoded value "administrator" = "YWRtaW5pc3RyYXRvcg==" plus base64 encoded password.

11.1.2 JSON Response

Every service will answer with a response from the device. This is the JSON object for the response:

JSON response object	
<pre>{ "cid" : number, "adr" : service_address, "reply" : "reply_data" "data" : "serialized_data", ... "code" : number(see http_codes) }</pre>	 The diagram shows an orange box labeled 'IoT Core' containing the text: 'DEVICE', 'DATA1', 'SERVICE (getData)', 'DATA2', 'SERVICE (getData)', and '...'. To the right of the box, a double-headed arrow is shown. The top half of the arrow points from the IoT Core box to the right and is labeled 'service request'. The bottom half of the arrow points from the right back to the IoT Core box and is labeled 'response'.
Field explanation	
cid	This is the very number that was given in the request. If an error happens so early that the cid can't be tracked by the system or no cid was given in the request, the cid = -1 will be returned.
adr	The address of the service that was called. For http GET request this field may be omitted.
reply	<u>Optional:</u> reply_data sent in the JSON request.
data	<u>Optional:</u> serialized data - in case the service has a data response
code	This is the http protocol response code. Code numbers greater or equal to 100 are valid response codes. See chapter -> IoT Core: HTTP status codes

11.2 HTTP request call

There are two methods to request a service from the device.

11.2.1 GET request

Using the GET method the user has read access to a data point. The syntax of the request to the IoT Core is:

Method: GET

URL:

`http://ip/data_point`

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

The syntax of the return of the IoT Core is:

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

Request element	Parameter	Description
cid	id	Correlation identifier for the assignment of request. In case of GET request, the value is always set to "-1"
data	resp_data	Value of the data point; depending on the data type of the data point
code	diag_code	Diagnostic code (See chapter -> IoT Core: HTTP status codes)

Example:

<http://192.168.0.79:8080/dte105-0000034652/getidentity>

or short form cause only one IoT Core is behind the URL

<http://192.168.0.79:8080/getidentity>

Response data:

```
{
  "cid":-1,
  "data":
    {
      "iot":
        {
          "name": "DTE105-0000034652",
          "version": "1.3",
          "catalogue":
            {
              "name": "ifm-std",
              "version": "1.2",
              "namespace": "JSON",
              "deviceclass": "device"
            },
          "security":
            {
              "securityMode": "disabled",
              "authScheme": "standard",
              "isPasswdSet": "FALSE"
            }
        },
      "code":200
    }
}
```

11.2.2 POST request

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

Method: POST

URL:

http://ip/data_point/service

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 Identifier for the assignment of request and response; ID freely assignable by the user in the range from 0...32767.
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 ⁽¹⁾	req_data	Data to be transferred to the IoT Core (e.g. new values). The syntax is depending on the service to send
auth ⁽²⁾	usr_id	user name (base64 encoded); default value: administrator
	password	password (base64 encoded)

⁽¹⁾ = optional; only required for services, that submit data to the IoT core (e. g. setdata)

⁽²⁾ = optional; only required, if security mode is activated

Response data:

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

Field	Parameter	Description
cid	id	Correlation Identifier for the assignment of request and response Device will respond with the same value as set in the request.
data	resp_data	Value of the data point; syntax depending on the service
code	diag_code	Diagnostic code (→ IoT Core: Diagnostic codes)

11.3 IoT general functions

The device itself has the type "device".

11.4 IoT Services

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
../getidentity	Reading device information
../querytree	Search device description for specific elements
../getsubscriberlist	Print a list of all active notification subscriptions
../getdatamulti	Reading several parameter values sequentially
../subscribe	Subscribe to an event
../unsubscribe	Unsubscribe from an event

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 device information

Task: Determine the device information .

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

11.4.1 First possibility, call of request over URL and POST method

URL:

<http://192.168.0.79:8080>

Method: POST

Request body data:

```
{
  "code":10,
  "cid":4711,
  "adr":"/deviceinfo/productcode/getdata"
}
```

Response data:

```
{
  "cid":-1,
  "data":{"value":"DTE105"},
  "code":200
}
```

11.4.2 Second possibility, direct call of request over URL

URL:

<http://192.168.0.79:8080/deviceinfo/productcode/getdata>

Response data:

```
{
  "cid":-1,
  "data":{"value":"DTE105"},
  "code":200
}
```

URL:

<http://192.168.0.79:8080/deviceinfo/serialnumber/getdata>

Response data:

```
{
  "cid":-1,
  "data":{"value":"0000034652"},
  "code":200
}
```

URL:

<http://192.168.0.79:8080/deviceinfo/swrevision/getdata>

Response data:

```
{
  "cid":-1,
  "data":{"value":"V4.1.6"},
  "code":200
}
```

Response data:

<http://192.168.0.79:8080/deviceinfo/hwrevision/getdata>

Response

```
{
  "cid":-1,
  "data":{"value":"00005"},
  "code":200
}
```

Request:

<http://192.168.0.79:8080/deviceinfo/swrevision/getdata>

Response

```
{
  "cid":-1,
  "data":{"value":"V4.1.6"},
  "code":200
}
```

11.4.3 First steps

First the HTTP server need to read the device description of the device.
With the received information the applicable services and data points can be accessed.

- Send the following POST request to the DEVICE:

URL:

<http://192.168.0.79:8080/gettree>

Method: POST

Request body data:

```
{
  "code": "request",
  "cid": 1,
  "adr": "gettree"
}
```

- > The device 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.

Example response for DTE105 device:

Method: POST

Response data:

```
{
  "cid": 0001,
  "data": {
    "identifier": "DTE105AA-0000032653",
    "type": "device",
    "subs": [
      {
        "identifier": "GetIdentity",
        "type": "service"
      },
      ...
      {
        "identifier": "ipconfig",
        "type": "structure",
        "profiles": [
          "network",
          "network_ethernet",
          "datablock",
          "parametergroup"
        ],
        ...
      },
      {
        "identifier": "setblock",
        "type": "service"
      },
      ...
    ]
  }
}
```

12 General IoT Core services

The following services can be used to get more information from the data elements of the device.

12.1 Get Identity of the device

With this service a device can be identified by the host.

Service available: All devices

Method: POST

URL:

<http://192.168.0.79:8080>

Request body data:

```
{
  "code": "request",
  "cid": 1234,
  "adr": "getidentity",
  "reply": "reply text "
}
```

Request element	Value	Remark
code	"request"	-
cid	<Number>	Correlation identifier. Can be set from 0...32767 e.g. 1234
adr	"<target_address>"	Service to be called e.g. getidentity
<i>reply</i>	"reply text"	Optional: Text to be send back in the response. If not needed, this element can be removed from the request.

Response data:

```
{
  "cid": 1234,
  "data": {
    "iot": {
      {
        "name": "DTE905AB-0000047712", "version": "1.3", "catalogue": {
          {
            "name": "ifm-std",
            "version": "1.2"
          }
        },
        "namespace": "JSON", "deviceclass": "device"
      }
    }
  },
  "adr": "XYZ",
  "code": 200
}
```

Response element	Value	Remark
cid	<Number>	Correlation identifier. Same as send in request.
data	":{"iot": {"name": "DTE905AB-0000047712", "version": "1.3", "catalogue": {"name": "ifm-std", "version": "1.2"}, "namespace": "JSON", "deviceclass": "device"}}	
<i>adr</i>	<string>	Optional. Text sent in element "reply" of the service request e.g. "reply text"
code	200	Response code of the IoT core See chapter -> IoT Core: HTTP status codes

12.2 Read diagnostics from the IO channel

With this service the diagnostics of the IO channels can be read.

Service available: All devices

This service is supported in all channel modes

Method: POST

URL:

<http://192.168.0.79:8080>

Request body data:

```
{
  "code": "request",
  "cid": 9000,
  "adr": "diagnostics/port[1]/list",
  "reply": "reply text"
}
```

Request element	Value	Remark
code	"request"	-
cid	<Number>	Correlation identifier. Can be set from 0...32767 e.g. 9000
adr	<target_address>	For DTE10x : N = 1...4 For DTE60x, DTE80x, DTE9Xx: N = 1...3 e.g.: "diagnostics/port[1]/list"
<i>reply</i>	<i>"reply text"</i>	Optional: Text to be send back in the response. If not needed, this element can be removed from the request.



If the error does not persist, it is deleted by reading it out.

Response data:

```
{
  "cid": 9000,
  "data": {
    "count": 1,
    "codes": ["F4FE9005"]
  },
  "adr": "reply text ",
  "code": 200
}
```

Response element	Value	Remark
cid	<Number>	Correlation identifier. Same as send in request. e.g. 9000
data	"count": N codes:["<error code>"]	Number of command error codes present Command error codes of the device: See chapter Error codes of the device
<i>adr</i>	<string>	Optional. Text sent in element "reply" of the service request e.g. "reply text"
code	200	Response code of the IoT core See chapter -> IoT Core: HTTP status codes

12.3 Subscribe to event

With this service an event can be subscribed by the host to get automatically a notification by the device if the data of the event has changed.

One event can be registered by different subscribers, but each need must use a unique "cid".



Before the TCP connection is closed the host must unsubscribe all active events !

The active subscribed events can be read out over the IoT core service -> [Get subscription information](#)



The subscription to an event is volatile and had to be renewed after each power on cycle.

Method: POST

URL:

<http://192.168.0.79:8080>

Request body data:

```
{
  "code": "request",
  "cid": "<Number>",
  "adr": "data/port[<N>]/<Event>/subscribe",
  "reply": "reply text",
  "data":
  {
    "callback": "http://<ip address>:<Port>/<Event>", "datatose": [ ]
  }
}
```



Request element	Value	Remark
code	"request"	-
cid	<Number>	Multiple subscriptions from one source are allowed, but each will need a unique "cid". CID must be stored for unsubscribe of event !
adr	"data/port[<N>]/<srv_element>/subscribe"	For DTE10x : N = 1...4 For DTE60x, DTE80x, DTE9Xx: N = 1 <srv_element>: Service element to subscribe to.
<i>reply</i>	"reply text"	Optional: Text to be send back in the response. If not needed, this element can be removed from the request.
data	{ "callback": "http://<ip address>: <Port>/ <Event>", "datatose": [] }	Data of the IoT Webserver ⁽¹⁾ : IP address : IP address. Port : IP port number. Event : Name of the event under which this is registered in the device. Can be set freely by the user. To unsubscribe the event from the device, this name must be taken. Optional: datatose : The data element addresses, whose values are sent with the event

⁽¹⁾ Please check, that the IoT Webserver is available under the IP address and the IP port is not blocked by the firewall.

Response data:

```
{  
  "cid":<Number>,  
  "adr":"reply text",  
  "code":200  
}
```

Response element	Value	Remark
cid	<Number>	Correlation identifier. Same as send in request
<i>adr</i>	<string>	Optional. Text sent in element "reply" of the service request e.g. "reply text"
code	200	Response code of the IoT core See chapter -> IoT Core: HTTP status codes , e.g. 200

Example:

See [Receive C/Qi from IO port automatically](#)

12.4 Unsubscribe from event

With this service an already registered event via subscribe service can be unsubscribed by the host from the device.



Before the TCP connection is closed the host must unsubscribe all active events !

The active subscribed events can be read out over the IoT core service -> [Get subscription information](#)

Method: POST

URL:

<http://192.168.0.79:8080>

Request body data:

```
{
  "code": "request",
  "cid": "<Number>",
  "adr": "data/port[<N>]/<Event>/unsubscribe ",
  "reply": "reply text",
  "data":
  {
    "callback": "http://<ip address>:<Port>/<Eventhandler>",
  }
}
```

Request element	Value	Remark
code	"request"	-
cid	<Number>	CID <u>must</u> match to the cid of the subscribe event otherwise the unsubscribe will not work !
adr	"data/port[<N>]/<Event>/unsubscribe"	For DTE10x : N = 1...4 For DTE60x, DTE80x, DTE9Xx: N = 1 <Event> : Event name to unsubscribe from.
reply	"reply text"	Optional: Text to be send back in the response. If not needed, this element can be removed from the request.
data	{ "callback": "http://<ip address> : <Port>/ <Eventhandler>" }	Data of the IoT Webserver ⁽¹⁾ : IP address : IP address. Port : IP port number. Eventhandler : Name of the eventhandler to unsubscribe from.

⁽¹⁾ Please check, that the IoT Webserver is available under the IP address and the IP port is not blocked by the firewall.

Response data:

```
{
  "cid": 1001,
  "adr": "mydevice/remote/someotherdevice/someevent/unsubscribe",
  "reply": "reply text",
  "code": 200
}
```

Response element	Value	Remark
cid	<Number>	Correlation identifier. Same as send in request.
<i>adr</i>	<string>	Optional. Text sent in element "reply" of the service request e.g. "reply text"
code	200	Response code of the IoT core See chapter -> IoT Core: HTTP status codes

Example:

See [Receive C/Qi from IO port automatically](#)

12.5 Get all subscriptions of the device

With this service the state of all registered events via subscribe service can be read by the host from the device.

Method: POST

URL: http://192.168.0.79:8080

Request body data:

```
{
  "code": "request",
  "cid": 9999,
  "adr": "getsubscriberlist",
  "reply": "reply text",
}
```

Request element	Value	Remark
code	"request"	-
cid	<Number>	CID <u>must</u> match to the cid of the subscribe event e.g. 9999
adr	"getsubscriberlist"	-
<i>reply</i>	<i>"reply text"</i>	Optional: Text to be send back in the response. If not needed, this element can be removed from the request.

(1) Please check, that the IoT Webserver is available under the IP address and the port is not blocked by a firewall.

Response data:

```
{
  "cid": 9999,
  "data": {
    "adr": "DTE805AB-0000001426/data/powersupply/datachanged",
    "cid": 1012,
    "callbackurl": "http://192.168.0.79:34003/dte-powersupply- state",
    "duration": "lifetime",
    "adr": "reply text",
    "code": 200
  }
}
```

Response element	Value	Remark
cid	<Number>	Correlation identifier. Same as send in request. e.g. 9999
data	{ "adr": "DTE805A...001426/data/powersupply/datachanged", "cid": 1012, "callbackurl": "http://192.168.0.79:34003/dte-powersupply-state", "duration": "lifetime" }	adr: Service cid of the subribed event callback URL duration of the event ⁽¹⁾
<i>adr</i>	<string>	Optional. Text sent in element "reply" of theservice request e.g. "reply text"
code	200	Response code of the IoT core See chapter -> IoT Core: HTTP status codes

(1)

Lifetime : Value stored only till next power up of IoT Core

Once : Value used only once, if request is [subscribe](#), Process [UnSubscribe](#) after one notification

12.6 Connect the IoT Webserver with the device

Start a HTTP Webserver server on the host.
Open and listen at free selectable TCP port.

Following message is sent from the device to the IoT Webserver via TCP protocol.

TCP payload data chunk 1:

Host: 192.168.0.61:12345
Content-Type: application/json
Transfer-Encoding: chunked
Connection: keep-alive

TCP payload data chunk 2:

```
{
  "code": "event",
  "cid": 1000,
  "adr": "/io1-epcs-changed",
  "data":
  {
    "eventno": "25652",
    "srcurl": "DTE905AB- 0000047712/data/port[1]/rfid/id_tag_list/epcs/datachanged",
    "payload":
    {
      "/data/port[1]/rfid/id_tag_list/epcs":
      {
        "code": "20 0",
        "data": "e008010c38325124"
      }
    }
  }
}
```

Response element	Value	Remark
code	"event"	-
cid	<CorrID>	Correlation identifier. Same as send in request e.g. 1000
adr	"<target_address>"	e.g.: /io1-epcs-changed
data	<pre>{ "eventno": "<number>", "srcurl": "<service_address>", "payload": { "<service>", { "code": "<response code>", "data": "<UID>" } } }</pre>	e.g. "25652" e.g. "DTE905AB-0000047712/data/port[1]/rfid/id_tag_list/epcs/datachanged" e.g. "/data/port[1]/rfid/id_tag_list/epcs" Response code of the IoT core. e.g. 200 The 'data' string element is coded in ASCII hex data. e.g. "e008010c38325124"
code	200	Response code of the IoT core See chapter -> IoT Core: HTTP status codes

TCP payload data chunk 3:

0

13 RFID IoT Core profile

In addition to the generally applicable IoT core data points, RFID devices also have special ones that allow to communicate with the LF / HF / UHF ID tags.

13.1 Configure IO-Ports

Service available: DTE105, DTE605, DTE805, DTE9x5

This service is supported in the channel mode "RWH"

Configure the IO-Ports according to the application requirements

Method: POST

URL:

<http://192.168.0.79:8080>

Request body data:

```
{
  "code": "request", "cid": 123, "adr": "/config/setblock",
  "data": {
    "datatoset": {
      "dte/temperature_unit": 0,
      "port[1]/mode": 0,
      "port[1]/data_hold_time": 20,
      "port[2]/mode": 1,
      "port[2]/data_hold_time": 20,
      "port[3]/mode": 2,
      "port[3]/data_hold_time": 20
    }
  }
}
```

Note: The device is configured on IO-1 as RWH, IO-2 as Input, IO-3 as output.

Request element	Value	Remark
code	"request"	
cid	<CorrID>	Correlation identifier. Can be set from 0...32767
adr	"/config/setblock"	
data	"datatoset"	Element values of the device to be set

Request element "datatset"	Value	Remark
dte/temperature_unit	0 = [°C] 1 = [F]	For DTE05 valid
port	1 = IO-1 2 = IO-2 3 = IO-3 4 = IO-4	For DTE05 valid For DTE05 valid For DTE05 valid Only for DTE105 valid
mode	0 = RFID 1 = INPUT 2 = OUTPUT 3 = OFF	For DTE605,DTE805,DTE9x5 fix set on IO-1
data_hold_time	nn	[ms] nn = { 0, 20, 40, 60, 80, 100, 200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000}
/id_tag_config/block_size	BS	[bytes] Only for DTE105 und DTE605 valid HF ID tags: {4,8,16,32,64,128}
/id_tag_config/number_of_blocks	NoB	[bytes] Only for DTE105 und DTE605 valid HF ID tags: {1...256}
/overload_detection_enable	0 = off 1 = on	Only for DTE105 valid on IO-1 ... IO-4
/overcurrent_detection_enable	0 = off 1 = on	Only for DTE105 valid on IO-1 ... IO-4
/highcurrent_detection_enable	0 = off 1 = on	Only for DTE105 valid on IO-3 ... IO-4

Response data:

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

Response element	Value	Remark
cid	100	-
adr	"/"	The 'data' string element is coded in Base64.
code	200	Response code of the IoT core ⁽²⁾ See chapter -> IoT Core: HTTP status codes

Parameter settings:

Request element	Value	Remark
port	1 = IO-1 2 = IO-2 3 = IO-3 4 = IO-4	For DTE05 valid For DTE05 valid For DTE05 valid Only for DTE105 valid
mode	0 = RFID 1 = INPUT 2 = OUTPUT 3 = OFF	
data_hold_time	0 ... 65525	[ms]
/id_tag_config/block_size	4 ... 128 2	[bytes] Only for HF ID tags valid [bytes] Fix for UHF ID tags
/id_tag_config/number_of_blocks	0...256 0	[bytes] Only for HF ID tags valid [bytes] Fix for UHF ID tags

13.2 Access data of the IO ports

The following services allow the access to the IO ports.

13.2.1 Read input C/Qi of the IO channel

Service available: DTE105, DTE605, DTE805, DTE9x5

This service is supported in the channel mode "Input".

Method: POST

URL:

<http://192.168.0.79:8080>

```
{  
  "code": "request",  
  "cid": 2002,  
  "adr": "data/port[2]/input/cqi/getdata",  
  "reply": "reply text"  
}
```

Request element	Value	Remark
code	"request"	-
cid	<CorrID>	Correlation identifier. Can be set from 0...32767 e.g. 2002
adr	data/port[N]/input/iq/getdata	For DTE10x : N = 1...4 For DTE60x, DTE80x, DTE9Xx: N = 2,3
reply	"<string>"	Optional: Text to be send back in the response. If not needed, this element can be removed from the request. e.g. "reply text"

Response data:

```
{  
  "cid": 2002,  
  "data":{"value":1},  
  "adr":"reply text",  
  "code":200  
}
```

Response element	Value	Remark
cid	<CorrID>	Correlation identifier. Same as send in request e.g. 2002
data	{"value":<state>}	State of the channel input C/Qi <0> = off (0V) <1> = on (24V) e.g. 1

<i>adr</i>	<string>	Optional. Text sent in element "reply" of the service request e.g. "reply text"
code	<status code>	Response code of the IoT core See chapter -> IoT Core: Status codes e.g. 200

13.2.2 Receive C/Qi from IO port automatically

Method: POST

URL:

<http://192.168.0.79:8080>

Request body data:

```
{
  "code": "request",
  "cid": 2012,
  "adr": "data/port3]/input/cqi/datachanged/subscribe",
  "reply": "reply text",
  "data":
  {
    "callback": http://192.168.1.1:34003/dte-io2-cqi-changed
  }
}
```

Request element	Value	Remark
code	"request"	-
cid	<CorrID>	Multiple subscriptions from one source are allowed, but each will need a unique "cid". e.g. 2012
adr	"data/port[N]/input/cqi/datachanged/subscribe"	For DTE10x : N = 1...4 For DTE60x, DTE80x, DTE9Xx: N = 1
<i>reply</i>	"<string>"	Optional: Text to be send back in the response. If not needed, this element can be removed from the request. e.g. "reply text"
data	{ "callback": <a href="http://<ip address>:<Port>/<Event>">http://<ip address>:<Port>/<Event> }	Data of the IoT Webserver ⁽¹⁾ : IP address : IP address of the subscriber Port : Port number of the subscriber Event : Name of the event. Can be set freely by the User, e.g. "http://192.168.1.1:34003/dte-io2-cqi-changed"

(1) Please check, that the IoT Webserver is available under the IP address and the ip.port is not blocked by the firewall.

Response data:

```
{
  "cid": 2012,
  "adr": "reply text",
  "code": 200
}
```

Response element	Value	Remark
cid	<CorrID>	Correlation identifier. Same as send in request, e.g. 2012
<i>adr</i>	<string>	Optional. Text sent in element "reply" of the service request, e.g. "reply text"
code	<status code>	Response code of the IoT core See chapter -> IoT Core: HTTP status codes e.g. 200

Data sent from the device to the IoT Webserver:

```
{
  "code": "event",
  "cid": 2012,
  "adr": "/io3-cqi-changed",
  "data": {
    "eventno": "45", "srcurl": "DTE105AA- 0000032653/data/port[3]/input/cqi/datachanged",
    "payload": {
      "/data/port[3]/input/cqi": {
        "code": 200,
        "data": 1
      }
    }
  }
}
```

Response element	Value	Remark
code	"event"	Type of response, here "event"
cid	<CorrID>	Correlation identifier. Same as send in subscribe request e.g. 2012
adr	"<event_name>"	Name of the event under which this is registered in the device. e.g. "/io3-cqi-changed"
data	{ "eventno": "<number>", "srcurl": "<service_address>", "payload": { "<service>", { "code": <response code>, "data": <state>" } } }	Number of the event send from the device e.g. "45" Address of the service send from the device e.g. "DTE105AA-.../input/cqi/datachanged" Service name, e.g. "/data/port[3]/input/cqi" Response code of the IoT core See chapter -> IoT Core: HTTP status codes e.g. 200 State of the channel input C/Qi 0 = off (0V) 1 = on (24V)

13.2.3 Read input I/Q of the IO channel

Service available: DTE105, DTE605, DTE805, DTE9x5

This service is supported in the channel mode "Input" and "Output".

Method: POST

URL:

<http://192.168.0.79:8080>

```
{
  "code": "request",
  "cid": 2001,
  "adr": "data/port[2]/input/iq/getdata",
  "reply": "reply text"
}
```

Request element	Value	Remark
code	"request"	-
cid	<CorrID>	Correlation identifier. Can be set from 0...32767 e.g. 2001
adr	"data/port[N]/input/iq/getdata"	For DTE10x : N = 1...4 For DTE60x, DTE80x, DTE9Xx: N = 2,3 e.g. 2
<i>reply</i>	"<string>"	Optional: Text to be send back in the response. If not needed, this element can be removed from the request. e.g. "reply text"

Response data:

```
{
  "cid":2001,
  "data":{"value":1},
  "adr":"reply text",
  "code":200
}
```

Response element	Value	Remark
cid	<CorrID>	Correlation identifier. Same as send in request, e.g. 2001
data	{"value":<state>}	State of the channel input I/Q <0> = off (0V) <1> = on (24V) e.g. 1
<i>adr</i>	<string>	Optional. Text sent in element "reply" of the servicerequest, e.g. "reply text"
code	<status code>	Response code of the IoT core See chapter -> IoT Core: HTTP status codes e.g. 200

13.2.4 Write output C/Qo of the IO channel

Service available: DTE105, DTE605, DTE805, DTE9x5

This service is supported in the channel mode "Output" .

Method: POST

URL:

<http://192.168.0.79:8080>

```
{
  "code": "request",
  "cid": 4711,
  "adr": "/data/port[4]/output/cqo/setdata",
  "reply": "reply text",
  "data": {"newvalue": 1}
}
```

Request element	Value	Remark
code	"request"	-
cid	<CorrID>	Correlation identifier. Can be set from 0...32767
adr	"/data/port[N]/output/cqo/setdata"	For DTE10x : N =1...4 For DTE60x, DTE80x, DTE9Xx: N = 2,3
<i>reply</i>	"<string>"	Optional: Text to be send back in the response. If not needed, this element can be removed from the request, e.g. "reply text"
data	{"newvalue": <val>}	State to set for channel output C/Qo <0> = off (0V) <1> = on (24V)

Note, that the IO port

Response data:

```
{
  "cid":4711,
  "adr": "reply text",
  "code":200
}
```

Response element	Value	Remark
cid	4711	Correlation identifier. Same as send in request
<i>adr</i>	<string>	Optional. Text sent in element "reply" of the service request, e.g. "reply text"
code	<status code>	Response code of the IoT core, see chapter -> IoT Core: HTTP status codes , e.g. "200"

13.3 Access data of the HF ID tag

The following services allow the access to the HF ID tags.

13.3.1 Read UID of HF ID tags

Service available: DTE105, DTE605

This service is supported in the channel mode "RWH"

Method: POST

URL:

<http://192.168.0.79:8080>

Request body data:

```
{
  "code": "request",
  "cid": 1001,
  "adr": "data/port[1]/rfid/id_tag/uid/getdata",
  "reply": "reply text"
}
```

Request element	Value	Remark
code	"request"	-
cid	<CorrID>	Correlation identifier. Can be set from 0...32767 e.g. 1001
adr	"data/port[N]/rfid/id_tag/uid/getdata"	For DTE10x : N = 1...4 For DTE60x, DTE80x, DTE9Xx: N = 1 e.g. 1
<i>reply</i>	"<string>"	Optional: Text to be send back in the response. If not needed, this element can be removed from the request. e.g. "reply text"

Response data:

```
{
  "cid":1001,
  "data":{"value":"e008010c38325124"},
  "adr":"reply text",
  "code":200
}
```

Response element	Value	Remark
cid	<CorrID>	Correlation identifier. Same as send in request e.g. 1001
data	{"value": "<uid>"}	value: UID coded in HEX, 2 characters per HEX number e.g. e008010c38325124
<i>adr</i>	"reply text"	Optional: Text sent in element "reply" of the service request
code	<status code>	Response code of the IoT core See chapter -> IoT Core: HTTP status codes e.g. 200

Method: GET

URL:

[http://192.168.0.79:8080//data/port\[1\]/rfid/id_tag/uid/getdata](http://192.168.0.79:8080//data/port[1]/rfid/id_tag/uid/getdata)

Response data:

```
{
  "cid":-1,
  "data":{"value":"e008010c38325124"},
  "code":200
}
```

Response element	Value	Remark
cid	<CorrID>	Correlation identifier. Same as send in request e.g. -1
data	{"value":"e008010c38325124"}	value: UID coded in HEX, 2 characters per HEX number, e.g. "e008010c38325124"
code	200	Response code of the IoT core See chapter -> IoT Core: HTTP status codes

13.3.2 Receive UID from HF ID tag automatically

Service available: DTE105, DTE605

This service is supported in the channel mode "RWH"

Method: POST

URL:

<http://192.168.0.79:8080>

Request body data:

```
{
  "code": "request",
  "cid": 1010,
  "adr": "data/port[1]/rfid/id_tag/uid/datachanged/subscribe",
  "reply": "reply text",
  "data": {"callback": "http://192.168.1.1:34003/io1-uid-nw"}
}
```

Request element	Value	Remark
code	"request"	-
cid	<CorrID>	Correlation identifier. Multiple subscriptions from one source are allowed, but each will need a unique "cid". e.g. 1010
adr	"data/port[N]/rfid/id_tag/uid/datachanged/subscribe"	For DTE10x : N = 1...4 For DTE60x, DTE80x, DTE9Xx: N = 1, e.g. 1
<i>reply</i>	"<string>"	Optional: Text to be send back in the response. If not needed, this element can be removed from the request. e.g. "reply text"
data	{ "callback": "http://<ip address>:<Port>/<Event>" }	Data of the IoT Webserver ⁽¹⁾ : IP address: IP address e.g. 192.168.1.1 Port: Port number. e.g. 34003 Event: Name of the event. Can be set freely by the User, e.g. io1-uid-nw

⁽¹⁾ Please check, that the IoT Webserver is available under the IP address and the ip port is not blocked by the firewall.

Response data:

```
{
  "cid": 1010,
  "adr": "reply text",
  "code": 200
}
```

Response element	Value	Remark
cid	<CorrID>	Correlation identifier. Same as send in request, e.g. 1010
<i>adr</i>	< <i>string</i> >	Optional. Text sent in element "reply" of the service request e.g. "reply text"
code	<status code>	Response code of the IoT core See chapter -> IoT Core: HTTP status codes e.g. 200

Data sent from the device to the IoT Webserver:

```
{
  "code": "event",
  "cid": 1010,
  "adr": "/io1-uid-nw",
  "data": {
    "eventno": "17579",
    "srcurl": "DTE105AA-0000032653/data/port[1]/rfid/id_tag/UID/datachanged",
    "payload": {
      "/data/port[1]/rfid/id_tag/UID": {
        "code": 200,
        "data": "e008010c38325124"
      }
    }
  }
}
```

Response element	Value	Remark
code	"event"	Type of response, here "event"
cid	<CorrID>	Correlation identifier. Same as send in subscribe request e.g. 1010
adr	"<event_name>"	Name of the event under which this is registered in the device. e.g. "/io1-uid-nw"
data	{ "eventno": "<number>", "srcurl": "<service_address>", "payload": { "<service>", { "code": "<response code>", "data": "<UID>" } } }	Number of the event send from the device e.g. "17579" Address of the service send from the device e.g. "DTE105AA-... rfid/id_tag/UID/datachanged" Service name, e.g. "/data/port[1]/rfid/id_tag/UID" Response code of the IoT core See chapter -> IoT Core: HTTP status codes e.g. 200 UID coded in HEX, 2 characters per HEX number e.g. "e008010c38325124"

13.3.3 Read User memory of HF ID tags

Service available: DTE105, DTE605

This service is supported in the channel mode "RWH"

Method: POST

URL:

<http://192.168.0.79:8080>

Request body data:

```
{
  "code": "request",
  "cid": 1002,
  "adr": "data/port[1]/rfid/id_tag/user_data/getblobdata",
  "reply": "reply text",
  "data": { "pos": 0, "length": 16 }
}
```

Request element	Value	Remark
code	"request"	-
cid	<CorrID>	Correlation identifier. Can be set from 0...32767, e.g. 1002
adr	"data/port[N]/rfid/id_tag/user_data/getblobdata"	For DTE10x : N = 1...4 For DTE60x, DTE80x, DTE9Xx: N = 1, e.g. 1
<i>reply</i>	"<string>"	Optional: Text to be send back in the response. If not needed, this element can be removed from the request, e.g. "reply text"
data	"pos": <offset> "length": <data_length>	Offset position in buffer [bytes] : 0...65535, e.g. 0 Length of data to be read [bytes] : 1...65534, e.g. 16

Note: getblobdata = "**Get Binary Large Object Data**"

Response data:

```
{
  "cid": 1002,
  "data": { "data": "UGxhbnQtQTFfMDEyMzQ1Ng==" },
  "code": 200
}
```

Response element	Value	Remark
cid	<CorrID>	Correlation identifier. Same as send in request. e.g. 1002
data	"data": "<string64>" ⁽¹⁾ or "error": "6469616700"	The 'data' string element is Base64 encoded. e.g. "UGxhbnQtQTFfMDEyMzQ1Ng==" or Message "Err"
<i>adr</i>	<string>	Optional. Text sent in element "reply" of the service request e.g. "reply text"
code	<status code>	Response code of the IoT core See chapter -> IoT Core: HTTP status codes e.g. 200

(1) Data on the ID tag:
Base64: UGxhbnQtQTFfMDEyMzQ1Ng==
ASCII: Plant-A1_0123456
HEX: 506C616E742D41315F30313233343536

13.3.4 Write User memory of HF ID tags

Service available: DTE105, DTE605

This service is supported in the channel mode "RWH"

Method: POST

URL:

<http://192.168.0.79:8080>

Request body data:

```
{
  "code": "request",
  "cid": 1010,
  "adr": "data/port[1]/rfid/id_tag/user_data/setblobdata",
  "reply": "reply text",
  "data": {"pos": 0, "data": "QmFzZTY0IGNvZGVkIHh0cmlyZW==" }
}
```

Request element	Value	Remark
code	"request"	-
cid	<CorrID>	Correlation identifier. Can be set from 0...32767. e.g. 1010
adr	"data/port[N]/rfid/id_tag/user_data/setblobdata"	For DTE10x : N = 1...4 For DTE60x, DTE80x, DTE9Xx: N = 1
<i>reply</i>	"<string>"	Optional: Text to be send back in the response. If not needed, this element can be removed from the request. e.g. "reply text"
data	"pos": <Offset> "data": "<string64>" ⁽¹⁾	Offset position in User memory of the ID tag [bytes] : 0...65535 tag, e.g. 0 The 'data' string element must be coded in Base64. The length of the string is calculated automatically.

Note: setblobdata = "**Set Binary LargeObjectData**"

⁽¹⁾ Example of data to be written to the ID tag's user memory

Base64: QmFzZTY0IGNvZGVkIHh0cmlyZW==

ASCII: Base64 encoded string

HEX: 42617365363420636F64656420737472696E67

Response data:

```
{
  "cid":1010,
  "code":200
}
```

Response element	Value	Remark
cid	<CorrID>	Correlation identifier. Same as send in request. e.g. 1010
code	<status code>	Response code of the IoT core See chapter -> IoT Core: HTTP status codes e.g. 200

13.4 Access data of the UHF ID tag

The following services allow the access to the UHF ID tags.

13.4.1 Read EPC data of UHF ID tags

Service available: DTE805, DTE9x5

This service is supported in the channel mode "RWH"

Method: POST

URL:

<http://192.168.0.79:8080>

Request body data:

```
{
  "code": "request",
  "cid": 3001,
  "adr": "data/port[1]/rfid/id_tag_list/epcs/getdata",
  "reply": "reply text"
}
```

Request element	Value	Remark
code	"request"	-
cid	<CorrID>	Correlation identifier. Can be set from 0...32767 e.g. 3001
adr	"data/port[N]/rfid/id_tag_list/epcs/getdata" "	For DTE10x : N = 1...4 For DTE60x, DTE80x, DTE9Xx: N = 1 e.g. 1
<i>reply</i>	"<string>"	Optional: Text to be send back in the response. If not needed, this element can be removed from the request, e.g. "reply text"

Response data:

```
{
  "cid":3001,
  "data":{"value":["\e008010c38325124\","\e008010c38325125\"]},
  "adr":"reply text",
  "code":200
}
```

Response element	Value	Remark
cid	<CorrID>	Correlation identifier. Same as send in request. e.g. 3001
data	{"value":["<string1>,<string2>,...,<stringN>"]}	The string elements are coded as 2 characters in ASCII hex data. e.g. "\e008010c38325124\","\e008010c38325125\"
<i>adr</i>	"reply text"	Optional: Text sent in element "reply" of the servicerequest, e.g. "reply text"
code	200	Response code of the IoT core, see chapter -> IoT Core: HTTP status codes , e.g. 200

Note: <string> elements are separated by \ " delimiters to show the elements are different string objects.

Method: GET

URL:

[http://192.168.0.79:8080//data/port\[1\]/rfid/id_tag_list/epcs/getdata](http://192.168.0.79:8080//data/port[1]/rfid/id_tag_list/epcs/getdata)

Response data:

```
{
  "cid":-1,
  "data":{"value":["eb80b0443c00001a"]},
  "code":200
}
```

13.4.2 Receive EPC data of UHF ID tags automatically

Service available: DTE805, DTE9x5

This service is supported in the channel mode "RWH"

Method: POST

URL:

<http://192.168.0.79:8080>

Request body data:

```
{
  "code": "request",
  "cid": 3010,
  "adr": "data/port[1]/rfid/id_tag_list/epcs/datachanged/subscribe",
  "reply": "reply text",
  "data": {"callback": "http://192.168.1.1:34003/io1-epcs-changed"}
}
```

Request element	Value	Remark
code	"request"	-
cid	<CorrID>	Multiple subscriptions from one source are allowed, but each will need a unique "cid". e.g. 3010
adr	"data/port[N]/rfid/id_tag_list/epcs/datachanged/subscribe"	For DTE10x : N = 1...4 For DTE60x, DTE80x, DTE9Xx: N = 1 e.g. N=1
<i>reply</i>	"<string>"	Optional: Text to be send back in the response. If not needed, this element can be removed from the request. e.g. "reply text"
data	{ "callback": "http://<ip address>:<Port>/<Event>" }	Data of the IoT Webserver ⁽¹⁾ : IP address: IP address. Port: Port number. Event: Name of the event. Can be set freely by the User. e.g. 192.168.1.1:34003/ io1-epcs-changed

⁽¹⁾ Please check, that the IoT Webserver is available under the IP address and the IP port is not blocked by the firewall.

Response data:

```
{
  "cid": 3010,
  "adr": "reply text",
  "code": 200
}
```

Response element	Value	Remark
cid	<CorrID>	Correlation identifier. Same as send in request e.g. 3010
adr	"<UTF-8 string>"	Text sent in element "reply" of the service request e.g. "reply text"
code	200	Response code of the IoT core See chapter -> IoT Core: HTTP status codes e.g. 200

Received data at the IoT Webserver

```
{
  "code": "event",
  "cid": 1010,
  "adr": "/io1-epcs-changed",
  "data": {
    {
      "eventno": "28",
      "srcurl": "DTE905AB- 0000047712/data/port[1]/rfid/id_tag_list/epcs/datachanged",
      "payload": {
        {
          "/data/port[1]/rfid/id_tag_list/epcs": {
            {
              "code": 200,
              "data": ["e200408071110000000003018305855"]
            }
          }
        }
      }
    }
  }
}
```

Response element	Value	Remark
code	"event"	-
cid	<CorrID>	Correlation identifier. Same as send in request e.g. 1010
adr	"<event_name>"	Name of the event under which this is registered in the device. e.g. "/ io1-epcs-changed "
data	{"value": "<epc_data>"}	value: EPC data coded with 2 characters per HEX number. e.g. e008010c38325124
<i>adr</i>	<i>"reply text"</i>	Optional: Text sent in element "reply" of the service request
code	200	Response code of the IoT core See chapter -> IoT Core: HTTP status codes e.g. 200

13.4.3 Read memory bank of the UHF ID tags tags

Service available: DTE805, DTE9x5

This service is supported in the channel mode "RWH"

Method: POST

URL:

<http://192.168.0.79:8080>

Request body data:

```
{
  "code": "request",
  "cid": 3002,
  "adr": "data/port[1]/rfid/id_tag/user_data/read", "reply": "reply text",
  "data": {
    "epc": "e200408071110000000003018305855",
    "bank": 3,
    "pos": 0,
    "length": 16
  }
}
```

Request element	Value	Remark
code	"request"	-
cid	<CorrID>	Correlation identifier. Can be set from 0...32767 e.g. 3002
adr	"data/port[N]/rfid/id_tag/user_data/read"	For DTE10x : N = 1...4 For DTE60x, DTE80x, DTE9Xx: N = 1
<i>reply</i>	"<string>"	Optional: Text to be send back in the response. If not needed, this element can be removed from the request. e.g. "reply text"
data	{ "epc": "<epc data>" "bank": <no> "pos": <offset> "length": <data_length> }	The epc data are coded as 2 characters in ASCII hex data. Bank: 1 - EPC bank 2 - TID bank 3 - User area Offset position in buffer [bytes] : 0...65535 Length of data to be read [bytes] : 1...65534

Response data:

```
{
  "cid":3002,
  "data":{"user":"506c616e742d41315f30313233343536"},
  "adr":"reply text",
  "code":200
}
```

Response element	Value	Remark
cid	<CorrID>	Correlation identifier. Same as send in request. e.g. 3002
data	"data": "<string1>" ⁽¹⁾ or "error": "6469616700"	The string elements are coded as 2 characters in ASCII hex data. e.g. 506C616E742D41315F30313233343536
adr	"<string>"	Text sent in element "reply" of the service request e.g. "reply text"
code	200 or 503	Response code of the IoT core See chapter -> IoT Core: HTTP status codes e.g. 200 or 205

- ⁽¹⁾ Data on the ID tag:
HEX: 506C616E742D41315F30313233343536,
ASCII: Plant-A1_0123456

13.4.4 Write User memory of the UHF ID tags

Service available: DTE805, DTE9x5

This service is supported in the channel mode "RWH"

Method: POST

URL:

<http://192.168.0.79:8080>

Request body data:

```
{
  "code": int/string,
  "cid": int,
  "adr": "/data/port[{1..n}]/rfid/id_tag/user_data/write",
  "data":
  {
    "epc": string,
    "bank": int,
    "pos": int,
    "data": string
  }
}
```

Request element	Value	Remark
code	"request"	-
cid	<CorrID>	Correlation identifier. Can be set from 0...32767 e.g. 3010
adr	"/data/port[{1..n}]/rfid/id_tag/user_data/write"	For DTE10x : N = 1...4 For DTE60x, DTE80x, DTE9Xx: N = 1
data	"epc": string, "bank": int, "pos": int, "data": string	epc : epc code of transponder to write data on bank : bank to write data, can be 0 (reserved), 1 (EPC bank), 2 (TID bank), 3 (user area) pos : offset to write to data : hexadecimal string of data

Response data:

```
{
  "cid":3010,
  "code":200
}
```

Response element	Value	Remark
cid	<CorrID>	Correlation identifier. Same as send in request. e.g. 3010
code	200	Response code of the IoT core See chapter -> IoT Core: HTTP status codes e.g. 200

13.4.5 Write EPC data of the UHF ID tags

Service available: DTE805, DTE9x5

This service is supported in the channel mode "RWH"

Method: POST

URL:

<http://192.168.0.79:8080>

Request body data:

```
{
  "code": int/string,
  "cid": int,
  "adr": "/data/port[{1..n}]/rfid/id_tag/EPC/setepc",
  "data":
  {
    "old": string,
    "new": string
  }
}
```

Request element	Value	Remark
code	"request"	-
cid	<CorrID>	Correlation identifier. Can be set from 0...32767 e.g. 3010
adr	"/data/port[{1..n}]/rfid/id_tag/EPC/setepc",	For DTE10x : N = 1...4 For DTE60x, DTE80x, DTE9Xx: N = 1
data	"old": string, "new": string	old : epc code of given transponder new : new epc code epc needs to be in hexadecimal format, e.g. E00401002079BA78

Response data:

```
{
  "cid":3010,
  "code":200
}
```

Response element	Value	Remark
cid	<CorrID>	Correlation identifier. Same as send in request. e.g. 3010
code	200	Response code of the IoT core See chapter -> IoT Core: HTTP status codes e.g. 200

14 IoT Core HTTP status codes

The http status code only shows whether the IoT Core command request was correct, not whether the execution of the service was successful.

Code	Text	Description
200	OK	The service could be process correctly
400	Bad request: <Reason>	The service request is invalid or malformed
401	Access denied: <Address>	Authentication is required and has failed or has not been provided.
403	Forbidden: <Reason>	The service request is forbidden
404	Element not found: <Address>	The requested service does not exist
413	Request payload too large	The payload of the request is too large
414	Request-URL too long	The URL of the request is too large
422	Invalid payload. Invalid <Parameter>: <Value>	The provided data/payload is invalid
423	The service is busy	The service is locked, disabled or busy at the moment
424	Failed dependency: <Reason> The service is currently not allowed. Example:	Failed dependency: first call <Service>
429	Too many requests	Too many requests
500 ⁽¹⁾	Internal error	An error condition, that is not considered and handled by IoTCore and therefore raised by the underlying runtime
501	Not implemented	The service is not implemented
502	Remote service failed with code <Code>: <Msg>	A request to the remote device failed with an error.
503 ⁽¹⁾	Not available: <Reason>	The service is not available
504	Remote service timeout	The remote device did not respond within the timeout.
507	Insufficient storage	The memory / storage of the device is insufficient
901	Already exists	The element already exists
902	Wrong chunk size	The chunk size is invalid / too big.

⁽¹⁾ Detailed device error codes need to be read by service [Read diagnostics from the IO channel](#) .

15 Error codes of the device

Device error codes can be received from device by running service -> [Read diagnostics from the IO channel](#).
The device error codes are transferred in the "data" element of the service.

```
"data":  
{  
  "count": 2,  
  "codes": ["F4FE9005","F4FE9005"]  
}
```

Subelement "count" show how many error codes are present.

Subelement "codes" show the code itself with 8 characters, where 2 characters represent a hexadecimal coded number

If more diagnostic events are available, the channel can transfer up to 4 diagnostics simultaneously.

Examples:

- Command response of the compact device for diagnostic read over bit "DR":
5101 0001 0000 **F4FE9000**
- Command response of the compact device for command "GET IDENT DIAGNOSIS"
0880 0008 62C8 0001 **F4FE9000**

15.1 Error group ID tag (F1FE)

Error group	Error Code	Description
ID Tag	F1FE0200	ID tag presence error or R/W head communication error with the ID tag
ID Tag	F1FE0300	Address or command does not fit the ID tag characteristics, memory size invalid
ID Tag	F1FE0400	ID tag is defective, replace ID tag or battery
ID Tag	F1FE0500	ID tag memory overflow. EPC > 16 bytes
ID Tag	F1FE0900	Command not supported by the ID tag
ID Tag	F1FE0A00	Access violation, e.g. block locked. Refer to ISO18000-x
ID Tag	F1FE0B00	General ID tag error which is not specified in detail
ID Tag	F1FE0C00	Unknown internal error

15.2 Error group compact device (F4FE)

Error group	Error Code	Description
Compact device	F4FE0100	Power supply failure
Ident Unit	F4FE0200	Hardware failure , short circuit and overload
Compact device	F4FE0201	Allowed temperature exceeded
Compact device	F4FE0300	Read-/Write head not operating cause time out occurred
Compact device	F4FE0400	Command buffer overflow IO-Server Queue (Internal error)
Compact device	F4FE0500	Data buffer overflow, memory allocation (Internal error)
Compact device	F4FE0600	Command in this mode not supported (Internal error)
Compact device	F4FE8100	ID-Link Master inactive. e.g. after power (Internal error)
Compact device	F4FE8200	Internal IO-Port server error (Internal error)
Compact device	F4FE8300	IO-Port invalid parameter Internal error, e.g. channel (Internal error)
Compact device	F4FE8400	Vendor specific error on PUT
Compact device	F4FE8500	IO-Port server resets channel
Compact device	F4FE8600	Data not available for delayed C/Q inputs or delayed EPC (Internal error)
Compact device	F4FE8700	IO-Port channel reconfiguration not allowed yet (internal error)
Compact device	F4FE8800	IO-Port parameter selector flag not set (internal error)
Compact device	F4FE8900	General error detected from ID-Link Master
Compact device	F4FE8A00	CRC error detected from ID-Link Master
Compact device	F4FE8B00	Object not found detected from ID-Link Master
Compact device	F4FE8C00	Data read/write size within command not valid
Compact device	F4FE8D00	IO-Port channel is reconfigured
Compact device	F4FE8E00	Read-/Write head could not process command e.g. Read/write length exceeded, ID tag memory error, write to locked block
Compact device	F4FE8F00	ID tag data length exceed (Block size * Block number)
Compact device	F4FE9001	Short circuit at output driver detected
Compact device	F4FE9002	Under voltage at output driver detected
Compact device	F4FE9003	Overload at output driver detected
Compact device	F4FE9004	Over temperature at output driver detected
Compact device	F4FE9005	Line break to Read-/Write head
Compact device	F4FE9006	Upper limit reached at output driver
Compact device	F4FE9007	Under voltage at C/Qo detected
Compact device	F4FE9008	Read-/Write head failure detected
Compact device	F4FE9009	Read-/Write head communication error
Compact device	F4FE900A	I ² C communication error (Internal error)
Compact device	F4FE900B	I ² C communication parity error (Internal error)
Compact device	F4FE900C	Command rejected cause antenna field switched off
Compact device	F4FE900D	Internal data of PROFNET stack corrupt (Internal error)
Compact device	F4FE900E	R/W head do not support this object
Compact device	F4FE9401	Frontend Error detected by Read-/Write head
Compact device	F4FE9402	General error detected by Read-/Write head
Compact device	F4FE9403	ID-Link Error detected by Read-/Write head
Compact device	F4FE9404	Buffer overrun Error detected by Read-/Write head

Compact device	F4FE9405	Over temperature at front end detected
Compact device	F4FE9406	R/W head error detect reverse power to high
Compact device	F4FE9407	Reset of R/W head detected
Compact device	F4FE9408	R/W head HAL error detected
Compact device	F4FEA000	Invalid command code detected
Compact device	F4FEA001	Invalid command parameter detected
Compact device	F4FEA002	Invalid command data detected
Compact device	F4FEA003	Ticket number or ticket length detected

Error group	Error Code	Description
Compact device	F4FEA100	Configuration of compact device failed (CR1 / CR2)
Compact device	F4FEA200	Configuration of IO-channel failed (Internal error)
Compact device	F4FEA300	Reading of Inputs C/Qi / IQ (Internal error)
Compact device	F4FEA400	Write of output C/Qo failed (Internal error)
Compact device	F4FEA500	Setting of high current failed (Internal error)
Compact device	F4FEA600	Read of EPC failed (Internal error)
Compact device	F4FEA700	Read of User data memory of the ID tag failed (Internal error)
Compact device	F4FEA800	Write to user memory of the ID tag failed, command WU (Internal error)
Compact device	F4FEA900	Write to user memory of the ID tag failed, command WV (Internal error)
Compact device	F4FEAA00	Verification of the user memory of the ID tag failed, commands "WV" (Internal error)
Compact device	F4FEAB00	Setting of Antenna field on/off failed, command "AN"
Compact device	F4FEAC00	ID-Link master could not read the ID tag blocks (Internal error)
Compact device	F4FEAD00	Block size/number of blocks could not be read from the ID tag

Error group	Error Code	Description
Compact device	F4FED100 ... F4FEFF00	Internal command execution failure

15.3 Error group Communication User – compact device (F5FE)

Error group	Error Code	Description
Communication User - Compact device	F5FE0800	Command from another user being processed (indicated by compact device)
Communication User - Compact device	F5FE8000	More than one command requested by User (DR,WR,Diag)
Communication User - Compact device	F5FE8100	Synchronous read or write command is tried to abort
Communication User - Compact device	F5FE8300	Asynchronous read command parameter invalid
Communication User - Compact device	F5FE8400	Invalid command request in module RWH_CMD detected

Communication User - Compact device	F5FE8500	Module size too short to execute commands
--	----------	---

15.4 Error group Command error – compact device (F6FE)

Error group	Error Code	Description
Communication User - Compact device	F6FE0300	Invalid command parameter (e.g. data range) (indicated by compact device)

16 Glossary

Definition	Remark
Antenna	RFID antenna built in a Read-/Write head.
Asynchronously	Data of the command response is updated after the compact device detects a state change of the ID tag from "not present" to "present" or vice versa.
Block size	Size of one block of the ID tag, for UHF ID tags always 2 bytes.
Compact device	RFID Identification unit DTE801, DTE802, DTE804 with integrated Read-Write head.
Connection	Describes the logical connection between two objects, e.g. Controller and slave.
Controller	See definition PLC.
Emergency system	Web server with reduced functionality to download the firmware of the compact device.
EPC	Electronic product code
ETSI	European Telecommunications Standards Institute. Responsible for standardization of Information and Communication in Europe.
FCC	Federal Communications Commission. Responsible for standardization of Information and Communication in North America.
Hexadecimal	Numeral format, which use 16 values to represent a numeric value. 0..9, A, B, C, D, E, F. In JSON format 2 characters are needed to transmit a hex value.e.g.: 0x51 == "51"
ID Tag, transponder	RFID ID tag, e.g. E80360, E80370.
IoT Webserver	Recipient of a IOT core service, which was subscribed e.g.: PC or ERP server
IoT client	Provider of the IoT services. e.g.: DTE105
Memory bank	The memory of the UHF ID is separated into 4 memory banks. EPC = Contain EPC data. Read-/write access. TID = Contain TID data. Read access only. User = Contain User data. Read-/write access. Reserved = Access and kill password data. Read-/write access.
N	Selected module size of the IO-channel.
PC	Personal computer, e.g. desktop computer, notebook.
PermData	Nonvolatile data area of the compact device for storage of user specific settings, like fieldbus parameter, address settings and so on.
PLC	Programmable Logic Controller, e.g. Allen Bradley Compact Logix, Beckhoff CX5020, Siemens CPU 315-2 DP/PN.
Process data input image	Data area where the PLC can read the outputs of the external periphery devices. (%IBx)
Process data output image	Data area where the PLC can write to the inputs of the external periphery devices. (%QBx)
Read-/Write head	RFID Read-/Write head. In compact devices already integrated.

RSSI	Receive Signal Strength Indicator. Reflect the received signal quality of the ID tag. Higher values mean a better reception of the ID tag signal.
Synchronously	Data of the command response is updated immediately with the currently detected state of the ID tag.
TID	Transponder Identifier. Unique number of the ID tag.
UHF	Ultra High Frequency. Frequency range 300MHz 3GHz ETSI range: 865 ... 868 MHz FCC range: 902 ... 928 MHz
User data	Data area of the ID tag which can be read and written randomly
Web client	PC program to send "http protocol" requests, e.g. Firefox, Internet Explorer
Web server	Built in "http protocol" server to service request from a PC

Reference:

- [1] EPC™ Radio-Frequency Identity Protocols Generation-2 UHF RFID, Specification for RFID Air Interface Protocol for Communications at 860 MHz – 960 MHz Version 2.0.0 Ratified